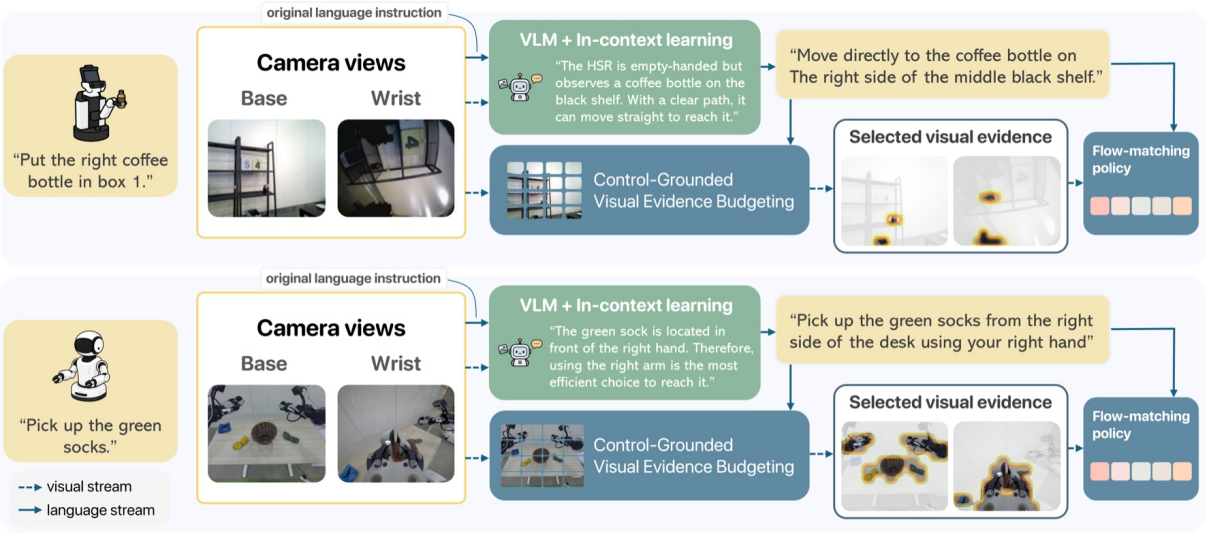
TECHNICAL REPORT

See Less, Specify More: Visual Evidence Budgets for Generalizable VLAs

Yueh-Hua (Kris) Wu* **Tatsuya Matsushima** **Kei Ota**
AIRoA

Abstract

Generalization remains a central bottleneck for vision-language-action (VLA) models: under distractors, appearance shifts, and semantically similar tasks, the policy must often infer local execution details from coarse instructions while also deciding which parts of the image matter for control. We present S2 (**See Less**, **Specify More**), a framework for improving VLA generalization by training the executor under a cleaner interface. **Specify More** preserves the original instruction as a stable high-level goal while relabeling each trajectory into refined trajectory- and subtask-level language that disambiguates the current execution mode. Unlike native attention, **See Less** imposes an explicit visual evidence budget, training the executor to act from task-sufficient evidence rather than unconstrained visual context, without any region or mask annotation. This interface lets the executor follow detailed guidance without relying on distracting visual patches or resolving avoidable ambiguity on its own, and it remains compatible with off-the-shelf VLM planners through in-context learning. Across our main evaluation settings, S2 improves overall generalization metrics by changing the executor's learning problem: coarse instructions induce avoidable supervision aliasing, goal-preserving local guidance outperforms instruction replacement in our main ablations, and explicit evidence budgeting reduces dependence on broad visual context beyond efficiency considerations. Across eight real-robot tasks on TX-G2 (an AgiBot G2-compatible variant) and HSR, S2 raises mean subtask success from 54.2% to 79.0% over $\pi_{0.5}$. Together, these results suggest that VLA generalization improves when the executor is trained to act from informative local guidance and task-sufficient visual evidence, rather than recovering both from weak supervision.



Generalization suffers when the executor must infer both *what to do now* and *what to look at* from a coarse goal alone. S2 resolves this ambiguity from both sides: an off-the-shelf VLM rewrites the same high-level instruction into state-specific local guidance, while a learned control-grounded visual evidence budget shifts the retained evidence from the manipulated object to the destination and the relevant local scene context.

Keywords: Vision-language-action models, Generalization

*Corresponding author: kris.wu@airoa.org.

1. Introduction

Vision-language-action (VLA) models have recently shown strong promise for robot manipulation, but their generalization remains brittle under distractors, appearance shifts, and semantically similar tasks that require different execution strategies (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024). General-purpose VLA control is substantially harder than vision-language understanding alone: the model must not only interpret open-ended goals and scenes, but also choose appropriate local behaviors and output precise low-level continuous actions under distractors, appearance shift, embodiment-specific noise, and tight inference constraints. Expecting a single low-level policy to absorb all of these demands from finite robot data is therefore a poor scaling strategy.

Some of this burden is better handled at high level. Modern vision-language models (VLMs) are stronger at open-ended instruction interpretation and in-context disambiguation than practical low-level executors, so they can provide trajectory-aware guidance about which local mode, route, or arm should be used without forcing the VLA to infer this from a coarse goal alone. Some burden should also be removed from perception. Not every pixel contributes to the current control decision, and training an executor to rely on unconstrained visual context encourages dependence on correlations that may not generalize. This suggests a cleaner design: rather than forcing one model to be both planner and executor, we should explicitly disentangle high-level planning from low-level action generation, and train the executor under a language-and-vision interface that reduces what it must infer on its own (Ahn et al., 2022; Myers et al., 2025; Shi et al., 2025).

In current robot datasets, this burden mismatch appears as avoidable ambiguity and apparent action multimodality. The same instruction may admit multiple valid trajectories: an object can be approached from different sides, grasped at different contact regions, or transported along different collision-free routes. Some of this variation is irreducible, arising from embodiment, contact dynamics, or control noise. But some of it is avoidable. Coarse instructions can alias distinct local behaviors even when a demonstration already commits to one of them, and full images can expose the policy to clutter, distractors, and appearance cues that are irrelevant to the current subtask. Both enlarge the set of modes the low-level policy must explain. We therefore ask whether executor generalization can be improved by shrinking this learning problem from both sides at once: providing more informative local language, and restricting control to a smaller task-sufficient subset of visual evidence.

We present S2 (**See Less**, **Specify More**), a framework built around a more specific claim: in modular planner-executor VLAs, executor generalization depends critically on the conditioning interface under which the executor is trained. Our goal is therefore not merely to add richer language or a learned visual selection mechanism in isolation, but to train the executor under a goal-preserving local language interface together with explicit visual evidence budgeting. **Specify More** realizes the language side of this interface. The original instruction is retained as a stable high-level goal, while refined trajectory-level and subtask-level descriptions make the demonstrated local behavior explicit. This separation preserves task identity while reducing ambiguity about which valid execution mode should be followed. **See Less** realizes the perceptual side. Unlike native attention, which can remain dense and unconstrained, it introduces an explicit visual evidence budget that trains the executor to act from a smaller task-sufficient subset of the image rather than broad visual context. This budget is learned without manual labels, region annotations, or external VLM supervision: the executor discovers from the control objective itself which evidence predicts success under the current subtask. Useful evidence need not correspond to an entire object; it may instead be a contact surface, target region, clearance cue, or local spatial relation. In finite-data robot learning, this bottleneck is introduced not for efficiency but to reduce nuisance dependence and improve generalization (Ryoo et al., 2021; Cheng et al., 2026).

S2 is designed for a modular planner-executor pipeline. At training time, we relabel demonstrations into refined instructions, ordered subtasks, and temporal alignments, then fine-tune the VLA under this executor-conditioning scheme. At deployment, an off-the-shelf VLM provides local subtask guidance through in-context learning while the low-level VLA focuses on execution. Because the planner communicates in the same language interface that the executor is trained to follow, different high-level modules can be swapped without changing the executor's training target. We therefore do not introduce a new planner; instead, we study a train-test-consistent executor setup in which high-level guidance narrows the decision space and explicit evidence budgeting limits what the policy must rely on perceptually. Across our main evaluation settings, the results support the same picture: coarse instructions induce avoidable supervision aliasing, goal-preserving local guidance outperforms instruction replacement in our main ablations, and explicit

evidence budgeting provides complementary robustness gains by reducing dependence on broad visual context. Across eight real-robot tasks on TX-G2 and HSR, S2 raises mean subtask success from 54.2% to 79.0% over $\pi_{0.5}$. Together, these results suggest that better VLA generalization comes from shrinking the executor's learning problem in both language and perception, so it can focus on following the intended behavior rather than recovering it from weak supervision.

Concretely, we recast generalization in modular planner-executor VLAs as an executor-conditioning problem, instantiate this view with goal-preserving hierarchical relabeling and annotation-free control-grounded visual evidence budgeting, and show empirically that this cleaner interface improves generalization while remaining compatible with swappable off-the-shelf VLM planners.

2. Related Work

VLA Generalization and Hierarchical Control. Recent VLA models have substantially expanded language-conditioned robot learning (Jang et al., 2021; Brohan et al., 2023; Ghosh et al., 2024; Kim et al., 2024; Black et al., 2024), including compact or cross-embodiment variants (Shukor et al., 2025; Wang et al., 2025a; Zheng et al., 2025), yet benchmarks such as LIBERO and CALVIN still expose brittle generalization under distribution shift (Liu et al., 2023; Mees et al., 2022). In parallel, language-model-based planners and hierarchical VLA systems decompose high-level decision making from low-level control (Ahn et al., 2022; Liang et al., 2022; Huang et al., 2022; Myers et al., 2025; Shi et al., 2025; Li et al., 2025; Long et al., 2026). These works motivate planner-executor decomposition; our focus is the executor side of that split, and in particular the conditioning interface under which the low-level policy is trained.

