ONEDEMO-α: FROM SPATIAL REASONING TO ONE-SHOT ROBOT SKILL TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

Imitation learning and pretrained robot models pursue generalization through large-scale training, yet rapidly transferring a newly demonstrated skill across objects and goals remains challenging. Vision-language models (VLMs) with strong spatial reasoning offer a complementary route to adaptation, but repeated inference and action generation during execution can incur substantial latency. Our key insight is to combine reusable motion experience from a demonstration with a spatially proficient VLM's ability to determine how that motion should change. We introduce ONEDEMO-α, a framework for one-shot skill transfer without task-specific training. From one RGB-D demonstration with synchronized robot poses, the VLM predicts an editable skill-adaptation vector α for the target scene and goal. Its functional bindings, motion assignments, and sparse edits guide a geometric executor, preserving reusable motion while concentrating inference on transfer decisions. We further introduce OneDemo-Bench, which systematically varies object position, appearance, size, morphology, and task requirements to evaluate transfer from a fixed demonstration. Experiments show that ONEDEMO-α achieves higher transfer success than trajectory-transfer baselines and substantially lower reasoning overhead than direct VLM control, enabling efficient one-shot skill reuse without task-specific training.

1 INTRODUCTION

A robot taught a skill through one demonstration should be able to apply it to different objects and goals (Figure 1(a)). Toward this goal, imitation learning acquires behavior from examples, while pretrained robot policies broaden generalization through diverse training data (Kim et al., 2025; Black et al., 2025). However, a newly supplied demonstration poses a more immediate question: how should its motion change for a new interaction? For example, after observing a vessel pour into a bowl, a robot may need to grasp a differently shaped container and pour into a new recipient. Transfer therefore requires coordinated changes to grasping, delivery, and pouring.

Existing approaches address this problem through different sources of adaptation. Policies conditioned on demonstrations acquire this ability through prior robot training (Duan et al., 2017; Finn et al., 2017); similarly, multimodal prompting relies on a learned action policy (Jiang et al., 2023). In contrast, explicit transfer anchors demonstrated motion to object poses (Vitiello et al., 2023) or visual correspondences (Heppert et al., 2024; Di Palo & Johns, 2024), as illustrated in Figure 1(b). Functional representations further extend this approach to semantic refinement and constrained motion adaptation across tools (Tang et al., 2025). Although these methods establish the value of reusable motion, transfer still depends on deciding how that motion should change with functional geometry and task goals. VLMs offer a complementary route through spatial affordance grounding (Fang et al., 2024; Yuan et al., 2025), relational constraints (Huang et al., 2025), and demonstration-guided action generation (Cheng et al., 2026). However, when a VLM directly controls execution, repeated observation and reasoning introduce inference overhead (Figure 1(c)). This raises a focused question: *how can spatial reasoning efficiently adapt demonstrated motion to new objects and goals?*

Our insight is that a demonstration already supplies much of what execution requires: concrete motion, operation order, and gripper events. Rather than regenerating these details, a VLM can focus its spatial reasoning on what must change. Specifically, it must identify corresponding functional parts, reason about their relative geometry, and select the motion phases that require adjustment.

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.

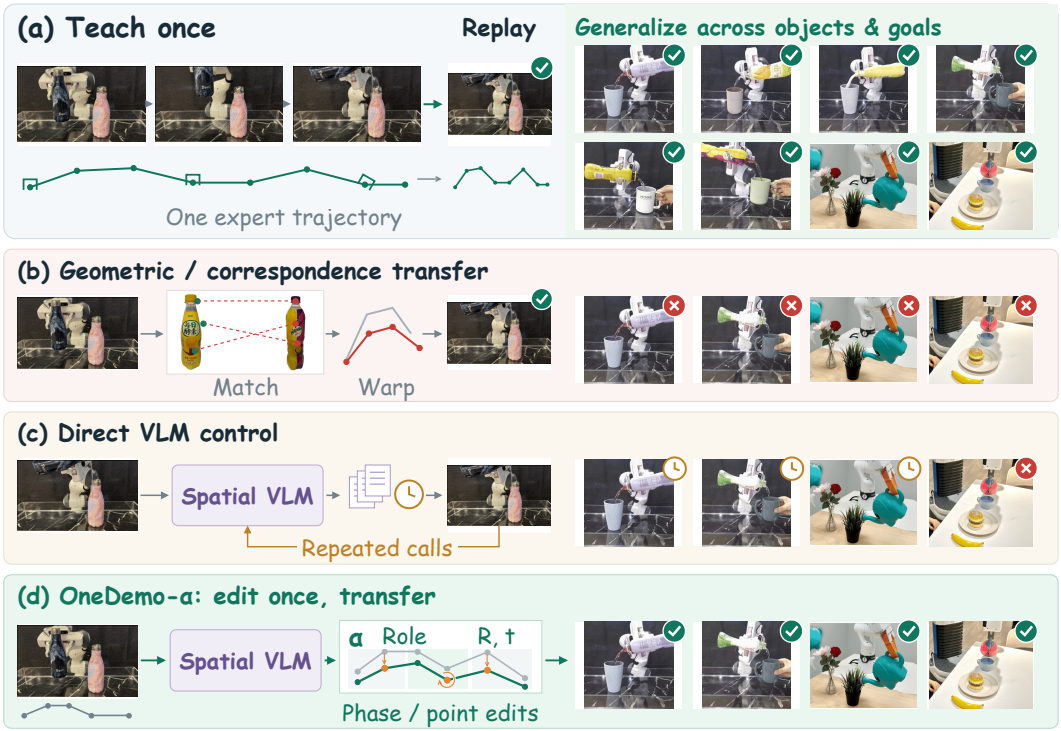


Figure 1: **One demonstration, different routes to transfer.** (a) The goal: reuse a demonstrated skill across objects and goals. (b) Correspondence-based motion transfer. (c) Direct VLM control with repeated inference. (d) ONEDEMO-α: a spatial VLM adapts motion through the structured parameter vector α. Symbols illustrate transfer goals, failure cases, and inference cost.

Recognizing an object's use alone is therefore insufficient: a new handle and receiving opening must lead to appropriate grasp and delivery poses. We consequently pair the demonstrated motion with a VLM capable of making these spatial judgments.

Building on this insight, we introduce ONEDEMO-α for one-shot skill transfer without task-specific training (Figure 1(d)). Given an RGB-D robot demonstration with synchronized poses, a target scene, and an instruction, rigid alignment initializes the recorded motion in the new scene. A frozen spatial VLM predicts α, an explicitly editable skill-adaptation vector encoding functional bindings, motion-to-reference assignments, and sparse edits to trajectory points. Depth grounds these references, and a geometric executor realizes the edits while preserving unedited local motion. For pouring, the new handle guides grasp orientation, while the recipient determines delivery position. The source motion supports repeated transfer without further demonstrations or training.

To evaluate this capability, we introduce OneDemo-Bench, pairing a fixed demonstration with changes in object pose, appearance, geometry, and task requirements. These settings distinguish visual robustness from functional transfer. Model comparisons examine spatial reasoning; success and inference cost measure the benefit of motion reuse. Our contributions are:

- **One-shot reuse.** We transfer one demonstrated trajectory across changed objects and goals without collecting task-specific training datasets or updating model weights.
- **Editable representation.** We introduce the structured skill-adaptation vector α, connecting functional bindings and indexed motion edits to deterministic geometric execution.
- **Transfer evaluation.** We introduce OneDemo-Bench to distinguish replay from functional transfer; comparisons reveal how motion reuse affects transfer success and inference cost across changed objects and goals.

2 Related Work

Learning to use demonstrations. One-shot imitation transfers knowledge acquired across training tasks to a new example, through policy adaptation (Finn et al., 2017; Yu et al., 2018) or demonstration-conditioned inference (Duan et al., 2017; Huang et al., 2019a). Multimodal prompts extend the task context (Jiang et al., 2023); pretrained action policies (Ghosh et al., 2024; Kim et al., 2025) and demonstration-conditioned world models (Zhou et al., 2026a; Chen et al., 2026) supply broader behavior priors. These routes learn an action-producing model before deployment. We instead investigate how a frozen general-purpose VLM can reason over a new demonstration while retaining its measured motion as the executable reference.

Transferring motion through objects and functions. Explicit transfer replaces task-specific policy fitting with task-relative geometry: demonstrated motion follows a new measurement of the relevant object or functional part (Manuelli et al., 2019; Gao et al., 2023). This shifts the difficulty toward finding a reference that remains meaningful under scene and shape changes. Visual correspondence supports alignment (Vitiello et al., 2023; Di Palo & Johns, 2024) and transfer between interaction frames (Heppert et al., 2024); functional alignment can further incorporate semantic correction and trajectory optimization (Tang et al., 2025). Alternatively, correspondence can condition a trained action predictor (Fang et al., 2025; She et al., 2026). The remaining design choice is how correspondence constrains the target action. We let the VLM jointly infer the functional reference and the changes required in the demonstrated motion, encoding both in the editable adaptation vector α.