Language Interfaces. Several recent works improve low-level controllability through richer command interfaces. STEER uses dense language grounding for flexible manipulation (Smith et al., 2024), Steerable Policies trains VLAs to follow subtasks, motions, traces, and grounded coordinates (Chen et al., 2026a), and ReSteer studies steerability through data refinement (Chen et al., 2026b). Pretrained vision-language models have also been used to enrich robot supervision with more informative language: DIAL and OCI add visual and object-centric detail (Xiao et al., 2023; Wen et al., 2024), while NILS and TREAD scale automatic relabeling to long-horizon trajectories and semantic subtasks (Blank et al., 2024; Kuramshin et al., 2025). Our work is closest to this relabeling line, but targets *specification-induced ambiguity*: the trajectory already commits to one execution mode while the task instruction does not. This motivates a goal-preserving local interface that reduces executor ambiguity without introducing planner-specific control codes. Related approaches instead strengthen monolithic VLA reasoning more directly through reasoning-augmented architectures, latent planner-to-policy interfaces, or RL post-training (Zhao et al., 2025; Lee et al., 2025; Yin et al., 2025; Huang et al., 2025; Chen et al., 2025).

Selective Perception. Another relevant line of work studies how policies should focus on task- relevant perceptual evidence. In robotics, attention-based and object-centric methods improve robustness by emphasizing relevant visual regions (Abolghasemi et al., 2018; Devin et al., 2017; Shridhar et al., 2022a,b), while recent work argues for suppressing irrelevant factors or exploiting locality in visuomotor learning (Zhao et al., 2024; Zhang et al., 2025; Chapin et al., 2026). In parallel, the vision literature and recent VLA work use token selection and token pruning to reduce redundancy or improve efficiency (Ryoo et al., 2021; Rao et al., 2021; Liang et al., 2022; Bolya et al., 2023; Cheng et al., 2026; Li et al., 2026). Our visual evidence budgeting instead introduces an explicit, control-grounded evidence bottleneck: unlike native attention, which can remain dense and unconstrained, it learns from the control objective which visual evidence is sufficient for the current behavior, without manual regions, external visual labels, or efficiency-first objectives.

3. S2: See Less, Specify More

S2 defines the executor-conditioning interface in a modular planner-executor VLA system. **Specify More** refines coarse task language into goal-preserving local supervision, and **See Less** learns annotation-free subtask-conditioned visual evidence restriction directly from the control objective. We consider language-conditioned robot demonstrations of the form $\tau = (\{(o_t, a_t)\}_{t=1}^{T}, g)$, where τ denotes a trajectory of length T paired with an original instruction g. Each observation is $o_t = (I_t^b, I_t^w, x_t)$, where I_t^b and I_t^w are the base and wrist images, respectively, and x_t denotes proprioceptive state.

Given an episode τ, S2 constructs the hierarchical conditioning interface $g \to \tilde{g} \to \mathcal{S}(\tau)$, where g is the original instruction, $\tilde{g}$ is a refined trajectory-level instruction, and $\mathcal{S}(\tau) = \{(s_i, b_{i-1}, b_i)\}_{i=1}^{N}$ denotes the ordered subtask sequence, with subtask instruction s_i and temporal span $[b_{i-1}, b_i)$. The boundaries satisfy $0 = b_0 \le b_1 \le \cdots \le b_N = T$.

During policy learning, the executor is conditioned on the original instruction g, the active subtask instruction s_i, and learned visual evidence masks over the base and wrist views. At deployment, the planner emits local guidance in the same form as the subtask instructions used during training, while the visual masks remain executor-side learned mechanisms rather than planner outputs.

3.1. Specify More: Goal-Preserving Hierarchical Relabeling

For each episode, we first construct the language side of the interface by producing a refined trajectory instruction $\tilde{g}$ that explains how the observed trajectory solves the original task. This relabeling is conditioned on the original instruction g together with representative images from the episode. The goal is not to paraphrase g, but to resolve omitted execution details such as approach direction, contact strategy, ordering, obstacle avoidance, or placement behavior while remaining faithful to the original task identity.

We then decompose the trajectory into an ordered sequence of refined subtasks $s_1, s_2, \ldots, s_N$ using a vision-language relabeling procedure over sparse trajectory context rather than action heuristics alone. Each subtask is intended to be directly executable by the low-level policy and specific enough to distinguish different valid execution modes. We then align each subtask to the trajectory timeline, yielding frame-aligned supervision of the form (s_i, b_{i-1}, b_i) for the executor.

The key design choice in **Specify More** is that we do not replace the original instruction with the relabeled one. Instead, we preserve the original instruction g as the persistent high-level goal and pair it with refined local language. This separation matters because the original instruction contains stable task identity, while the refined local instruction resolves the current execution mode. In other words, the original instruction tells the policy *what overall task is being solved*, and the refined subtask instruction tells it *how the current phase should be executed*. Concrete training-time and evaluation-time prompt-construction procedures for LIBERO, CALVIN, and LIBERO-Pro are given in Appendix B.1.

3.2. See Less: Learned Visual Evidence Budgeting

We next construct the perceptual side of the interface. Even after language ambiguity is reduced, the executor must still determine which parts of the visual observation matter for the current behavior. Unlike native attention, which can remain dense and unconstrained, **See Less** imposes an explicit evidence bottleneck on what the executor may rely on for control. We do not manually specify these regions, nor do we rely on external visual supervision from a VLM. Instead, the executor learns directly from the control objective which evidence is sufficient for successful execution under the current subtask. Useful evidence may include manipulated objects, targets, contact-relevant surfaces, and local spatial relations or clearance cues; importantly, it need not correspond to an entire object. We therefore introduce task-conditioned visual evidence masks $m_t^b \in [0,1]^{P_b}$ and $m_t^w \in [0,1]^{P_w}$ over image patches or visual tokens in the base and wrist views, respectively.

These masks are produced by lightweight learned gate heads inside the executor. For each view $v \in \{b, w\}$ and token p, the gate head takes the current image token $z_{t,p}^v$ together with a pooled summary of the goal-preserving language context and predicts a soft keep value $m_{t,p}^v = \sigma\big(h_v(z_{t,p}^v, q_t)/\tau\big)$, where q_t summarizes the current language context and h_v is a small learned gating network for that view. In our implementation, h_v is a small MLP over the image token, the pooled language summary, and their elementwise product. We then apply a nonzero gate floor via $\hat{m}_{t,p}^v = \gamma + (1-\gamma) m_{t,p}^v$ and $\hat{z}_{t,p}^v = \hat{m}_{t,p}^v z_{t,p}^v$, where $\gamma \in [0,1)$. Because the gate heads are part of the executor itself, masks are produced online during every forward pass from the current observation and language context. Training and inference therefore use the same learned masking mechanism, without external region proposals, mask or box annotations, planner-provided visual labels, or any other explicit supervision over which regions should be kept.

To control how much evidence each stream may softly retain, we define separate visual budgets $\rho_b \in (0,1]$ and $\rho_w \in (0,1]$ for the base and wrist views. These budgets may differ because the two streams need not carry the same information: the wrist view often captures contact-local detail, whereas the base view provides broader spatial context. Their role is not primarily efficiency, but robustness: the policy is encouraged to solve the current subtask while depending on less nuisance visual evidence.

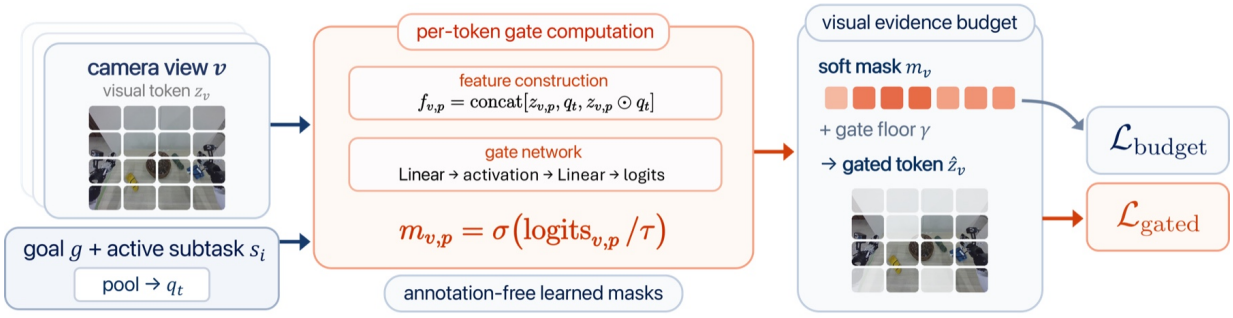


Figure 1 | Control-grounded visual evidence budgeting. For each camera view, the executor predicts a soft mask from visual tokens and goal-preserving language context, then trains the gated representation with a task loss and a budget regularizer, without any region, box, or mask annotation.

To encourage the learned masks to satisfy these targets, we penalize the deviation between the average pre-floor gate value and the desired soft budget in each view. Let $\bar{m}_t^v = \frac{1}{P_v}\sum_{p=1}^{P_v} m_{t,p}^v$ denote the mean pre-floor gate value in view $v \in \{b, w\}$. We then optimize

$$\mathcal{L}_{\text{budget}} = \sum_{v\in\{b,w\}} \lambda_{\text{budget}}^v \mathbb{E}_{\tau,t}\left[(\bar{m}_t^v - \rho_v)^2\right]. \tag{1}$$

This term constrains how much evidence is softly retained on average while still allowing the model to decide *which* tokens or patches should be more strongly retained or suppressed. The nontrivial selection pressure then comes from maintaining task sufficiency under the action losses below.

3.3. Policy Learning

The executor-conditioning claim of S2 is instantiated through two coupled paths: an ungated path that preserves the full observation, and a gated path that applies the learned visual evidence restriction to the image-token representation under the same language interface. Let $c_t^{\text{lang}} = (g, s_i)$ denote the language-side conditioning context, and let $\mathcal{L}_{\text{base}}(\theta; o_t, c_t, a_{t:t+h-1})$ denote a generic language-conditioned visuomotor training objective, where o_t is the current observation defined above, c_t is the conditioning context presented to the policy, $a_{t:t+h-1}$ is the target action chunk, and h is the action horizon. Here g preserves global task identity and s_i specifies the current local behavior. The language portion of the conditioning has the same form as the local guidance emitted by the planner at deployment.

In the current implementation, we retain the ungated base path

$$\mathcal{L}_{\text{full}} = \mathcal{L}_{\text{base}}(\theta; o_t, c_t^{\text{lang}}, a_{t:t+h-1}), \tag{2}$$

and add a second task loss on the gated token representation

$$\mathcal{L}_{\text{gated}} = \mathcal{L}_{\text{base}}(\theta; \tilde{o}_t(m_t^b, m_t^w), c_t^{\text{lang}}, a_{t:t+h-1}), \tag{3}$$

where $\tilde{o}_t(m_t^b, m_t^w)$ denotes the same observation after the learned token gates are applied to the visual streams. The purpose of $\mathcal{L}_{\text{gated}}$ is to ensure that the gated representation remains sufficient for action prediction. Together with $\mathcal{L}_{\text{budget}}$, it determines *which* evidence can be discarded: $\mathcal{L}_{\text{budget}}$ constrains how much evidence is retained, while $\mathcal{L}_{\text{gated}}$ forces the retained subset to remain control-relevant.

The resulting S2 objective combines the ungated path, the gated path, and the budget regularizer defined above:

$$\mathcal{L}_{\text{S2}} = \mathcal{L}_{\text{full}} + \mathcal{L}_{\text{gated}} + \mathcal{L}_{\text{budget}}. \tag{4}$$

Early gate collapse or trivial all-keep behavior is avoided by three implementation choices: a nonzero gate floor, the ungated loss $\mathcal{L}_{\text{full}}$, and an annealed soft-gating schedule that keeps the gate smoother early in

Table 1 | Full LIBERO-PRO perturbation breakdown across the evaluated suites. The two S2 rows are lightly highlighted.

Methods	libero_10				libero_goal				libero_object				libero_spatial			
	Obj.	Pos.	Sem.	Task	Obj.	Pos.	Sem.	Task	Obj.	Pos.	Sem.	Task	Obj.	Pos.	Sem.	Task
OpenVLA-OFT (Kim et al., 2025)	5.5	0.0	38.5	0.0	8.5	0.0	43.5	2.5	66.5	9.5	89.0	0.0	30.0	13.5	65.0	0.0
X-VLA (Zheng et al., 2025)	61.0	1.0	70.5	19.5	71.5	1.0	94.0	7.5	91.5	4.5	98.0	0.0	89.5	0.0	69.0	38.5
VLA-Adapter (Wang et al., 2025b)	47.0	0.0	91.0	10.0	61.0	0.0	75.0	12.0	89.0	0.0	99.0	8.0	98.0	0.0	**98.0**	49.0
$\pi_{0.5}$ (Black et al., 2025)	66.0	6.0	91.0	17.0	90.0	29.0	95.0	17.0	89.0	19.0	95.0	10.0	**99.0**	53.0	97.0	55.0
S2 (GPT-5.4 nano)	**70.0**	17.0	94.0	30.5	**92.0**	34.5	94.5	**44.0**	**95.5**	**63.5**	**100.0**	24.0	**99.0**	53.5	95.0	**57.0**
S2 (Kimi K2.5)	68.0	**18.5**	**94.5**	**36.0**	87.0	**49.5**	**97.0**	43.0	90.0	62.5	**100.0**	**29.5**	**99.0**	**55.0**	90.5	54.5

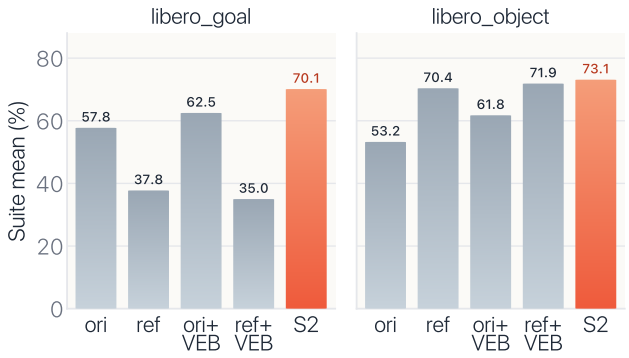


Figure 2 | Mean-only object/goal ablation.

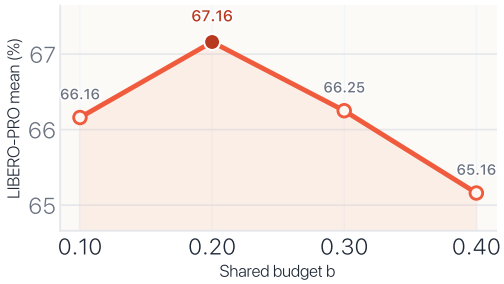


Figure 3 | LIBERO-PRO mean vs. shared visual evidence budget b.

training before sharpening later. The full path stabilizes optimization, while the coupled $\mathcal{L}_{\text{gated}} + \mathcal{L}_{\text{budget}}$ objective learns an annotation-free control-grounded bottleneck rather than a generic saliency map. Exact backbone-specific instantiations of $\mathcal{L}_{\text{base}}$, together with the concrete schedules and loss weights used in our experiments, are given in Appendix A.7 and Table 3.

At deployment, the same interface is reused. A user may provide a coarse task instruction corresponding to g, while an off-the-shelf VLM produces refined local guidance in the same form as the subtask instructions used during training, via in-context learning rather than task-specific fine-tuning. The low-level VLA then executes the current detailed instruction under the learned visual evidence budget. This preserves the intended split of responsibilities: the VLM resolves which local behavior should be executed, while the low-level policy focuses on robust execution under reduced visual distraction. Because the planner communicates through the same guidance form that the executor is trained to follow, different planners can be swapped in without changing the executor's training target.

4. Experiments

We evaluate the central executor-conditioning claim of S2. The experiments ask whether coarse task instructions create conditioning ambiguity, whether the benefit of refined local language comes from goal-preserving disambiguation rather than instruction replacement alone, and whether learned visual evidence restriction provides robustness gains beyond a language-only interface.

4.1. Experimental Setup

Unless otherwise specified, all experiments instantiate S2 with a $\pi_{0.5}$-based executor (Physical Intelligence et al., 2025) and fine-tune from the released `pi05_base` checkpoint. Unless noted otherwise, we use the goal-preserving hybrid prompt together with shared VEB at $\rho_b = \rho_w = 0.2$. Unless otherwise noted, our default relabeling and evaluation-time prompting/planning configuration uses Kimi K2.5. Appendix B.1 details the benchmark-specific prompt refinement, relabeling, and evaluation-time prompting procedures, and Appendix B.2 summarizes the real-robot dataset composition.

4.2. LIBERO-PRO Benchmark

We first compare the full S2 interface against prior methods across all LIBERO-PRO perturbation categories.

The benchmark comparison also highlights that S2 is not tied to a planner-specific VLM: both S2 rows

Table 2 | Real-robot mean subtask success on TX-G2 and HSR, with standard deviation in gray. Full per-task Subtask/E2E tables are deferred to Appendix B.7.

Method	TX-G2				HSR			
	Cutlery	Bowl	Clothes	Dish	Coffee	Bottles	Box	Mug
X-VLA	0.0 ±0.0	7.5 ±4.0	5.0 ±2.6	5.0 ±3.4	8.3 ±7.6	4.2 ±3.8	0.0 ±0.0	25.0 ±12.3
VLA-Adapter	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
$\pi_{0.5}$	5.0 ±3.2	47.5 ±6.8	83.3 ±4.8	60.0 ±7.2	83.3 ±9.6	45.8 ±8.5	41.7 ±14.0	66.7 ±9.6
S2	**15.0** ±5.4	**67.5** ±6.8	**96.7** ±2.2	**90.0** ±4.6	**100.0** ±0.0	**87.5** ±6.6	**91.7** ±7.6	**83.3** ±9.6

use the same trained low-level VLA and differ only in the off-the-shelf planner used to issue refined local guidance: an open-source Kimi K2.5 planner (Kimi Team, 2026) and a proprietary GPT-5.4 nano planner (OpenAI, 2026). Across both choices, the two S2 planner instantiations remain among the strongest on the perturbation axes that most directly stress executor generalization, with the clearest gains on `libero_goal` and `libero_object`. The planner choice changes which axes peak, but the overall pattern is consistent: the cleaner executor interface helps the policy identify the intended local behavior while reducing dependence on task-irrelevant visual factors.

4.3. LIBERO-PRO Object/Goal Ablation

The clearest test of the language-side claim is the object/goal ablation on `libero_goal` and `libero_object`, which most directly expose the difference between preserving task identity and merely replacing the original instruction with richer local text. Figure 2 summarizes five representative settings, and Appendix Table 9 gives the full perturbation breakdown, including the goal-preserving hybrid without learned VEB.

The result is not simply that more language helps. Refined-only supervision remains weak on `libero_goal`, so local text alone does not solve the task-identity problem; conversely, recovering task identity without learned VEB still underperforms the full interface. Only the full S2 interface remains strong on both suites, supporting the claim that goal-preserving local guidance and learned visual evidence restriction are both needed.

4.4. Budget Sensitivity

Figure 3 shows that the best LIBERO-PRO mean occurs at $b = 0.20$, with nearby budgets remaining competitive. We therefore use $\rho = 0.2$ as the main-paper default.

4.5. Real-Robot Results

We additionally report real-robot subtask success on TX-G2 and HSR. The main text reports mean *Subtask* success, while the full per-task *Subtask/E2E* breakdowns are deferred to Appendix B.7. For fairness, every compared method is fine-tuned for 150k steps on the corresponding robot dataset and evaluated on the same RTX 5070 setup; under this constraint, OpenVLA-OFT exceeds available memory and is therefore omitted from the real-robot comparison.

Figure 4 | TX-G2 and HSR.

On TX-G2, each task is evaluated ten times with shared initial states and object placements across methods (8 near-ID and 2 OOD placements). Because TX-G2 is bimanual, the policy must also infer which arm should execute the current behavior. HSR uses six trials per task, adds locomotion, and is queried at 10 Hz on TX-G2 and 2 Hz on HSR. Appendix B.8 provides the detailed task definitions, and Appendix B.10 reports TX-G2 cluttered-scene stress tests with unseen distractors and online object/basket perturbations.