Turning model reasoning into robot behavior. Language-driven robotics first connected model decisions to available skills (Ichter et al., 2023) and executable programs (Liang et al., 2022; Singh et al., 2023). Finer spatial interfaces express decisions as value maps (Huang et al., 2023a), keypoint constraints (Huang et al., 2025), or visual interaction locations (Fang et al., 2024; Yuan et al., 2025). These interfaces expose increasing amounts of geometry to model reasoning. Visual path prediction (Li et al., 2025) and contextual action tools with execution feedback (Cheng et al., 2026) further connect that reasoning to motion. OneDemo-α studies a complementary interface: the model receives measured motion from one example and reasons about its use in a changed interaction. Its predictions determine the adaptation, while demonstrated details remain available without being generated again. Additional context on motion and policy representations appears in Appendix H.

3 OneDemo-α

3.1 Problem Formulation

We transfer one demonstrated interaction to changed objects, scenes, and task requirements without updating model weights. Given demonstration D, target observation O, and instruction ℓ, we seek executable commands Y:

$$Y = \mathcal{F}(D, O, \ell), \qquad D = (V, H, S), \qquad S = \{(T_i, g_i, \tau_i)\}_{i=1}^{N}. \tag{1}$$

Here V is RGB video, H is calibrated initial depth, and S contains N synchronized end-effector poses $T_i \in \mathrm{SE}(3)$, gripper values g_i, and timestamps τ_i. Each arm has such a stream. O provides target RGB-D, calibration, and robot state. Source and target share an embodiment. Functional parts and action stages are inferred, not annotated inputs; target expert actions and simulator object states are withheld. Figure 2 shows the procedure $\mathcal{F}$.

3.2 Initialize a Transferable Motion Reference

The demonstration records motion and gripper events (Figure 2(a)). Before asking a VLM to modify it, we align this motion with the target object, providing a concrete geometric starting point.

We retain addressed poses that preserve motion and gripper transitions within reconstruction tolerances. RGB correspondences lifted through depth give source points x_j, target points y_j, and confidence weights w_j. A rigid fit initializes every retained pose:

$$(R, t) = \arg\min_{R \in \mathrm{SO}(3),\, t \in \mathbb{R}^3} \sum_j w_j \|R x_j + t - y_j\|^2, \qquad \bar{T}_i = G T_i, \quad G = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}. \tag{2}$$

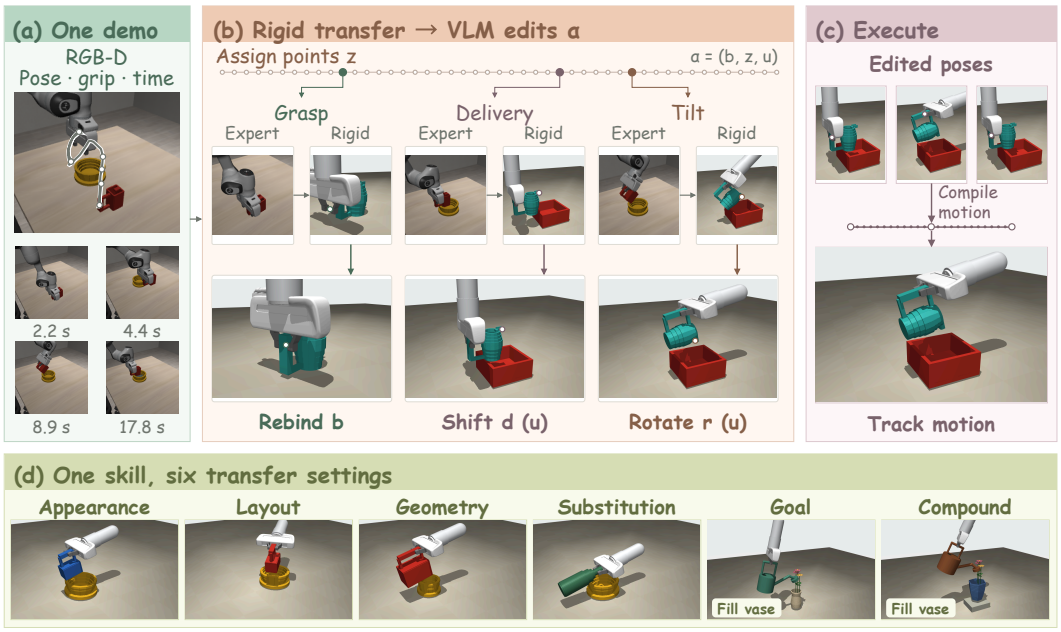


Figure 2: **From rigid initialization to editable transfer.** (a) Recorded motion supplies an indexed reference. (b) After rigid alignment, the VLM predicts the adaptation vector α: bindings b, assignments z, and edits u. Paired scenes show the reference and the effect of each edit. (c) Compilation produces tracked commands. (d) Six settings probe reuse of the skill.

Weighted alignment estimates rotation R and translation t. The initialized reference $\bar{S}$ preserves source indices, timing, and gripper commands. It follows the aligned object, but cannot by itself resolve a changed handle or independently moved recipient.

3.3 Represent Adaptation with α

A new grasp and a redirected pour require different references. We parameterize these decisions with an explicitly editable skill-adaptation vector α:

$$\alpha = f(D, O, \bar{S}, \ell) = (b, z, u), \qquad u = [(i, o, v), \ldots]. \tag{3}$$

The frozen VLM f receives sampled video frames, indexed source and initialized poses, target images, and coordinate conventions; depth grounds its predictions. α is a structured parameter vector with three variable-length blocks, combining discrete selectors with continuous geometric, gripper, and timing parameters:

Bindings b: where to anchor. Paired source and target regions, three surface landmarks, a frame type, and an origin offset identify functional references, such as a handle or receiving opening. Segmentation (Ravi et al., 2024) and depth convert them into metric frames.

Assignments z: which points to bind. Source-indexed intervals select a binding for each motion phase. The response also chooses the execution entry and arm assignment. Grasping can follow the tool while delivery follows the recipient; phase names describe inferred intervals.

Edits u: what to change. Each record in u selects source point i, operation o, and arguments v. Operations include offset, replacement, insertion, deletion, gripper adjustment, and timing. Arguments use the assigned local frame. Every retained point is addressable; unselected points preserve their source-local motion. Figure 2(b) illustrates rebinding, translation, and rotation.

3.4 Compile Edits into Executable Motion

The blocks of α specify where motion is anchored and how it changes. The compiler $\mathcal{C}$ produces $Y = \mathcal{C}(\bar{S}, \alpha)$ without further VLM inference (Figure 2(c)).

Rebind. For point i, its assignment z_i selects source and target frames A and B. Three lifted landmarks define oriented axes; a gravity-aligned option fixes the vertical axis. Rebinding corrects the initialized pose through

$$L_i = (GA)^{-1}\bar{T}_i = A^{-1}T_i, \qquad \hat{T}_i = BL_i. \tag{4}$$

Here L_i is the source-local pose and $\hat{T}_i$ its rebound pose. If $B = GA$, this operation returns $\bar{T}_i$: rigid initialization is retained exactly.

Offset. For local rotation Q_i and position p_i, the six arguments $v = (d_i, r_i)$ specify translation d_i in meters and rotation vector r_i in radians:

$$p'_i = p_i + d_i, \qquad Q'_i = \mathrm{Exp}([r_i]_\times)Q_i, \qquad T'_i = B(Q'_i, p'_i). \tag{5}$$

$[\cdot]_\times$ forms a skew matrix and Exp converts it to a rotation; $B(Q'_i, p'_i)$ composes the target frame and edited local pose. Shift and Rotate in Figure 2(b) are the two components of this operation. Rotation acts about the end-effector origin, so holding an outlet fixed requires a coupled translation.

Restructure and schedule. Replace and Insert take a local position, unit quaternion, gripper value, and minimum duration; insertion follows the addressed point. Delete removes that point. Grip replaces $g_i \in [-1, 1]$; Hold sets a minimum segment duration in seconds. Original indices remain stable after edits. The compiler validates the parameters, connects the current state to the selected entry, and constructs synchronized pose and gripper commands. Low-level feedback tracks this motion. Appendix A specifies frame construction and temporal realization.

3.5 Relation to VLA and WAM

Pretrained VLAs and WAMs supply action priors, sometimes conditioned on demonstrations (Physical Intelligence et al., 2025; Zhou et al., 2026a). OneDemo-α reuses a frozen pretrained spatial VLM and one recorded trajectory, requiring neither additional large-scale robot dataset collection nor end-to-end VLA/WAM pretraining or fine-tuning. The source supplies motor experience; α adapts it across objects and goals. A native VLM agent's successful exploration could also supply the reference, given recorded RGB-D and synchronized robot states. This separates expensive skill acquisition from repeated transfer. The same interface supports different spatial models, allowing their grounding and motion-editing capabilities to be compared without additional robot training.

4 OneDemo-Bench

4.1 Fixed-Demonstration Transfer

Building on manipulation benchmarks (Yu et al., 2020; James et al., 2019; Mees et al., 2021; Gu et al., 2023; Liu et al., 2023; Zhou et al., 2025; Chen et al., 2025; Nasiriany et al., 2024), OneDemo-Bench pairs one successful demonstration per task with changed objects, layouts, and interactions under the same embodiment. Trajectory methods use the inputs in Section 3.1; native policies retain their interfaces. Target expert actions and evaluator state are withheld.

4.2 Six Transfer Settings

Appearance changes visual properties, preserving geometry and intent. *Layout* changes object poses. *Geometry* changes dimensions or functional parts within a role class. *Substitution* replaces a role's instance or category. *Goal* changes the recipient, contact, direction, or use. *Compound* combines independently specified factors. These settings separate perceptual robustness, contact adaptation, and functional reuse (Zhou et al., 2026b), without imposing a difficulty ordering.

Conditions record affected roles and factors. Shape changes inherent to substitution are not independent perturbations; goal comparisons hold the scene fixed. Controls vary neither role, either role, or both. Source replay checks the interface; independent expert rollouts establish target feasibility (Mandlekar et al., 2023). Eligibility is independent of method success.

Figure 3 covers 30 tasks: 12 LIBERO-Pro, 14 RoboTwin, and four RoboCasa tasks, with 456, 615, and 20 configurations. The 1,091 entries include 48 configurations from eight tasks, including eight



Figure 3: **OneDemo-Bench overview.** (a) Vocabulary of task queries and asset names, sized by frequency. (b) Interaction keyframes from 30 source tasks. (c) Distribution of six transfer settings over 2,022 recorded taxonomy labels; configurations can carry multiple labels.

source settings. Evaluation eligibility requires source replay and target-feasibility checks. RoboCasa is an outcome-informed development set with 12 OneDemo-α successes. Taxonomy statistics use configurations with complete annotations (Appendix B).

Physical outcomes. Success requires the intended interaction and terminal state. Pouring uses particle-fluid dynamics (Macklin & Müller, 2013), with a separate criterion for rigid dumping. Methods share eligible targets and controls; complete three-layout groups measure robustness. Native outcomes remain separate. Appendix C defines these measures and their timing conventions.

5 Experiments

We examine whether one demonstrated motion supports functional transfer, how the spatial reasoner affects adaptation, and how motion reuse changes inference cost. Pretrained policies and physical trials provide complementary system references.

5.1 Experiment Setting

Baselines. We compare with correspondence-based DITTO (Heppert et al., 2024), function-aware MimicFunc (Tang et al., 2025), and direct-control Native Astra. Both Ours and Native Astra use Astra with low reasoning effort (Astra low). Pretrained $\pi_{0.5}$, Zero-WAM, and Faster-WAM (Physical Intelligence et al., 2025; Zhou et al., 2026a; Zhao et al., 2026) serve as native policy references.

Metrics. SR measures episode success (%) on shared targets; R3 requires success across three layouts. Infer. measures model-request time (s), while target time also includes observation, geometry, and execution. Tokens report target language-model usage in thousands; they do not apply to continuous policy outputs. Costs include failures and use complete paired records.

Table 1: **Transfer and reasoning cost.** Category SR (%) across six settings, with inference time (s) and target tokens (k). Native Astra uses single-response transfer for SR and multiround control for costs. Source preparation is accounted separately.

	SR (%) ↑						Cost ↓	
Method	Appear.	Layout	Geom.	Subst.	Goal	Compound	Infer.	Tokens
Ours	**63.7**	**65.9**	**56.6**	**38.8**	**35.2**	**59.8**	43.8	6.7
DITTO	41.8	45.1	32.2	11.8	11.4	38.2	0.0	0.0
MimicFunc	23.4	25.8	18.8	6.5	6.8	21.6	65.8	10.5
Native Astra	26.7	28.4	23.0	9.4	9.1	25.6	315.2	49.3

Implementation details. Trajectory methods receive one RGB-D demonstration with synchronized robot poses and gripper states. DITTO and MimicFunc use robot-demonstration adaptations. Ours predicts one adaptation vector per target; Native Astra uses demonstration RGB and observation

Table 2: **Spatial reasoners across six settings.** Category SR (%), mean inference time (s), and target tokens (k). All reasoners share the same transfer interface and targets. DITTO, MimicFunc, and Native Astra provide method controls.

	SR (%) ↑						Cost ↓	
Method	Appear.	Layout	Geom.	Subst.	Goal	Compound	Infer.	Tokens
Ours	80.6	83.3	**75.0**	**66.7**	**61.1**	**58.3**	32.4	8.1
Sol	61.1	66.7	47.2	30.6	27.8	22.2	25.8	7.3
Fable	69.4	72.2	55.6	44.4	36.1	30.6	24.9	11.9
Opus	69.4	69.4	58.3	44.4	38.9	33.3	13.6	24.7
Qwen	52.8	55.6	41.7	30.6	25.0	19.4	48.2	13.4
Gemini	72.2	75.0	61.1	47.2	41.7	36.1	18.9	17.8
DITTO	66.7	**86.1**	47.2	25.0	16.7	19.4	0.0	0.0
MimicFunc	41.7	52.8	27.8	13.9	11.1	11.1	67.5	10.8
Native Astra	**83.3**	80.6	72.2	58.3	58.3	50.0	230.0	60.8

feedback. Target identities and outcome criteria are fixed within each comparison. Appendix D specifies baseline adaptations, native policy interfaces, evaluation populations, and cost accounting.

5.2 Main Results

One-shot motion reuse. Ours leads all six categories in the common-target comparison (Table 1), improving over DITTO by 20.8–27.0 percentage points. The benefit extends beyond Layout to changed interaction objects and goals, supporting the use of spatial reasoning to adapt demonstrated motion rather than requiring a new task-specific policy.

Table 1 evaluates Native Astra in a single-response transfer setting; its cost columns measure multi-round control. Category tags can overlap; Appendix E.1 defines the evaluation sets.

Functional adaptation. Table 3 and Figure 4(b) summarize transfer. On Goal, Ours exceeds DITTO by 37.5 points and Native Astra by 15.9 points. Geometry alone cannot determine how a new recipient or intended use changes the interaction. The bindings and indexed edits in α provide a way to express those changes while retaining demonstrated motion. Compound settings preserve a 14.3-point advantage over Native Astra, whereas the smaller 5.8-point Substitution margin identifies replacement objects as a harder case. These aggregate patterns show the transfer behavior of the combined binding and motion-editing interface.

Table 3: **Transfer by setting.** SR (%) evaluated on identical targets across methods.

Setting	Ours	DITTO	MimicFunc	Native Astra
Appearance	**75.1**	45.1	27.8	61.2
Layout	**76.9**	48.5	28.8	62.1
Geometry	**77.0**	47.6	29.1	63.6
Substitution	**68.2**	34.1	21.8	62.4
Goal	**69.3**	31.8	15.9	53.4
Compound	**73.1**	42.2	25.2	58.8

5.3 Ablation Study

Spatial reasoning matters. With the transfer interface and targets fixed, Ours with Astra reaches 70.8% macro SR versus 55.6% for Gemini and 37.5% for Qwen (Table 2; Figure 4(c)). Its advantage over Gemini more than doubles from Appearance to Substitution. This pattern is consistent with the need to connect functional parts to metric motion edits, beyond recognizing objects. Response reliability and reasoning settings also vary, so these results reflect their joint effect on adaptation.

Reuse reduces reasoning. Native Astra uses 7.2 times the inference time and 7.4 times the tokens of Ours (Figure 4(a)). Reusing the demonstrated motion lets the VLM specify changes once instead of repeatedly generating actions. DITTO incurs no target VLM calls, but its lower transfer success exposes the tradeoff between fixed geometric alignment and semantic adaptation. These costs measure reasoning overhead, while physical execution also depends on the motion and controller.

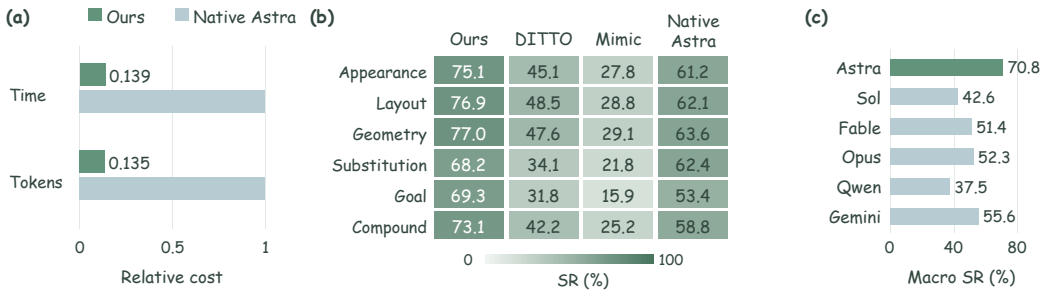


Figure 4: **Reasoning and transfer.** (a) Relative inference and token cost. (b) Category SR (Table 3). (c) Six-setting macro SR (Table 2).