Across both TX-G2 and HSR, S2 improves over $\pi_{0.5}$ on all reported tasks, with especially large gains on HSR tasks that require combining locomotion with precise manipulation. VLA-Adapter remains at 0.0 mean subtask success on both robots despite receiving the same 150k fine-tuning budget; Appendix B.11 discusses diagnostic evidence and likely causes. These results are consistent with the intended role of S2: a clearer executor interface improves local subtask execution even when the robot must combine fine manipulation

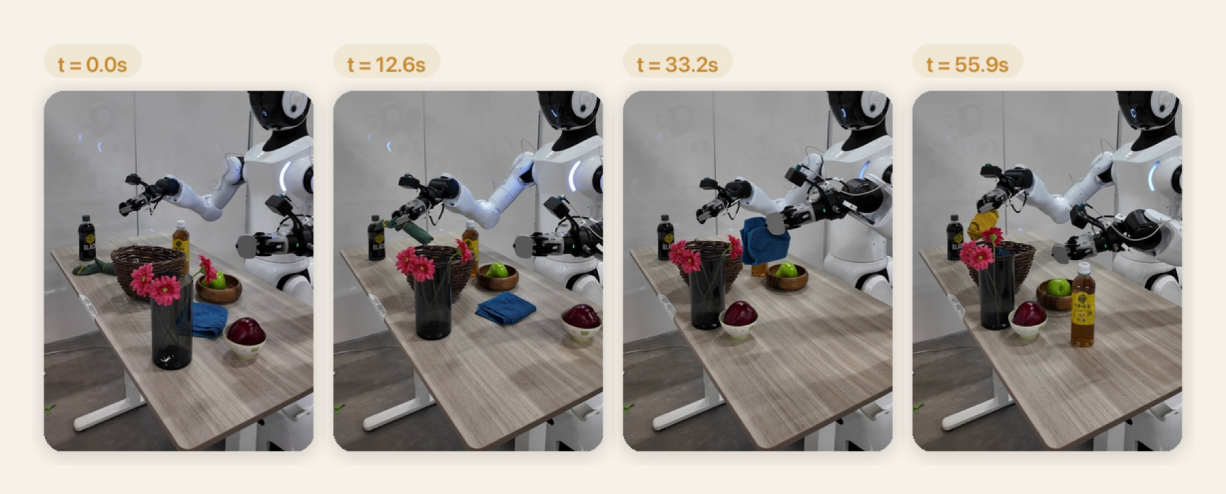


Figure 5 | A successful TX-G2 cluttered-scene rollout for the ordered task `green socks` → `handkerchief` → `yellow socks`. Most objects on the table (flowers, fruit, bottles) are unseen distractors, and a person perturbs object and basket positions during execution, yet S2 re-grounds the current target online and preserves the required completion order. Further examples are given in Appendix B.10.

Figure 6 | Qualitative comparison between the learned visual evidence budget and the backbone's native attention on representative real-robot observations. VEB concentrates on behavior-relevant objects, contact regions, and local context, while the backbone attention is more diffuse. The learned masks require no region or mask annotation.

with arm selection, locomotion, or both.

Robustness under unseen clutter. Beyond the standard placements, we stress-test S2 on a deliberately cluttered TX-G2 clothes-sorting setup in which most tabletop objects are unseen distractors and a person perturbs the scene online during execution. Figure 5 shows one such rollout: S2 keeps acting on the instructed garment and basket while ignoring the surrounding clutter, and completes the ordered sequence despite these perturbations. Additional cluttered rollouts and their learned masks are reported in Appendix B.10.

Qualitative evidence selection. Figure 6 compares the learned visual evidence budget with the backbone's native attention on representative real-robot observations. VEB is more concentrated on the behavior-relevant object, end-effector, and destination context, while the backbone attention remains more diffuse and often allocates mass to broad or weakly task-related regions. The masks are learned without any region or mask annotation; Appendix B.9 provides additional TX-G2 and HSR examples across all real-robot task families.

5. Conclusion

We presented S2, a framework for improving VLA generalization by training the executor under a cleaner conditioning interface. Across our evaluated settings, goal-preserving local guidance outperforms instruction replacement, and learned visual evidence restriction provides complementary robustness gains. More broadly, the results suggest that generalizable robot control benefits from giving the executor a clearer and more task-faithful interface, rather than forcing it to infer both the intended behavior and the relevant evidence from weak supervision alone.

Acknowledgments

This paper is based on results obtained from a project, JPNP25015, commissioned by the New Energy and Industrial Technology Development Organization (NEDO). We thank Ryosuke Takanami and Haru Kondoh for their assistance in the real-robot experiments.

References

A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, et al. Rt-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.025. URL https://arxiv.org/abs/2212.06817.

M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, et al. Openvla: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

M. Ahn, A. Brohan, N. Brown, et al. Do as i can, not as i say: Grounding language in robotic affordances, 2022. URL https://arxiv.org/abs/2204.01691.

V. Myers, C. Zheng, O. Mees, K. Fang, and S. Levine. Policy adaptation via language optimization: Decomposing tasks for few-shot imitation. In *Proceedings of the 8th Conference on Robot Learning*, pages 1402–1426, 2025. URL https://proceedings.mlr.press/v270/myers25a.html.

L. X. Shi, B. Ichter, M. Equi, L. Ke, K. Pertsch, Q. Vuong, J. Tanner, A. Walling, H. Wang, N. Fusai, A. Li-Bell, D. Driess, L. Groom, S. Levine, and C. Finn. Hi robot: Open-ended instruction following with hierarchical vision-language-action models, 2025. URL https://arxiv.org/abs/2502.19417.

M. S. Ryoo, A. J. Piergiovanni, M. Tan, and A. Angelova. Tokenlearner: What can 8 learned tokens do for images and videos?, 2021. URL https://arxiv.org/abs/2106.11297.

J. Cheng, H. Wang, W. Li, G. Wang, Y. Zhang, X. Tang, J. Wu, X. Chen, Y. Liu, and W. Zhang. Vla-iap: Training-free visual token pruning via interaction alignment for vision-language-action models, 2026. URL https://arxiv.org/abs/2603.22991.

E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Proceedings of the 5th Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=8kbp23tSGYv.

D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, S. Levine, et al. Octo: An open-source generalist robot policy, 2024. URL https://arxiv.org/abs/2405.12213.

M. Shukor, D. Aubakirova, F. Capuano, P. Kooijmans, S. Palma, A. Zouitine, M. Aractingi, C. Pascal, M. Russi, A. Marafioti, S. Alibert, M. Cord, T. Wolf, and R. Cadene. Smolvla: A vision-language-action model for affordable and efficient robotics, 2025. URL https://arxiv.org/abs/2506.01844.

H. Wang, C. Xiong, R. Wang, and X. Chen. Bitvla: 1-bit vision-language-action models for robotics manipulation, 2025a. URL https://arxiv.org/abs/2506.07530.

J. Zheng, J. Li, Z. Wang, D. Liu, X. Kang, Y. Feng, Y. Zheng, J. Zou, Y. Chen, J. Zeng, Y.-Q. Zhang, J. Pang, J. Liu, T. Wang, and X. Zhan. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model, 2025. URL https://arxiv.org/abs/2510.10274.

B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL https://arxiv.org/abs/2306.03310.

O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3): 7327–7334, 2022. doi: 10.1109/LRA.2022.3180108. URL https://arxiv.org/abs/2112.03227.

J. Liang, W. Huang, F. Xia, et al. Code as policies: Language model programs for embodied control, 2022. URL https://arxiv.org/abs/2209.07753.

W. Huang, F. Xia, T. Xiao, et al. Inner monologue: Embodied reasoning through planning with language models, 2022. URL https://arxiv.org/abs/2207.05608.

Y. Li, Y. Deng, J. Zhang, J. Jang, M. Memmel, R. Yu, C. R. Garrett, F. Ramos, D. Fox, A. Li, A. Gupta, and A. Goyal. Hamster: Hierarchical action models for open-world robot manipulation, 2025. URL https://arxiv.org/abs/2502.05485.

Q. Long, Y. Wang, J. Song, J. Zhang, P. Li, W. Wang, Y. Wang, H. Li, S. Xie, G. Yao, H. Zhang, X. Wang, Z. Wang, X. Lan, H. Liu, and X. Li. Scaling world model for hierarchical manipulation policies, 2026. URL https://arxiv.org/abs/2602.10983.

L. Smith, A. Irpan, M. G. Arenas, S. Kirmani, D. Kalashnikov, D. Shah, and T. Xiao. Steer: Flexible robotic manipulation via dense language grounding, 2024. URL https://arxiv.org/abs/2411.03409.

W. Chen, J. S. Bhatia, C. Glossop, N. Mathihalli, R. Doshi, A. Tang, D. Driess, K. Pertsch, and S. Levine. Steerable vision-language-action policies for embodied reasoning and hierarchical control, 2026a. URL https://arxiv.org/abs/2602.13193.

Z. Chen, A. Tian, L. Wang, B. Joffe, Y. C. Lin, Y. Chen, S. Karamcheti, and D. Xu. Resteer: Quantifying and refining the steerability of multitask robot policies, 2026b. URL https://arxiv.org/abs/2603.17300.

T. Xiao, H. Chan, P. Sermanet, A. Wahid, A. Brohan, K. Hausman, S. Levine, and J. Tompson. Robotic skill acquisition via instruction augmentation with vision-language models. In *Robotics: Science and Systems*, 2023. URL https://rss2023.github.io/rss2023-website/program/papers/029/. Introduces Data-driven Instruction Augmentation for Language-conditioned control (DIAL).

J. Wen, Y. Zhu, M. Zhu, J. Li, Z. Xu, Z. Che, C. Shen, Y. Peng, D. Liu, F. Feng, and J. Tang. Object-centric instruction augmentation for robotic manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2024. URL https://oci-robotics.github.io/.