Table 4: **Policy references on a compatible subset.** SR (%), inference time (s), and tokens (k) on 128 RoboTwin-derived cases supported by Zero-WAM. Upper: native pretrained references. Lower: transfer and direct control.

	SR (%) ↑						Cost ↓	
Method	Appear.	Layout	Geom.	Subst.	Goal	Compound	Infer.	Tokens
$\pi_{0.5}$	87.5	81.8	70.0	61.9	52.6	54.5	21.4	—
Faster-WAM	**91.7**	**90.9**	75.0	57.1	47.4	50.0	14.6	—
Zero-WAM	79.2	77.3	65.0	47.6	42.1	40.9	126.8	—
DITTO	75.0	**90.9**	55.0	33.3	26.3	27.3	0.0	0.0
MimicFunc	45.8	54.5	35.0	19.0	15.8	18.2	71.6	11.2
Native Astra	**91.7**	86.4	80.0	66.7	63.2	59.1	250.6	56.9
Ours	87.5	**90.9**	**85.0**	**76.2**	**73.7**	**68.2**	34.8	7.8

What the controls isolate. ① *Bindings.* Re-anchor holds the model response and grounded references fixed while removing motion edits; Rigid-only also removes rebinding. These controls distinguish functional reference selection from local motion changes. ② *Execution.* An empty-edit check produces identical successful 115-step rollouts, verifying that the editor preserves the reference when no change is requested. This establishes the reference-preservation property of the compiler for an empty edit set. ③ *Source access.* Native Astra receives source RGB, whereas trajectory methods also use measured robot motion. Their comparison evaluates the complete reuse interface, combining access to measured motion with adaptation through α.

5.4 Further Analysis

Pretrained policy references. Table 4 separates pretrained VLA/WAM references from demonstration-based transfer and direct control. The 128-case RoboTwin-derived subset is restricted by Zero-WAM compatibility. Ours exceeds $\pi_{0.5}$ by 14.3 points on Substitution and 13.6 on Compound. Faster-WAM leads on Appearance and has lower inference cost, showing the benefit of a learned action prior in compatible scenes. Our advantage concerns adapting a supplied motion to changed interactions without additional task-specific robot data or weight updates. Native policy inputs and prior training remain distinct.

When geometry is insufficient. A rigid transformation preserves the demonstrated relation to an object, but cannot decide whether a new goal requires another contact region or motion endpoint. Functional bindings address the former; indexed pose edits address the latter. This distinction explains why Layout and Goal probe different capabilities, and why Compound changes require both operations within a shared trajectory. For example, moving a recipient changes where delivery should end, while replacing a handled vessel can also change the grasp frame and wrist orientation. The source still supplies the ordering, timing, and gripper events of the interaction. Encoding the required changes in α therefore connects task intent to executable motion while retaining the demonstrated structure that does not need to be regenerated. Reasoning is thus concentrated on the changes needed for transfer rather than on reconstructing the entire action sequence.

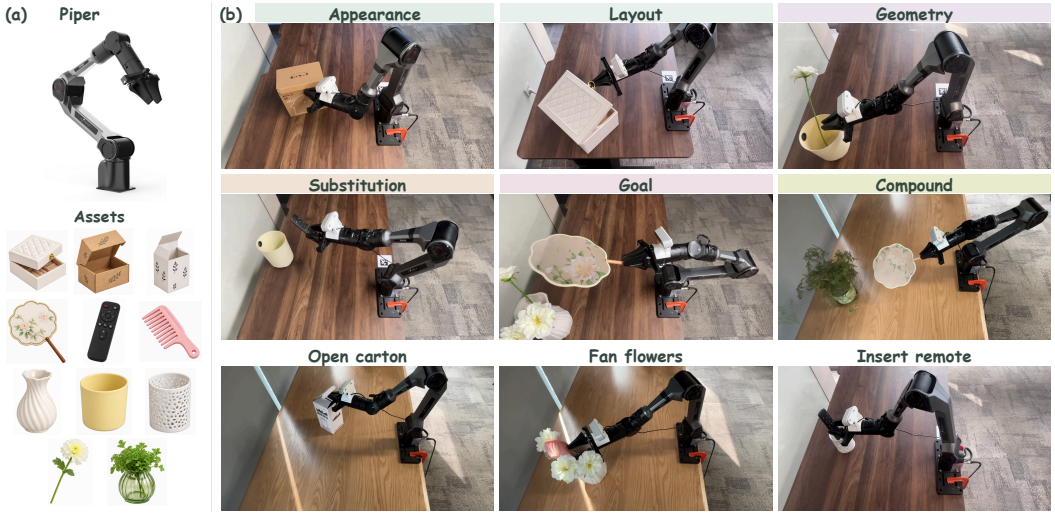


Figure 5: **Real-robot transfer.** (a) Piper and reconstructed asset illustrations. (b) Nine recorded configurations spanning box opening, fanning, and insertion. The first six exemplify the transfer taxonomy; factors can co-occur.

Physical skill transfer. Piper reuses demonstrations for box opening, fanning, and insertion (Figure 5(b)). Table 5 distinguishes accurate replay from adaptation: DITTO reaches 83.3% on replay but 36.7% on transfer, compared with 80.0% and 60.0% for Ours. Spatial edits therefore matter most when the interaction changes. Native Astra reaches 53.3% transfer SR at 238.6 seconds, versus 46.2 seconds for Ours. This connects motion reuse to a practical success–time tradeoff. The upper $\pi_{0.5}$ row provides a pretrained policy reference evaluated through its native observation and action interface.

Table 5: **Piper transfer.** SR (%) and target time (s); the first row is a pretrained policy reference.

	Replay		Transfer	
Method	SR ↑	Time ↓	SR ↑	Time ↓
$\pi_{0.5}$	73.3	45.9	43.3	65.8
DITTO	**83.3**	**28.6**	36.7	**41.5**
Native Astra	76.7	96.4	53.3	238.6
Ours	80.0	34.8	**60.0**	46.2

The boundary is physical grounding. Among 22 failed RoboTwin trials in the execution analysis, eight report tracking errors and four fail landmark or mask grounding; the remaining failures concern output validity or unsuccessful interaction. Thus a correct high-level goal is insufficient: selected functional parts must produce feasible poses and continuous commands. This motivates retaining both geometric initialization and explicit local edits, while identifying contact-aware execution as the next source of improvement.

6 Conclusion

OneDemo-α combines reusable demonstrated motion with spatial reasoning for one-shot skill transfer. Its structured adaptation vector α connects functional bindings and indexed motion edits to geometric execution without task-specific training. Across the six settings of OneDemo-Bench, motion reuse improves transfer over correspondence-based baselines while reducing inference time and token usage relative to direct VLM control. Physical experiments show that adapting the demonstrated interaction improves transfer beyond accurate replay. Together, these findings support using spatial reasoning to modify existing skills while preserving reusable motor detail. The present method assumes calibrated depth and a shared source and target embodiment; its single transfer decision also limits adaptation to changes that arise during execution. Future work will use selective visual feedback to revise bindings and motion edits during execution. We will also explore autonomous acquisition of motion references, allowing successful experiences to support repeated skill transfer across new objects and goals.

AI USE STATEMENT

Generative AI assisted with literature retrieval, method formalization, experimental planning, data aggregation and analysis code, result interpretation, manuscript drafting and editing, and illustrative asset reconstruction. Measured performance values derive from evaluation records; task scenes are benchmark inputs or recorded observations. Generated object illustrations depict the asset set only. The authors are responsible for the final content.

REFERENCES

Sai Haneesh Allu, Jishnu Jaykumar P, Ninad Khargonkar, Tyler Summers, Jian Yao, and Yu Xiang. HRT1: One-Shot Human-to-Robot Trajectory Transfer for Mobile Manipulation, 2025. URL https://arxiv.org/abs/2510.21026.

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

Guangyan Chen, Meiling Wang, Te Cui, Zichen Zhou, Qi Shao, Shalfun Li, Hang Su, Roy Gan, Hao Wang, Mengyin Fu, Yi Yang, and Yufeng Yue. HOST:Robots Acquire Manipulation Skills in Seconds from a Single Human Video, 2026. URL https://arxiv.org/abs/2607.20033.

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. URL https://robotwin-platform.github.io/.

Dongzhou Cheng, Taoran Yi, Ye Fang, Xingwu Zhang, Fan Feng, Yixuan Li, Gengxiong Zhuang, Rongze Wang, Shuai Yang, Wei Song, Weizhi Xue, Minyan Wu, Jie Gui, Jiaqi Wang, and Tong Wu. In-Context Robot Learning with VLM Agents, 2026. URL https://arxiv.org/abs/2609.19138.

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, 2023. URL https://arxiv.org/abs/2303.04137.

Norman Di Palo and Edward Johns. DINOBot: Robot Manipulation via Retrieval and Alignment with Vision Foundation Models, 2024. URL https://arxiv.org/abs/2402.13181.

Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-Shot Imitation Learning. In *Advances in Neural Information Processing Systems*, 2017. URL https://arxiv.org/abs/1703.07326.

Kuan Fang, Fangchen Liu, Pieter Abbeel, and Sergey Levine. MOKA: Open-World Robotic Manipulation through Mark-Based Visual Prompting. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.062.

Xiaolin Fang, Bo-Ruei Huang, Jiayuan Mao, Jasmine Shone, Joshua B. Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. KALM: Keypoint Abstraction using Large Models for Object-Relative Imitation Learning. In *IEEE International Conference on Robotics and Automation*, 2025. URL https://kalm-il.github.io/.

Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-Shot Visual Imitation Learning via Meta-Learning. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pp. 357–368. PMLR, 2017. URL https://proceedings.mlr.press/v78/finn17a.html.

Peter R. Florence, Lucas Manuelli, and Russ Tedrake. Dense Object Nets: Learning Dense Visual Object Descriptors By and For Robotic Manipulation. In *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 373–385. PMLR, 2018. URL https://proceedings.mlr.press/v87/florence18a.html.

Jianfeng Gao, Zhi Tao, Noémie Jaquier, and Tamim Asfour. K-VIL: Keypoints-Based Visual Imitation Learning. *IEEE Transactions on Robotics*, 39(5):3888–3908, 2023. doi: 10.1109/TRO.2023.3286074. URL https://publikationen.bibliothek.kit.edu/1000163489.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An Open-Source Generalist Robot Policy. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.090.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. RVT: Robotic View Transformer for 3D Object Manipulation. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 694–710. PMLR, 2023. URL https://proceedings.mlr.press/v229/goyal23a.html.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. ManiSkill2: A Unified Benchmark for Generalizable Manipulation Skills. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2302.04659.

Nick Heppert, Max Argus, Tim Welschehold, Thomas Brox, and Abhinav Valada. DITTO: Demonstration Imitation by Trajectory Transformation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2024. URL https://arxiv.org/abs/2403.15203.

De-An Huang, Suraj Nair, Danfei Xu, Yuke Zhu, Animesh Garg, Li Fei-Fei, Silvio Savarese, and Juan Carlos Niebles. Neural Task Graphs: Generalizing to Unseen Tasks from a Single Video Demonstration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019a. URL https://arxiv.org/abs/1807.03480.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 540–562. PMLR, 2023a. URL https://proceedings.mlr.press/v229/huang23b.html.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner Monologue: Embodied Reasoning through Planning with Language Models. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782. PMLR, 2023b. URL https://proceedings.mlr.press/v205/huang23c.html.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. ReKep: Spatio-Temporal Reasoning of Relational Keypoint Constraints for Robotic Manipulation. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4573–4602. PMLR, 2025. URL https://proceedings.mlr.press/v270/huang25g.html.

Yanlong Huang, Leonel Rozo, João Silvério, and Darwin G. Caldwell. Kernelized Movement Primitives. *The International Journal of Robotics Research*, 38(7):833–852, 2019b. doi: 10.1177/0278364919846363. URL https://arxiv.org/abs/1708.08638.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn,

Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 2023. URL https://proceedings.mlr.press/v205/ichter23a.html.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors. *Neural Computation*, 25 (2):328–373, 2013. doi: 10.1162/NECO_a_00393. URL https://www.pure.ed.ac.uk/ws/portalfiles/portal/7874487/NECO_a_00393.pdf.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The Robot Learning Benchmark & Learning Environment, 2019. URL https://arxiv.org/abs/1909.12271.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. VIMA: Robot Manipulation with Multimodal Prompts. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 14975–15022. PMLR, 2023. URL https://proceedings.mlr.press/v202/jiang23b.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 *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Raymond Yu, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. HAMSTER: Hierarchical Action Models For Open-World Robot Manipulation. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2502.05485.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as Policies: Language Model Programs for Embodied Control, 2022. URL https://arxiv.org/abs/2209.07753.

Haitao Lin, Yanwei Fu, and Xiangyang Xue. Pourit!: Weakly-supervised liquid perception from a single image for visual closed-loop robotic pouring. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. URL https://hetolin.github.io/PourIt/.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning, 2023. URL https://arxiv.org/abs/2306.03310.

Peiqi Liu, Yaswanth Orru, Jay Vakil, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. OK-Robot: What Really Matters in Integrating Open-Knowledge Models for Robotics, 2024. URL https://arxiv.org/abs/2401.12202.

Miles Macklin and Matthias Müller. Position based fluids. *ACM Transactions on Graphics*, 32 (4), 2013. doi: 10.1145/2461912.2461984. URL https://mmacklin.com/pbf_sig_preprint.pdf.

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 *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1820–1864. PMLR, 2023. URL https://proceedings.mlr.press/v229/mandlekar23a.html.

Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kPAM: KeyPoint Affordances for Category-Level Robotic Manipulation, 2019. URL https://arxiv.org/abs/1903.06684.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks, 2021. URL https://arxiv.org/abs/2112.03227.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-Scale Simulation of Household Tasks for Generalist Robots. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.050.

Chuer Pan, Brian Okorn, Harry Zhang, Ben Eisner, and David Held. TAX-Pose: Task-Specific Cross-Pose Estimation for Robot Manipulation. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1783–1792. PMLR, 2023. URL https://proceedings.mlr.press/v205/pan23a.html.

Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. Probabilistic Movement Primitives. In *Advances in Neural Information Processing Systems*, volume 26, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/e53a0a2978c28872a4505bdb51db06dc-Abstract.html.

Claudia Pérez-D'Arpino and Julie A. Shah. C-LEARN: Learning Geometric Constraints from Demonstrations for Multi-Step Manipulation in Shared Autonomy. In *IEEE International Conference on Robotics and Automation*, 2017. URL https://ai.stanford.edu/~cdarpino/CLEARN/ICRA17_DArpino_CLEARN.pdf.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, et al. $\pi_{0.5}$: A Vision-Language-Action Model with Open-World Generalization, 2025. URL https://arxiv.org/abs/2504.16054.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment Anything in Images and Videos, 2024. URL https://arxiv.org/abs/2408.00714.

Qijin She, Hanyang Yu, Zeming Li, and Ping Tan. MatchingPolicy: Correspondence-Aware Policy Enables Cross-Object In-Context Learning, 2026. URL https://arxiv.org/abs/2608.16715v1.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. CLIPort: What and Where Pathways for Robotic Manipulation. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 894–906. PMLR, 2022. URL https://proceedings.mlr.press/v164/shridhar22a.html.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-Actor: A Multi-Task Transformer for Robotic Manipulation. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 785–799. PMLR, 2023. URL https://proceedings.mlr.press/v205/shridhar23a.html.

Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B. Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural Descriptor Fields: SE(3)-Equivariant Object Representations for Manipulation, 2021. URL https://arxiv.org/abs/2112.05124.

Anthony Simeonov, Yilun Du, Yen-Chen Lin, Alberto Rodriguez Garcia, Leslie Pack Kaelbling, Tomás Lozano-Pérez, and Pulkit Agrawal. SE(3)-Equivariant Relational Rearrangement with Neural Descriptor Fields. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 835–846. PMLR, 2023. URL https://proceedings.mlr.press/v205/simeonov23a.html.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. ProgPrompt: Generating Situated Robot Task Plans using Large Language Models. In *IEEE International Conference on Robotics and Automation*, 2023. URL https://arxiv.org/abs/2209.11302.

Chao Tang, Anxing Xiao, Yuhong Deng, Tianrun Hu, Wenlong Dong, Hanbo Zhang, David Hsu, and Hong Zhang. MimicFunc: Imitating Tool Manipulation from a Single Human Video via Functional Correspondence. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 4473–4492, 2025. URL https://proceedings.mlr.press/v305/tang25a.html.

Pietro Vitiello, Kamil Dreczkowski, and Edward Johns. One-Shot Imitation Learning: A Pose Estimation Perspective. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 943–970. PMLR, 2023. URL https://proceedings.mlr.press/v229/vitiello23a.html.

Tianhe Yu, Chelsea Finn, Sudeep Dasari, Annie Xie, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning. In *Proceedings of Robotics: Science and Systems*, 2018. doi: 10.15607/RSS.2018.XIV.002.

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 *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100. PMLR, 2020. URL https://proceedings.mlr.press/v100/yu20a.html.

Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. RoboPoint: A Vision-Language Model for Spatial Affordance Prediction in Robotics. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4005–4020. PMLR, 2025. URL https://proceedings.mlr.press/v270/yuan25c.html.

Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, and Johnny Lee. Transporter Networks: Rearranging the Visual World for Robotic Manipulation. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 726–747. PMLR, 2021. URL https://proceedings.mlr.press/v155/zeng21a.html.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016.

Weiheng Zhao, Haoyi Jiang, Xin Shi, Liu Liu, Fan Huang, Zhizhong Su, Wei Sui, and Xinggang Wang. Faster-WAM: Efficient Inference-Time Future Conditioning for Robust World Action Models, 2026. URL https://arxiv.org/abs/2608.04404.

Jiaming Zhou, Qihang Zhang, Gangwei Xu, Cunxin Fan, Yujie Zhao, Ruilin Wang, Yiming Luo, Shuai Yang, Xing Zhu, Yujun Shen, Junwei Liang, and Yinghao Xu. Zero-WAM: In-Context World-Action Modeling from Human Videos for Open-Ended Task Generalization, 2026a. URL https://arxiv.org/abs/2608.26103.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. LIBERO-PRO: Towards Robust and Fair Evaluation of Vision-Language-Action Models Beyond Memorization, 2025. URL https://arxiv.org/abs/2510.03827.

Xunzhe Zhou, Yiyang Cai, Fengyi Wang, Ran Ju, Hanxiang Ren, Ruizhe Liu, Yu Zhang, Qian Luo, Feng Chen, Pei Zhou, Yi Ma, and Yanchao Yang. The Imitator Game: Benchmarking Robot Imitative Ability Beyond Action Prediction, 2026b. URL https://arxiv.org/abs/2608.22301.

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 *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A Representation and Geometric Semantics

A.1 Source Observations and Simplification

The source frontend reads synchronized robot end-effector poses, gripper values, timestamps, and RGB frames, together with the initial depth image and camera calibration. The visual input to the VLM is a nine-frame summary with timestamps, the initial source view, and the initial target view. Measured source motion and its rigid initialization accompany those images. Calibrated depth supports the initial rigid fit and subsequently grounds the functional references selected in the structured response. Source and edited gripper values are continuous in the normalized range $[-1, 1]$, including partial closure.

For an interval between retained samples, translation and gripper values are interpolated linearly in time and orientation by spherical interpolation. For sample i, the residual is

$$e_i = \max\left(\frac{\|p_i - \bar{p}_i\|}{\epsilon_p}, \frac{d_R(R_i, \bar{R}_i)}{\epsilon_R}, \frac{|g_i - \bar{g}_i|}{\epsilon_g}\right), \tag{6}$$

where bars denote interpolation and d_R is the rotation geodesic angle. The source frontend uses tolerances of $\epsilon_p = 5\,\text{mm}$, $\epsilon_R = 0.06\,\text{rad}$, and $\epsilon_g = 0.03$ in the normalized gripper range. Endpoints and the two neighboring samples of each gripper transition are mandatory. The largest residual determines the next retained sample. Compression that exceeds the tolerance is rejected; a point-count limit does not override the error criterion. All arms share a timeline with their original timestamps.

A.2 Rigid Initialization

The initializer fits one rotation and translation to source–target object correspondences in metric coordinates. It transforms the complete retained motion, preserving sample addresses, timestamps, and gripper values. A rigid fit does not resolve differences in local shape, independently moved recipients, or task intent. The VLM therefore receives both the original and initialized poses, together with the corresponding observations, before proposing functional bindings and motion edits. After a binding is revised, the compiler recomputes the affected entries from their source-relative poses; it does not apply a second global transform to an already transformed trajectory.

A.3 Structured Representation

The skill-adaptation vector $\alpha = (b, z, u)$ is a typed parameterization of changes to the supplied motion reference. It combines discrete reference, point, and operation identifiers with continuous geometric and control values. The numbers of roles and edit records determine its length. The VLM emits these blocks as structured fields, and the compiler interprets each field according to its units and operation semantics.

The binding block b defines paired roles with a region, three surface landmarks, an origin displacement, a frame type, and a shared-object label. The landmarks refer to the initial source or target image, not to the visual summary. Phase starts refer to original reference entries and induce a complete, nonoverlapping role assignment for each arm. Neither phase boundaries nor semantic event

names are provided as input; the VLM infers them jointly with the bindings and edits. A role may own multiple phases. Each arm also has an execution entry and a target arm assignment. Mapping two source arms to the same target arm is invalid.

Table 6 defines the operations that turn predicted adaptations into local motion changes. Every entry in the supplied reference is addressable. A compact entry retains its association with the raw demonstration sample, but raw samples omitted during compression are not separately exposed to the VLM. Edits are evaluated against the original reference rather than an incrementally renumbered list. Multiple insertions after the same entry preserve their response order. A deletion combined with modification of the same entry is rejected. This makes edit meaning independent of changes in sequence length. Validation checks finite values, valid rotations, consistent phases, and bounded durations before commands are sent to the controller.

A.4 Pose Edits and Coordinated Execution

For a local pose $L = (R, p)$, with rotation R and position p, offsets act as

$$p' = p + \delta p, \qquad R' = \mathrm{Exp}([\delta r]_\times)R. \tag{7}$$

Here δp is a translation offset, δr is a rotation vector, $[\cdot]_\times$ is its skew-symmetric matrix, and Exp is the matrix exponential. Primes denote the edited values. Both offsets use the role frame. Rotation modifies the waypoint orientation without rotating its position. A pose replacement or insertion supplies a local position, quaternion, gripper value, and minimum duration. Unchanged entries retain their local poses; their output indices may move after insertion or deletion.

For multiple arms, phase assignments and edits refer to each source stream. An injective mapping assigns moving source streams to target arms; unassigned arms retain their current state. Source timestamps provide a shared schedule and edits impose duration requirements. The selected entry excludes initial home motion, and the controller connects its actual state to that entry. Gravity-aligned roles on a shared rigid body can use paired role origins to coordinate frame directions. These operations preserve the meaning of the model's choices without solving a coupled collision or grasp optimization problem.

A.5 Metric Frame Construction

Let $x_0, x_1, x_2 \in \mathbb{R}^3$ be the back-projected landmarks. Region masks constrain the depth support used around them. For a surface frame, the axes are

$$u = \frac{x_1 - x_0}{\|x_1 - x_0\|}, \quad w = \frac{u \times (x_2 - x_0)}{\|u \times (x_2 - x_0)\|}, \quad v = w \times u. \tag{8}$$

For a gravity-aligned frame, w is world vertical and u is the normalized horizontal projection of $x_1 - x_0$; again $v = w \times u$. Given the model's local displacement d, the origin is $x_0 + [u\ v\ w]d$. The three landmarks encode the functional origin and oriented axes used to express local motion. Axis signs must be consistent across source and target.

Empty openings provide no surface depth at their center. Selecting a visible rim and specifying an origin displacement separates observable support from the desired interaction location. Invalid or degenerate measurements fail geometric construction. For several upright roles on a shared rigid body, planar alignment of their paired origins supplies a common orientation change. This operation does not enforce distances between roles or solve a coupled grasp constraint.

A.6 Illustrative Transfer through a Changed Vessel

Consider a demonstration that grasps a handled vessel, pours into a bowl, and recovers upright. A target replaces the vessel and requests a different recipient. Initial rigid transfer carries the demonstrated motion with the vessel. The VLM then compares that initialization with the target's functional geometry. The grasp role pairs the visible handle parts; the recipient role pairs the receiving openings. Approach and closure follow the grasp role, while the delivery, tilt, and recovery use the recipient role. The model places phase transitions during transit and can edit neighboring entries to connect the two references.

Table 6: **Semantics of source-indexed edits.** Arguments use the local frame of the addressed entry. Insertion and deletion preserve the addresses of every other original source entry.

Operation	Arguments	Effect
Offset	Translation and rotation vector	Modify position and orientation using Equation 7
Replace pose	Position, quaternion, gripper, duration	Replace the addressed local command
Insert	Position, quaternion, gripper, duration	Add a command after the addressed entry, inheriting its role
Delete	Source index	Remove the addressed command
Hold	Minimum duration	Replace the dwell requirement
Gripper	Continuous value in $[-1, 1]$	Replace the gripper command

The new handle determines the local grasp reference. If the vessel's outlet is displaced relative to that grasp, preserving the old recipient-relative wrist motion would displace the pouring location. The model can change the local delivery pose or offset while retaining the demonstrated tilt and recovery where appropriate. A changed handle may similarly require an orientation edit at grasp. These continuous edits are inferred from the paired observations; the interpreter only realizes their stated meaning. Transfer requires compatible grasps and visible functional geometry.

B Task Inventory and Construction Protocol

Table 7 lists the benchmark's 30 tasks: 12 LIBERO-Pro tasks, 14 RoboTwin tasks, and four RoboCasa diagnostic tasks. Eight tasks, four on each primary platform, use six configurations each (two asset variants across three layouts), including one source setting. These 48 entries provide 40 target configurations. Source replay and independent target-feasibility checks determine evaluation eligibility; inventory counts and evaluation denominators are reported separately.