N. Blank, M. Reuss, M. Rühle, Ö. E. Yağmurlu, F. Wenzel, O. Mees, and R. Lioutikov. Scaling robot policy learning via zero-shot labeling with foundation models. In *Proceedings of the 8th Conference on Robot Learning*, 2024. URL https://proceedings.mlr.press/v270/blank25a.html.

A. Kuramshin, O. Aslan, C. Neary, and G. Berseth. Task robustness via re-labelling vision-action robot data. In *CoRL 2025 Robot Data Workshop*, 2025. URL https://openreview.net/forum?id=M6M5W0lmaY.

Q. Zhao, Y. Lu, M. J. Kim, Z. Fu, Z. Zhang, Y. Wu, Z. Li, Q. Ma, S. Han, C. Finn, A. Handa, M.-Y. Liu, D. Xiang, G. Wetzstein, and T.-Y. Lin. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models, 2025. URL https://arxiv.org/abs/2503.22020.

J. Lee, J. Duan, H. Fang, Y. Deng, S. Liu, B. Li, B. Fang, J. Zhang, Y. R. Wang, S. Lee, W. Han, W. Pumacay, A. Wu, R. Hendrix, K. Farley, E. VanderBilt, A. Farhadi, D. Fox, and R. Krishna. Molmoact: Action reasoning models that can reason in space, 2025. URL https://arxiv.org/abs/2508.07917.

C. Yin, Y. Lin, W. Xu, S. Tam, X. Zeng, Z. Liu, and Z. Yin. Deepthinkvla: Enhancing reasoning capability of vision-language-action models, 2025. URL https://arxiv.org/abs/2511.15669.

C.-P. Huang, Y.-H. Wu, M.-H. Chen, Y.-C. F. Wang, and F.-E. Yang. Thinkact: Vision-language-action reasoning via reinforced visual latent planning, 2025. URL https://jasper0314-huang.github.io/thinkact-vla/.

Z. Chen, R. Niu, H. Kong, Q. Wang, Q. Xing, and Z. Fan. Tgrpo: Fine-tuning vision-language-action model via trajectory-wise group relative policy optimization, 2025. URL https://arxiv.org/abs/2506.08440.

P. Abolghasemi, A. Mazaheri, M. Shah, and L. Bölöni. Pay attention! - robustifying a deep visuomotor policy through task-focused attention, 2018. URL https://arxiv.org/abs/1809.10093.

C. Devin, P. Abbeel, T. Darrell, and S. Levine. Deep object-centric representations for generalizable robot learning, 2017. URL https://arxiv.org/abs/1708.04225.

M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Proceedings of the Conference on Robot Learning*, 2022a. URL https://arxiv.org/abs/2109.12098.

M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation, 2022b. URL https://arxiv.org/abs/2209.05451.

Y. Zhao, K. Wu, T. Yi, Z. Xu, X. Ju, Z. Che, C. H. Liu, and J. Tang. Efficient training of generalizable visuomotor policies via control-aware augmentation, 2024. URL https://arxiv.org/abs/2401.09258. EAGLE.

T. Zhang, Y. Hu, J. You, and Y. Gao. Leveraging locality to boost sample efficiency in robotic manipulation. In *Proceedings of the 8th Conference on Robot Learning*, pages 3264–3284, 2025. URL https://proceedings.mlr.press/v270/zhang25h.html.

A. Chapin, B. Machado, E. Dellandréa, and L. Chen. Spotlighting task-relevant features: Object-centric representations for better generalization in robotic manipulation, 2026. URL https://arxiv.org/abs/2601.21416.

Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, and C.-J. Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems*, 2021. URL https://arxiv.org/abs/2106.02034.

Y. Liang, G. Zhang, Z. Zhang, Y. Hu, B. Chandramouli, et al. Evit: Expediting vision transformers via token reorganizations, 2022. URL https://arxiv.org/abs/2202.07800.

D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, C. Feichtenhofer, and J. Hoffman. Token merging: Your vit but faster. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.09461.

H. Li, W. Mao, Z. Lan, H. Xiong, H. Wang, C. Si, Z. Liu, X. Deng, and H. Chen. Bfa++: Hierarchical best-feature-aware token prune for multi-view vision language action model, 2026. URL https://arxiv.org/abs/2602.20566.

Physical Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025. URL https://arxiv.org/abs/2502.19645.

Y. Wang, P. Ding, L. Li, C. Cui, Z. Ge, X. Tong, W. Song, H. Zhao, W. Zhao, P. Hou, S. Huang, Y. Tang, W. Wang, R. Zhang, J. Liu, and D. Wang. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model, 2025b. URL https://arxiv.org/abs/2509.09372.

Kimi Team. Kimi k2.5: Visual agentic intelligence, 2026. URL https://arxiv.org/abs/2602.02276.

OpenAI. Introducing gpt-5.4 mini and nano, Mar. 2026. URL https://openai.com/index/introducing-gpt-5-4-mini-and-nano/.

Y. Wang, X. Li, W. Wang, J. Zhang, Y. Li, Y. Chen, X. Wang, and Z. Zhang. Unified vision-language-action model, 2025c. URL https://arxiv.org/abs/2506.19850.

A. Additional Method Details

A.1. Backbone-Agnostic Formulation and $\pi_{0.5}$ Instantiation

The S2 formulation is largely orthogonal to the choice of underlying vision-language-action (VLA) backbone. At a high level, 'Specify More' modifies the language interface presented to the policy, and 'See Less' modifies how visual evidence is weighted or suppressed. Neither idea requires a specific action parameterization. In principle, the same formulation could be combined with other language-conditioned visuomotor policies, provided they support conditioning on refined language inputs and some mechanism for task-conditioned visual gating or masking.

In this work, however, we instantiate S2 using the $\pi_{0.5}$ backbone from OpenPI (Physical Intelligence et al., 2025) and fine-tune from the released 'pi05_base' checkpoint. In our current setup, the corresponding LIBERO training configuration is 'pi05_libero', which uses action horizon 10 and initializes from 'pi05_base'. We use this instantiation to evaluate the effect of S2, but do not claim that the method itself is specific to $\pi_{0.5}$.

A.2. Base Flow-Matching Objective

In the $\pi_{0.5}$ instantiation, the policy predicts an action chunk using a flow-matching objective. Let a denote the target action chunk, let $\epsilon \sim \mathcal{N}(0, I)$ denote Gaussian noise, and let $t \sim p(t)$ denote the flow-matching time variable. The noised action state is

$$x_t = t\epsilon + (1-t)a, \tag{5}$$

and the target flow is

$$u_t = \epsilon - a. \tag{6}$$

Given an observation o_t and a language input ℓ_t, the backbone predicts a velocity field $v_\theta(x_t, o_t, \ell_t)$ and is trained with the standard mean-squared flow-matching loss

$$\mathcal{L}_{\text{base}} = \mathbb{E}_{(\tau,t,\epsilon)} \left[\left\| v_\theta(x_t, o_t, \ell_t) - u_t \right\|_2^2 \right]. \tag{7}$$

A.3. S2-Conditioned Objective

Under S2, the generic language input ℓ_t is replaced with goal-preserving hierarchical language context: the original instruction g provides stable task identity, and the active refined subtask instruction s_i specifies the current execution mode. In addition, 'See Less' modifies the visual context through task-conditioned visual evidence masks m_t^b, m_t^w. At the level of the backbone objective, this yields

$$\mathcal{L}_{\text{S2-cond}} = \mathbb{E}_{(\tau,t,\epsilon)} \left[\left\| v_\theta(x_t, o_t, g, s_i, m_t^b, m_t^w) - u_t \right\|_2^2 \right]. \tag{8}$$

This formulation emphasizes that S2 changes the information made available to the policy, while remaining compatible with the underlying backbone objective.

A.4. Visual Budget Gate Architecture

The learned visual budget gate operates directly on image-token embeddings before they are concatenated with prompt tokens into the backbone prefix. For each image token $z_{t,p}^v$, the gate head receives three feature blocks:

$$\phi_{t,p}^v = \left[z_{t,p}^v,\ q_t,\ z_{t,p}^v \odot q_t \right], \tag{9}$$

where q_t is a pooled prompt summary and $\odot$ denotes elementwise product. In the current implementation, each view uses a lightweight two-layer MLP

$$\ell_{t,p}^v = W_2\, \text{swish}(W_1 \phi_{t,p}^v), \tag{10}$$

followed by temperature-scaled sigmoid gating

$$m^v_{t,p} = \sigma(\ell^v_{t,p}/\tau). \tag{11}$$

The effective multiplicative gate applied to the token embedding is

$$\hat{m}^v_{t,p} = \gamma + (1-\gamma)m^v_{t,p}, \qquad \hat{z}^v_{t,p} = \hat{m}^v_{t,p} z^v_{t,p}, \tag{12}$$

where γ is the configured gate floor. The floor ensures that every token retains a small residual contribution even when its learned gate is near zero.

A.5. Visual Evidence Budget Objective

In our $\pi_{0.5}$ implementation, visual evidence budgeting is optimized with both an ungated prediction path and a gated path. The total training objective takes the form

$$\mathcal{L}_{\text{S2}} = \mathcal{L}_{\text{full}} + \lambda_{\text{gated}}\mathcal{L}_{\text{gated}} + \lambda^b_{\text{budget}}\mathcal{L}^b_{\text{budget}} + \lambda^w_{\text{budget}}\mathcal{L}^w_{\text{budget}}, \tag{13}$$

where $\mathcal{L}_{\text{full}}$ is the base flow-matching loss computed on the ungated representation, $\mathcal{L}_{\text{gated}}$ is the corresponding loss on the gated representation, $\mathcal{L}^b_{\text{budget}}$ and $\mathcal{L}^w_{\text{budget}}$ penalize deviations from the target base and wrist soft keep ratios. Concretely, if

$$\bar{m}^b_t = \frac{1}{P_b}\sum_{p=1}^{P_b} m^b_{t,p}, \qquad \bar{m}^w_t = \frac{1}{P_w}\sum_{p=1}^{P_w} m^w_{t,p}, \tag{14}$$

then the budget penalties take the form

$$\mathcal{L}^b_{\text{budget}} = \mathbb{E}_{\tau,t}\left[\left(\bar{m}^b_t - \rho_b\right)^2\right], \qquad \mathcal{L}^w_{\text{budget}} = \mathbb{E}_{\tau,t}\left[\left(\bar{m}^w_t - \rho_w\right)^2\right]. \tag{15}$$

This matches the implementation choice of penalizing the squared error between the per-sample mean pre-floor gate value and the desired soft keep ratio in each view.