Statistical figure. Figure 3(a) visualizes word frequencies from 63 distinct task queries and 176 task–asset name entries spanning all 30 tasks. Queries and asset names are deduplicated within each task before tokenization; common stopwords and numeric asset identifiers are removed. Figure 3(b) shows one interaction frame from each source demonstration in six columns and five rows: 12 LIBERO-Pro, 14 RoboTwin, and four RoboCasa tasks. Frames are selected for visible manipulation and cropped around the interaction, without trajectory overlays. They illustrate task content rather than method success. LIBERO-Pro has eight tasks with 54 configurations and four with six. RoboTwin has six tasks with 54, two with 57 (microphone handover and object scanning), one with 75 (pot lifting), one with 78 (bowl stacking), and four tasks with six. RoboCasa has 7, 5, 4, and 4 configurations for OpenBlenderLid, TurnSinkSpout, OpenElectricKettleLid, and StartElectricKettle, respectively. The 1,091 inventory entries include control, target, and source configurations. Eight source settings belong to the tasks with six configurations. Evaluation denominators count eligible targets separately.

Figure 3(c) pools positive taxonomy annotations across platforms. Appearance, Layout, Geometry, Substitution, Goal, and Compound have 345, 387, 426, 372, 90, and 402 recorded label assignments, respectively. Each slice divides its count by the total of 2,022 assignments. A multiply tagged configuration contributes once to each recorded category; consequently, the pie describes annotation frequency rather than an exclusive partition of scenes. Missing annotations are not inferred as negative labels. The taxonomy statistics exclude RoboCasa and the eight six-configuration tasks because their annotations are incomplete. Exact per-task allocations and source identities accompany the figure data.

What varies. Figure 2(d) illustrates how one skill is reused across the six settings. Configurations vary appearance, scale, local shape, asset family, recipient, and goal-role assignment, separately or jointly. Variants preserve the contact or articulation structure required for an expert to demonstrate the intended interaction. A new asset family and a new functional category are distinct annotations; neither is inferred from a changed color or model identifier. Three-layout groups are tracked by configuration rather than reconstructed by pooling unrelated scenes. Cross-category conditions are

Table 7: **Task inventory.** Tasks are grouped by their configuration allocation. RoboCasa contributes diagnostic scenes only. Task labels are not a count of distinct motor primitives.

Platform	Group	Tasks
LIBERO-Pro	Group 1	Drawer, lid, pour, wipe, insert, hang, button, knob
RoboTwin	Group 1	Hammer block; press stapler; stamp seal; scan object; open laptop; turn switch; dump bin; hand over microphone; lift pot; stack bowls
RoboCasa	Diagnostic	OpenBlenderLid; TurnSinkSpout; OpenElectricKettleLid; StartElectricKettle
LIBERO-Pro	Group 2	Orient bottle on rack; open–store–close drawer; scoop and transfer; push plate through route
RoboTwin	Group 2	Regrasp and hang mug; orient phone in stand; hold drawer and store; shake bottle in alternating directions

linked to their original controls, so a shared physical scene is not counted again merely because it appears in another comparison.

What is held fixed. One fixed source is reused across a task's targets. Calibration and robot limits are public; simulator object poses, joints, part labels, asset paths, target expert motion, and evaluator state remain private. The target expert establishes feasibility independently of method evaluation.

Physical outcomes. A task's scorer checks the relevant interaction and terminal state. Examples include required mechanism travel, insertion depth, stable support after release, coverage of a contact region, and coordinated grasp ownership. The custom pouring task additionally checks delivery and spillage using an approximate particle-fluid simulator. It requires the demonstrated grasp, lift, alignment, tilt, and upright recovery together with the designated fluid outcome. These quantities are not interchangeable with rigid dumping or a device's activation state.

Environment and method failures. A malformed prediction, invalid grounding, rejected compiled motion, or unsuccessful physical interaction is a method outcome. Simulator initialization failure, corrupted input, and an unavailable model service before a usable response are infrastructure failures. Infrastructure reruns retain the same source and target identity. Method failures are retained rather than rerun until successful. Aggregate scores report exclusions and coverage.

C Benchmark Measures

Let $\mathcal{Q}_k$ be the common eligible target set for task k and $y_{mkq} \in \{0, 1\}$ the outcome of method m on target q. Task-macro success is

$$S_m = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{|\mathcal{Q}_k|} \sum_{q \in \mathcal{Q}_k} y_{mkq}. \tag{9}$$

Episode success instead weights each target equally. Both retain the number of tasks and target episodes, since task-dependent allocation changes their interpretation. Pairwise and multi-method comparisons form their common eligible set before aggregation. Unsupported method conditions are reported with coverage rather than silently removed from a denominator shared by other methods.

For complete three-layout configurations $\mathcal{C}_k$, all-layout robustness is

$$R_m = \frac{1}{|\mathcal{K}_3|} \sum_{k \in \mathcal{K}_3} \frac{1}{|\mathcal{C}_k|} \sum_{c \in \mathcal{C}_k} \prod_{\lambda=1}^{3} y_{mkc\lambda}, \tag{10}$$

where $\mathcal{K}_3$ contains tasks with at least one complete configuration. Incomplete groups cannot establish this measure. For uncertainty estimates, paired task and configuration resampling keeps the layouts and all compared method outcomes together.

Interaction accuracy. Position and orientation errors refer to independently specified task interactions. They retain their units and measurement counts. Stage reach rates accompany conditional

errors so that trials that never establish contact remain visible. Articulation displacement, supported placement, received fluid fraction, and contact coverage are reported within compatible task families rather than averaged as one generic geometric error.

Latency and computational cost. Target wall time starts before target observation preparation and reset, and includes inference, geometry, planning, physical execution, and within-episode retries. Infrastructure reruns are handled separately as described in Appendix B. Full process time additionally includes interpreter and dependency initialization and output persistence; resource queues are recorded separately. The first-action clock uses the same target origin. Source preparation is recorded independently, with both its initial cost and its amortized contribution over reused targets. Cold model loading is separate from warm target inference. Wall time is measured directly rather than by summing parallel durations.

All valid outcomes contribute to aggregate target time. Successful-trial time uses only successful trials and reports its sample count; comparisons at unequal success are not described as execution speedups. We report source-plus-target tokens separately from target-only tokens. Shared source requests are counted once in total usage, and model-dependent unknown usage remains missing. Missing token values never become zero.

Paired perturbations. For paired control and changed conditions, Control and Transfer are the respective episode success rates. Both averages the product of their two binary outcomes over pairs. A control may occur in several pairs, so uncertainty resampling keeps all pairs with that control together. Factor-isolated comparisons must exclude jointly changed conditions; compound conditions retain their full factor tags. The report retains both the number of pairs and the number of unique physical scenes.

D Experimental Interfaces and Controls

Evaluation populations. The benchmark contains 456 LIBERO-Pro, 615 RoboTwin, and 20 RoboCasa configurations across 30 tasks. Tables 1–4 use the evaluation sets specified in Appendix E and the reference protocol below. RoboCasa is an outcome-informed development set and is excluded from the common-target comparisons. The execution analysis uses 6 LIBERO-Pro and 25 RoboTwin conditions with prior development exposure; it characterizes execution behavior rather than held-out transfer performance.

Shared observations and motion. Ours, adapted DITTO, and adapted MimicFunc receive one fixed RGB-D source recording with calibrated cameras, measured TCP poses, gripper state, and timestamps. Target observations and a task query are public; simulator object poses, target expert trajectories, segmentation ground truth, and evaluator state are private. Source preparation is cached once and its cost is retained separately. Native Astra receives the source RGB sequence and fresh target RGB, calibration, robot state, and feedback, without source numerical motion or depth. It is consequently a system reference, not an input-matched removal of α.

DITTO adaptation. The implementation retains the upstream LoFTR matcher, depth lifting, flow-control RANSAC rigid fitting, and Gaussian SE(3) mixing between interaction and destination transforms. Source-only VLM boxes initialize source masks; there is no target-time VLM correction. Measured robot TCP/gripper motion replaces human hand reconstruction. The demonstrated grasp is inherited, rather than regenerated for an arbitrary gripper or tool. The adaptation evaluates DITTO with recorded robot motion in place of human-video reconstruction.

MimicFunc adaptation. Measured robot motion replaces human reconstruction while the functional/grasp correspondence, SD+DINO feature matching, rendered pose selection, and constrained optimization stages are retained. A local public SD+DINO implementation replaces an author-hosted feature service. Optimized motion uses the fixed source clock and the same public physical command interface. Unsupported simultaneous-arm conditions remain outside the common comparison; the adapted dual-arm variant solves per-arm motion on a shared clock rather than a new coupled optimizer.

Decision budget. Ours uses one completed target response. In simulation, Native Astra uses Astra low and may repeatedly observe and issue Cartesian commands within 900 seconds, without a separate call or physical-step cap. The SR columns of Table 1 instead use single-response transfer;

Table 5 uses the 300-second Piper budget. DITTO and MimicFunc retain their required preparation and local optimization costs. Source-stage failures are distinguished from target-stage outcomes. A completed malformed response, infeasible optimization, or unsuccessful interaction is a method failure; technical recovery retains the original input and response history. Source preparation, model loading, resource queues, and active target time are measured separately.

Execution analysis. The 31-condition analysis examines approach-path continuity, inverse-kinematics branch selection, joint position and velocity consistency, and inactive-arm handling. Two terminations occur before interaction, and some command sequences demand abrupt joint changes. These failures identify motion realization as a source of unsuccessful transfer. Source and target identities and model responses are fixed when comparing execution variants.

Model comparison. Table 2 compares GPT-6 Astra low (Ours), GPT-6 Sol low, Fable 5.1, Opus 5, Qwen 3.8 Max, and Gemini 3.8 Flash. Astra in Figure 4(c) denotes the same Ours configuration. Provider-default reasoning applies to the last four. For the shared-response control, Ours and Re-anchor use the same response and grounded frames; Re-anchor removes only motion edits. The executed empty-edit drawer check gives identical 115-step successful commands. This check establishes interpreter consistency for an empty edit set.

Pretrained inputs and outputs. For LIBERO-Pro, the published $\pi_{0.5}$ interface uses front and wrist RGB, the instruction, and an 8-D state (TCP position, axis-angle orientation, and gripper joints). Native 7-D Cartesian action chunks are executed five steps at a time before observing again. Faster-WAM uses its published LIBERO image/state transforms and 7-D actions, executing ten steps per replan. Both receive no source demonstration and use a 900-second target budget on the bottle-transfer conditions. Image orientation, quaternion conversion, gripper conventions, and native action scaling are checked independently of outcomes.

On RoboTwin, Faster-WAM consumes head and two wrist RGB views, language, and 14-D joint/gripper state. Its released policy predicts 32 joint-space steps and executes a 28-step prefix. Zero-WAM additionally conditions its ICL cache on the same fixed expert RGB video. Its 16-D dual-arm predictions comprise pose and gripper commands relative to the initial end-effector frames, converted with the published composition rule. Recorded observations update its cache between chunks. These checkpoint-specific interfaces preserve their trained semantics and are evaluated only on compatible embodiments.

Reference protocol. Table 4 uses 128 RoboTwin-derived target cases selected for native-interface compatibility, because Zero-WAM does not support all benchmark conditions. Appearance, Layout, Geometry, Substitution, Goal, and Compound have 24, 22, 20, 21, 19, and 22 cases. Each case has one reporting category in this slice; Compound contains joint changes. Eligibility is fixed before observing outcomes, and all seven methods share the resulting identities, initial states, and task scorers. The resulting scores characterize transfer on this supported subset. Pretrained policies retain their native cameras, states, and action representations; language tokens do not apply to continuous action outputs. All methods in Table 4 use a common 900-second active target budget. Inference includes all warm model requests and failures; loading, source preparation, geometry, and execution are separate. Local policy timing uses one A100 80GB, batch size one; hosted VLM timing includes request latency.

Piper protocol. Replay comprises ten trials for each of three skills; transfer comprises five trials per skill and taxonomy setting, giving 30 replay and 90 transfer trials. Starts and task criteria are shared across methods, with a 300-second target budget. Native Astra uses Astra low; $\pi_{0.5}$ retains its native policy interface as a pretrained reference. Target time includes failed trials and physical execution, so its ratios differ from the inference-only ratios in the simulation tables.

E Evaluation Populations and Aggregation

E.1 Main-table taxonomy aggregation

Table 1 uses the LIBERO-Pro and RoboTwin common-target identities. Each category uses identical condition identities for all four methods. Appearance, Layout, Geometry, Substitution, Goal, and Compound contain 273, 264, 357, 170, 88, and 301 conditions, respectively. Rates are sums of successes divided by sums of conditions, not averages of platform percentages. Goal has only

RoboTwin coverage and Layout only LIBERO-Pro coverage. Positive tags overlap; LIBERO-Pro Appearance and Compound share the same 243 conditions. Conditions without taxonomy labels and RoboCasa development scenes are excluded. Native Astra uses a single response for the category-success comparison. Its inference and token costs measure multiround control, separately from these success rates.

Cost accounting. Each comparison fixes the target identities and includes both successful and unsuccessful episodes. Ours makes one semantic transfer decision; Native Astra can repeatedly observe and act under the same target budget. DITTO has no target-time VLM requests, so its zero inference and token entries do not imply zero geometric or execution cost.

E.2 Model-slice taxonomy and cost

The model comparison assigns 36 target conditions to each of the six settings, with the same identities and source demonstrations across methods. Macro SR is the unweighted average of the six category rates, distinct from the task-macro measure in Appendix C. Each cell's success count is an integer, and failures remain in the timing averages. Table 3 aggregates the separate complete evaluation pool of 590 unique tagged conditions. Native Astra uses multiround control in this pool. Category memberships overlap, so rates are computed from condition identities rather than by adding category counts; a shared condition has the same outcome in every applicable category. Table 1 uses a separate evaluation population. The model slice compares six VLMs with DITTO, MimicFunc, and multiround Native Astra under common task outcomes and fixed controller interfaces.

E.3 Execution-analysis population

The 31-condition execution analysis uses scene annotations to characterize failures. RoboTwin labels follow full episode identities; LIBERO-Pro layout and shape-replacement labels follow explicit condition fields. Appearance, Layout, Geometry, Substitution, Goal, and Compound have 8, 2, 12, 5, 0, and 1 scored condition assignments. Tags overlap, and conditions without taxonomy labels remain in the overall execution analysis. These assignments describe the coverage of the failure analysis; paired control–transfer degradation is not estimated from category averages.

F Inference and Execution Cost

Table 1 separates category outcomes from reasoning costs. Inference time is 43.8 seconds for Ours and 315.2 seconds for Native Astra; target usage is 6.7k versus 49.3k tokens. Figure 4(a) reports the corresponding ratios, 0.139 and 0.135. Source preparation and resource queues remain separate; geometric execution is not included in the inference clock.

G Limitations

Transfer performance depends jointly on the underlying VLM's spatial reasoning, response reliability, and inference configuration. The model comparison measures their combined effect.

The method uses measured robot motion and calibrated depth, including an initial source depth view. It does not recover actions from arbitrary human video. Source and target must admit compatible robot interactions: a change requiring a different grasp topology, substantial collision avoidance, or a new force strategy can exceed what local editing supplies. Roles are grounded in the initial images, so moving objects and ambiguous depth can invalidate their frames. The single semantic decision cannot revise an incorrect interpretation after observing contact failure. Coupled collision and grasp constraints require additional motion optimization. Finally, the benchmark evaluates transfer within each embodiment; its different robot platforms do not establish cross-embodiment generalization.

H Additional Context on Skill Representations

Motion evidence and spatial invariance. Movement primitives summarize demonstrated motion and support changes in goals (Ijspeert et al., 2013; Paraschos et al., 2013) or intermediate con-

straints (Huang et al., 2019b). Demonstration-derived constraints express which geometric relations matter (Pérez-D'Arpino & Shah, 2017). Dense visual descriptors (Florence et al., 2018) and neural descriptor fields (Simeonov et al., 2021; 2023) connect such relations to new observations; task-specific cross-object pose estimation similarly predicts interaction geometry (Pan et al., 2023). These representations motivate retaining structure in the example rather than treating it only as a task label. Human-to-robot trajectory transfer additionally addresses motion extraction and embodiment constraints (Allu et al., 2025), whereas our source motion is recorded on the robot.

Learned action priors. Spatial action models can exploit translation structure (Zeng et al., 2021; Shridhar et al., 2022) or three-dimensional scene representations (Shridhar et al., 2023; Goyal et al., 2023). Action chunking (Zhao et al., 2023) and diffusion (Chi et al., 2023) model extended behavior, while broader vision-language supervision (Zitkovich et al., 2023) and heterogeneous robot training (Black et al., 2025) expand the prior repertoire. These are complementary to our deployment-time question: how a specific observed action sequence should guide transfer. Our method uses a frozen model and explicit motion evidence, so the contribution of that evidence must be distinguished from pretrained policy knowledge.

Feedback and integrated execution. Language-driven systems use execution history to revise decisions (Huang et al., 2023b), while integration of pretrained perception, navigation, and grasping supports open-vocabulary behavior (Liu et al., 2024). Such capabilities address sustained execution in an environment. The present method instead makes one semantic transfer decision per target and uses low-level tracking feedback thereafter. Its representation could serve a repeated reasoning loop, but that loop is a separate extension.

Figure credits. Figure 1 combines schematic overlays with photographs from PourIt! (Lin et al., 2023) (Apache License 2.0) and MimicFunc (Tang et al., 2025) (CC BY 4.0).

Real-robot materials. Figure 5(a) identifies the robot and asset set used to illustrate physical skill transfer. It uses the manufacturer's white-background Piper product image. Its eleven object illustrations are AI-assisted reconstructions of visible props, not camera observations or quantitative geometric measurements. The nine task frames in Figure 5(b) are extracted from the original recordings without synthesizing interactions. Appearance changes in the cardboard-box example co-occur with geometry and object replacement; these illustrations are not controlled single-factor trials. Source filenames, frame timestamps, and asset provenance accompany the editable figure.