This objective is specific to our $\pi_{0.5}$ implementation of 'See Less'. The broader S2 idea does not depend on this exact loss decomposition; what is essential is that the policy is trained under refined hierarchical language conditioning and task-conditioned suppression of irrelevant visual evidence.

A.6. Gated Loss

Beyond the base task loss and the budget penalties, our implementation also uses a gated prediction loss. The gated loss has the same form as the base task loss, but is computed after applying the learned visual evidence masks to the image-token representation:

$$\mathcal{L}_{\text{gated}} = \mathbb{E}_{(\tau,t,\epsilon)}\left[\left\|v_\theta(x_t, o_t, g, s_i, m^b_t, m^w_t) - u_t\right\|^2_2\right]. \tag{16}$$

In other words, the model is trained both on the original ungated visual representation and on the gated representation. This is important because the budget loss alone only constrains the average pre-floor gate value; without $\mathcal{L}_{\text{gated}}$, many degenerate keep patterns would satisfy the budget equally well. The gated task loss breaks this degeneracy by penalizing any gate pattern that removes evidence needed for action prediction.

A.7. Schedules and Early-Training Stabilization

Several scheduled quantities are used to keep the gate stable early in training. The gate sigmoid temperature (denoted here as τ_{gate} to distinguish it from the trajectory variable τ) is annealed linearly from $\tau_{\text{start}} = 1.0$ to $\tau_{\text{end}} = 0.7$:

$$\tau_{\text{gate}}(k) = \tau_{\text{start}} + (\tau_{\text{end}} - \tau_{\text{start}}) \frac{k}{K-1}, \tag{17}$$

where k is the current optimization step and K is the total number of training steps. Equivalently, early training uses a higher temperature, so the gate remains smoother and less saturated, while later training uses a lower temperature, producing sharper keep/suppress decisions that better match the target soft evidence budget. At inference time, we use the sharpened endpoint value τ_{end}. Combined with the ungated base path and the nonzero gate floor, this schedule is the main mechanism used to avoid immediate gate collapse while still converging to selective evidence restriction.

A.8. Per-View Visual Budgets

Our formulation permits separate visual evidence budgets for the base and wrist views,

$$\rho_b \in (0, 1], \qquad \rho_w \in (0, 1]. \tag{18}$$

This is important because the two views need not contain the same type or amount of useful information. In particular, the wrist view often contains more contact-local detail, while the base view provides broader scene context and may contain different forms of clutter or distractors.

In the main experiments, however, we set $\rho_b = \rho_w = 0.2$ in order to reduce the experimental search space and isolate the effect of visual evidence budgeting itself without introducing an additional per-view hyperparameter sweep. This shared-budget setting should be understood as an experimental simplification rather than a limitation of the method.

A.9. Key Hyperparameters

Table 3 summarizes the most important gate-related and training-level hyperparameters used in our experiments. The gate-specific values are shared across our strongest TX-G2 and HSR runs, while LIBERO differs mainly in the soft keep-target sweep and some training-level schedule choices.

Table 3 | Key S2 hyperparameter values used in the main experiment families.

Hyperparameter	Typical value(s)
Gate-head hidden dim	256
Gate floor γ	0.1
Soft keep targets ρ_b, ρ_w	0.2
λ_{gated}	1.0
$\lambda^b_{\text{budget}}$, $\lambda^w_{\text{budget}}$	0.1 / 0.1
τ_{start}, τ_{end}	$1.0 \rightarrow 0.7$
Batch size	LIBERO: 128 TX-G2/HSR: 64

A.10. Subtask Alignment Noise and Planner Sensitivity

The current training path uses hard frame-aligned subtask spans. As a result, subtask-boundary errors appear directly as supervision noise: a frame near a transition may be paired with a subtask instruction that is slightly too early or too late. We mitigate this in practice by using trajectory-aware relabeling and semantically coarse subtask phases, but we do not explicitly model boundary uncertainty in the current policy-training path.

At deployment, the executor assumes that the current subtask instruction is approximately correct. If the planner emits an incorrect subtask or revises it poorly, the executor will typically attempt to follow that incorrect local guidance rather than recover autonomously. This is therefore a current failure mode of the modular closed-loop setup: planner errors appear to the executor as conditioning errors.

B. Additional Experimental Details

B.1. Policy-Facing Language Refinement

This subsection describes how we construct policy-facing language prompts at training time and, when applicable, how we adapt them at evaluation time. The procedure is benchmark-specific: LIBERO requires phase-aware relabeling from demonstrations, CALVIN exposes an official current goal or phase signal during evaluation, and LIBERO-Pro adds a low-frequency closed-loop prompt planner on top of LIBERO-trained executors. Unless otherwise specified, the default VLM for both dataset relabeling and evaluation-time prompting/planning is Kimi K2.5.

Shared trajectory-level refinement. We first convert each coarse task annotation into a trajectory-specific executable instruction. A VLM receives the original task instruction together with a sparse set of representative keyframes and rewrites the task into one instruction that describes how the current trajectory solves it. The refiner may add visually supported execution details such as spatial anchors, approach direction, grasp style, motion ordering, or placement strategy, but it must preserve task identity and avoid retrospective commentary. To reduce task drift, we provide contrastive refined instructions from other demonstrations of the same task as additional references and run candidate generation, grounded selection, and QC repair before export. A compact prompt skeleton is:

```
TASK: <original instruction>
KEYFRAMES: initial scene + sparse trajectory images
ADDITIONAL REFERENCES: refined instructions from other visibly
  different strategies of the same task
GOAL: rewrite the task into one imperative instruction for the
  current trajectory's strategy
OUTPUT: one executable instruction for this trajectory only
STYLE: concrete, scene-grounded, imperative, and concise
CONSTRAINTS: preserve task identity; add only visually supported
  details; avoid unsupported objects, frame numbers,
  retrospective narration, and free-form commentary
```

LIBERO training-time prompts. For LIBERO, the trajectory-level instruction serves as an intermediate representation for phase-aware supervision. We decompose each trajectory into an ordered sequence over a small canonical manipulation vocabulary: `approach`, `engage`, `execute`, `disengage`, and `transit`. The decomposition returns phase-labeled subtasks with coarse image anchors, which are then grounded to frame spans and locally refined while keeping the phase structure fixed. The final rendered language view is written into the LeRobot `task` field, so the exported prompt surface is part of the training specification. In the main paper, this view is the goal-preserving hybrid interface consisting of the original instruction together with the active refined subtask instruction. The phase-decomposition interface is:

```
INPUTS: original task, refined trajectory instruction, ordered
  keyframes
PHASE VOCAB: approach, engage, execute, disengage, transit
GOAL: return an ordered phase plan grounded to the observed
  trajectory rather than an abstract language plan
OUTPUT: JSON list with phase, subtask, and image-anchor fields
STYLE: each subtask should be locally executable and phase-aware
CONSTRAINTS: preserve object identity and phase order; refine
  wording locally, but do not split, reorder, or
  rename segments once the coarse decomposition is set
```

CALVIN training and inference. CALVIN already exposes a current goal or phase signal at evaluation time, so we do not perform LIBERO-style phase discovery. For training data, we apply only trajectory-level refinement and render the prompt as `High-level goal: <original goal>` together with `Current instruction: <refined full-trajectory instruction>`. At inference time, the refiner receives the official current goal or phase, recent observations, and robot state, and returns one short executable instruction in strict JSON form under a single `current_instruction` field. In the goal-change mode used in our experiments, this instruction is recomputed only when the official CALVIN goal or phase changes, preventing unnecessary high-frequency language churn while preserving the benchmark-provided task identity. The inference-time prompt skeleton is:

```
OFFICIAL CURRENT GOAL/PHASE: <goal>
RECENT OBSERVATIONS: <images>
ROBOT STATE: <state vector>
GOAL: write one short executable instruction for the official
  current goal or phase only
OUTPUT: JSON with a single current_instruction field
```

```
STYLE: short, imperative, scene-grounded, and policy-facing
CONSTRAINTS: preserve the official goal exactly; return one short
  executable instruction; do not expand into a plan,
  change the goal, or copy retrieved scene details
```

LIBERO-Pro closed-loop prompting. LIBERO-Pro reuses LIBERO-trained executors but evaluates them under a stricter closed-loop prompting protocol. At the start of an episode, an off-the-shelf VLM compiles a trajectory-level strategy from the original task and current scene, converts it into an ordered phase plan, and activates the first local instruction. During execution, the planner periodically reviews recent observations after a minimum commitment window and chooses among `keep_current`, `refine_current`, `next_subtask`, `recover`, and `task_complete`. The planner never outputs robot actions directly; it only revises the policy-facing prompt, while continuous actions remain the responsibility of the trained VLA executor. The review interface is:

```
INPUTS: original task, current active instruction, recent views,
  recent robot progress
ALLOWED DECISIONS: keep_current, refine_current, next_subtask,
  recover, task_complete
GOAL: revise the active policy prompt conservatively, only when
  recent evidence indicates progress, completion, or drift
OUTPUT: one decision + optional revised short imperative
  instruction compatible with the training prompt form
CONSTRAINTS: default to keep_current when execution remains on
  track; advance only when the current phase is visibly
  satisfied; revise language without emitting actions
```

B.2. Dataset Composition

This subsection summarizes the current real-robot datasets used in the TX-G2 and HSR experiments.

TX-G2 dataset composition. The current TX-G2 dataset contains 1,198 source trajectories and 5,287 primitive-action (PA) segments after segmentation. Table 4 summarizes the source-trajectory count for each task family, and the canonical PA sequence for each family is listed below for reference.

Table 4 | TX-G2 dataset composition by task family.

Task family	Source trajs
Bowl Stacking	129
Clothes Sorting	514
Cutlery Transfer	206
Dish Racking	349

Canonical PA sequences. `Bowl Stacking`: Pick up the yellow bowl from the desk → stack the yellow bowl on the grey bowl → pick up the light blue bowl from the desk → stack the light blue bowl on the yellow bowl. `Clothes Sorting`: Pick up the green socks from the desk → place the green socks into the basket → pick up the handkerchief from the desk → place the handkerchief into the basket → pick up the yellow socks from the desk → place the yellow socks into the basket. `Cutlery Transfer`: Pick up the light blue spoon from the grey bowl → place the light blue spoon in the yellow bowl → pick up the pink fork from the grey bowl → place the pink fork in the yellow bowl. `Dish Racking`: Pick up the yellow dish from the desk → place the yellow dish in the wooden dish rack → pick up the green dish from the desk → place the green dish in the wooden dish rack.

HSR dataset composition. The current HSR dataset contains 1,205 source trajectories. Table 5 summarizes the source-trajectory count for each task family, and the canonical PA sequence for each family is listed below for reference.

Table 5 | HSR dataset composition by task family.

Task family	Source trajs
Mug Rectangle	399
Coffee Bottle → Box 1	295
Box Rearrangement	195
Coffee Bottles → Table	316

Canonical HSR PA sequences. `Mug Rectangle` (instruction: form a rectangle by relocating the mug that is not at a rectangle corner): pick up the mug that is not at a rectangle corner → place the mug at the missing rectangle corner. `Coffee Bottle → Box 1` (instruction: place the coffee bottle on the right into the box labeled `1`): pick up the coffee bottle on the right → place the coffee bottle into the box labeled `1`. `Box Rearrangement` (instruction: relocate box number `2` beside the box numbered `1`): pick up the box labeled `2` → place the box next to the box labeled `1`. `Coffee Bottles → Table` (instruction: relocate the two coffee bottles from the shelves to the table beside the mugs): pick up the right-hand bottle from the shelves → place it next to any mug on the table → pick up the left-hand bottle from the shelves → place it next to a mug on the table that does not already have a bottle.

B.3. Backbone and Fine-Tuning Setup

All experiments in the main paper instantiate S2 with the $\pi_{0.5}$ backbone and fine-tune from the 'pi05_base' checkpoint. Unless otherwise specified, we use the 'pi05_libero' configuration for LIBERO experiments and the corresponding OpenPI fine-tuning pipeline.

B.4. Instruction Views

The relabeling pipeline produces multiple language views, including the original instruction, refined trajectory instruction, and refined subtask instruction. The main paper focuses on the goal-preserving hybrid setting, which conditions the policy on both the original instruction and the active refined subtask instruction.

B.5. Visual Budget Defaults

Our implementation supports separate base and wrist soft keep targets. While the method is view-specific by design, the main experiments use a shared-budget setting with $\rho_b = \rho_w = 0.2$ unless otherwise noted.

B.6. CALVIN ABC→D Benchmark

We additionally report CALVIN ABC→D average sequence length in Table 6. Because CALVIN already evaluates staged long-horizon tasks with explicit task boundaries, this benchmark does not exercise closed-loop phase detection in the same way as LIBERO-PRO. We therefore instantiate S2 only through its Specify More language interface, without an additional phase detector or phase-change module. The **S2** row in Table 6 refers to this language-side instantiation. We compare against reported results for OpenVLA-OFT, UniVLA, VLA-Adapter, and X-VLA (Kim et al., 2025; Wang et al., 2025b,c; Zheng et al., 2025).

Table 6 | CALVIN ABC→D average sequence length. Higher is better. Here **S2** denotes the CALVIN instantiation that uses only the Specify More language interface.

Suite	Method	Avg. Len. ↑
CALVIN ABC→D	OpenVLA-OFT	3.27
CALVIN ABC→D	UniVLA	3.80
CALVIN ABC→D	VLA-Adapter	4.42
CALVIN ABC→D	X-VLA	**4.43**
CALVIN ABC→D	$\pi_{0.5}$	3.92
CALVIN ABC→D	**S2**	3.95

B.7. Real-Robot End-to-End Results

Main-text real-robot comparisons report only mean subtask success. Tables 7 and 8 provide the full per-task *Subtask/E2E* breakdowns, where *E2E* counts an episode as successful only if every subtask succeeds consecutively from start to finish. As in the main paper, all compared methods in the real-robot study are fine-tuned for 150k steps before evaluation. OpenVLA-OFT is omitted from these tables because it exceeds memory on the shared RTX 5070 evaluation setup. TX-G2 uses ten trials per task, while HSR uses six.

B.8. Real-Robot Task Definitions

TX-G2. `Cutlery Transfer` requires moving a spoon and a fork individually between bowls without lifting the bowl itself. `Bowl Stacking` stacks the instructed colored bowl. `Clothes Sorting` places the instructed garment into a basket. `Dish Racking` places the instructed plate into a tight rack.

Table 7 | Full TX-G2 real-robot results. For each task, we report mean subtask success with standard deviation and end-to-end success.

Method	Cutlery		Bowl		Clothes		Dish	
	Subtask	E2E	Subtask	E2E	Subtask	E2E	Subtask	E2E
X-VLA	0.0 ±0.0	**0.0**	7.5 ±4.0	0.0	5.0 ±2.6	0.0	5.0 ±3.4	0.0
VLA-Adapter	0.0 ±0.0	**0.0**	0.0 ±0.0	0.0	0.0 ±0.0	0.0	0.0 ±0.0	0.0
$\pi_{0.5}$	5.0 ±3.2	**0.0**	47.5 ±6.8	0.0	83.3 ±4.8	50.0	60.0 ±7.2	20.0
S2	**15.0** ±5.4	**0.0**	**67.5** ±6.8	**30.0**	**96.7** ±2.2	**90.0**	**90.0** ±4.6	**70.0**

Table 8 | Full HSR real-robot results. For each task, we report mean subtask success with standard deviation and end-to-end success.

Method	Coffee		Bottles		Box		Mug	
	Subtask	E2E	Subtask	E2E	Subtask	E2E	Subtask	E2E
X-VLA	8.3 ±7.6	0.0	4.2 ±3.8	0.0	0.0 ±0.0	0.0	25.0 ±12.3	16.7
VLA-Adapter	0.0 ±0.0	0.0	0.0 ±0.0	0.0	0.0 ±0.0	0.0	0.0 ±0.0	0.0
$\pi_{0.5}$	83.3 ±9.6	66.7	45.8 ±8.5	0.0	41.7 ±14.0	16.7	66.7 ±9.6	33.3
S2	**100.0** ±0.0	**100.0**	**87.5** ±6.6	**66.7**	**91.7** ±7.6	**83.3**	**83.3** ±9.6	**66.7**

HSR. The HSR suite combines locomotion with manipulation. In `Coffee`, the robot must navigate to a coffee bottle, grasp it, move to the box labeled `1`, and place it through a narrow opening. `Bottles` consists of bottle-transport tasks under similar mobile-manipulation constraints. `Box` requires transporting the box labeled `2` beside the box labeled `1`, which is challenging because the box is large and relatively heavy for the gripper. `Mug` relocates a mug to the corner that does not already contain one.

B.9. Additional Real-Robot Mask/Attention Comparisons

Figure 9 expands the qualitative comparison in the main paper with additional TX-G2 and HSR examples. Across both robots, the learned VEB masks remain more selective than the backbone's native attention: they concentrate on the manipulated object, end-effector, and destination region needed for the current local behavior, while the backbone attention often spreads across broader scene context or weakly task-related regions. The examples also illustrate that the retained evidence shifts with task phase, from source-object grounding to target-region placement, without any region or mask annotation.

B.10. Cluttered-Scene Stress Tests

We additionally evaluated S2 on a deliberately cluttered TX-G2 clothes-sorting setup designed to probe robustness beyond the training distribution. Only the two sock colors, the handkerchief, and the basket were seen during training; the remaining tabletop objects were unseen distractors. The policy was required to complete the ordered sequence `green socks` → `handkerchief` → `yellow socks`, while a human operator moved distractor objects and, in some cases, the basket itself during execution. Despite these interventions and the changing clutter layouts, S2 completed all three tested rollouts successfully.

Figure 10 shows a representative cluttered observation and the corresponding S2 mask. Even in the presence of many unseen objects, the learned mask concentrates on the task-relevant garment, end-effector, and placement region rather than diffusing across the full scene. Figure 11 then shows three successful rollouts under different initial clutter arrangements and intervention patterns, illustrating that the policy can re-ground the current target and continue the required ordering constraint.

B.11. Why VLA-Adapter Underperforms on TX-G2

We did not find evidence of a major inference-time integration failure for VLA-Adapter. Its preprocessing path, proprio normalization, and bundle-level I/O all passed our validation checks. However, in a 30-sample dataset-driven replay test, its predicted actions were only marginally better than a mean-action baseline, suggesting that the fine-tuned policy remained close to an average regressor rather than a strongly task-conditioned controller. This diagnosis is consistent with its 0% closed-loop success on TX-G2.

B.12. Full Core Ablation Breakdown

Table 9 reports the full perturbation breakdown for the mean-only comparison shown in the main text.

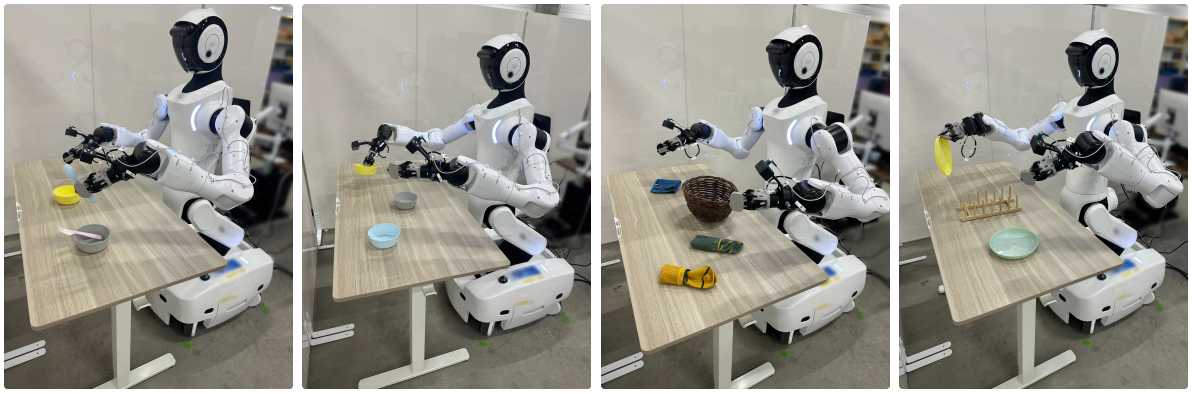


Figure 7 | Representative TX-G2 evaluation trajectories for the four bimanual manipulation tasks used in the appendix real-robot study. Each panel shows keyframes from a successful rollout, highlighting arm selection, semantic object grounding, and precise placement under the TX-G2 benchmark.

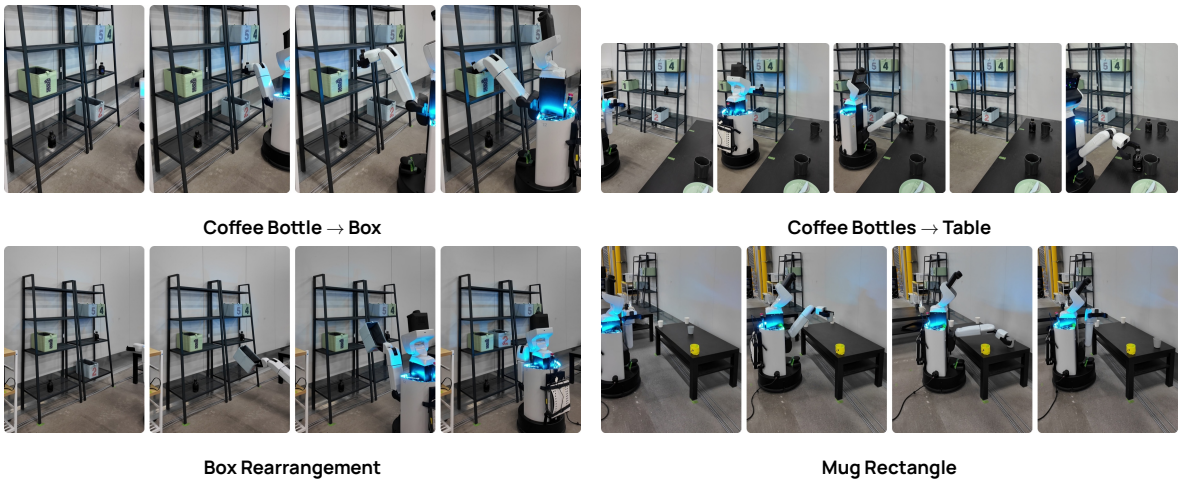


Figure 8 | Representative HSR evaluation trajectories for the four mobile-manipulation tasks used in the appendix real-robot study. Each panel shows keyframes from a successful rollout, illustrating the combination of locomotion, object grounding, and precise placement required by the HSR benchmark.

Table 9 | Full perturbation breakdown for Figure 2. *Original* denotes the original-instruction baseline; all other rows are S2 ablations.

Methods	libero_goal					libero_object				
	Obj.	Pos.	Sem.	Task	Mean	Obj.	Pos.	Sem.	Task	Mean
Original	90.0	29.0	95.0	17.0	57.8	89.0	19.0	95.0	10.0	53.3
Refined	56.0	24.0	61.0	10.0	37.8	78.0	**69.5**	98.5	35.5	70.4
Original + VEB	87.0	36.0	**97.0**	30.0	62.5	**93.0**	33.5	99.5	21.0	61.8
Refined + VEB	52.0	18.0	60.0	10.0	35.0	88.0	68.0	99.5	32.0	71.9
Hybrid, no VEB	56.5	26.0	58.5	10.5	37.9	73.0	48.5	98.5	28.5	62.1
S2	**90.5**	**42.0**	95.5	**52.5**	**70.1**	90.5	64.5	**100.0**	**37.5**	**73.1**

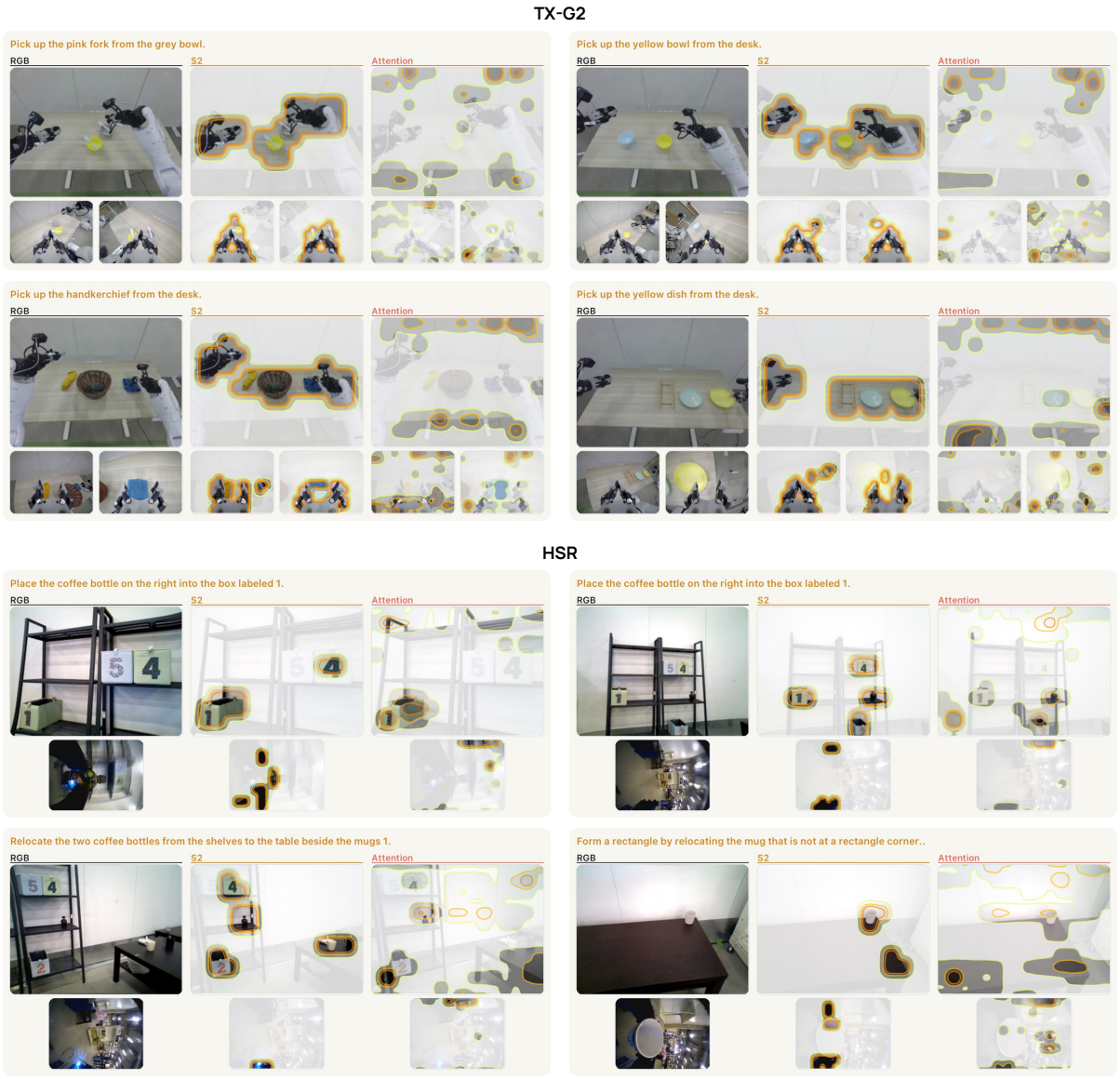


Figure 9 | Additional real-robot qualitative comparisons between the learned visual evidence budget and the backbone's native attention. Each panel shows the RGB observation, the learned S2 mask, and the corresponding $\pi_{0.5}$ attention map for the same state. Across both TX-G2 and HSR, VEB remains more concentrated on the object, contact region, and target placement context relevant to the current behavior, while native attention is often more diffuse.

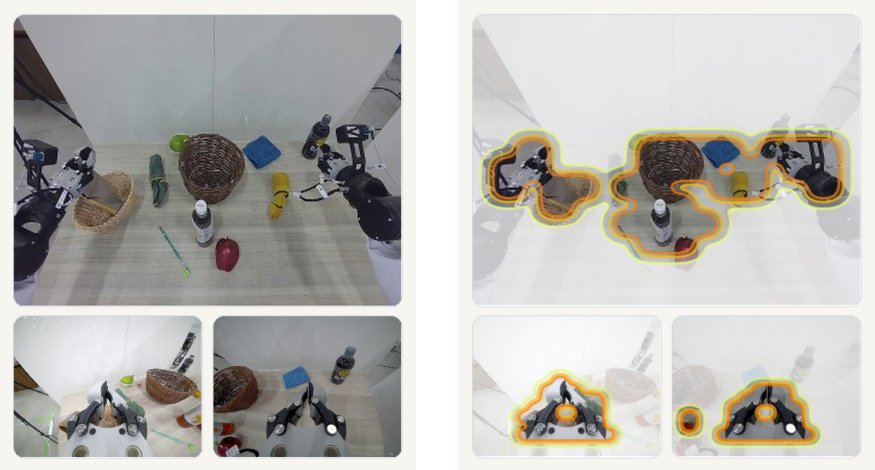


Figure 10 | Representative cluttered TX-G2 observation (left) and the corresponding S2 mask (right) for the ordered clothes-sorting task. The tabletop contains many distractors absent from the training set, yet the learned mask remains concentrated on the task-relevant garment, end-effector, and basket region.

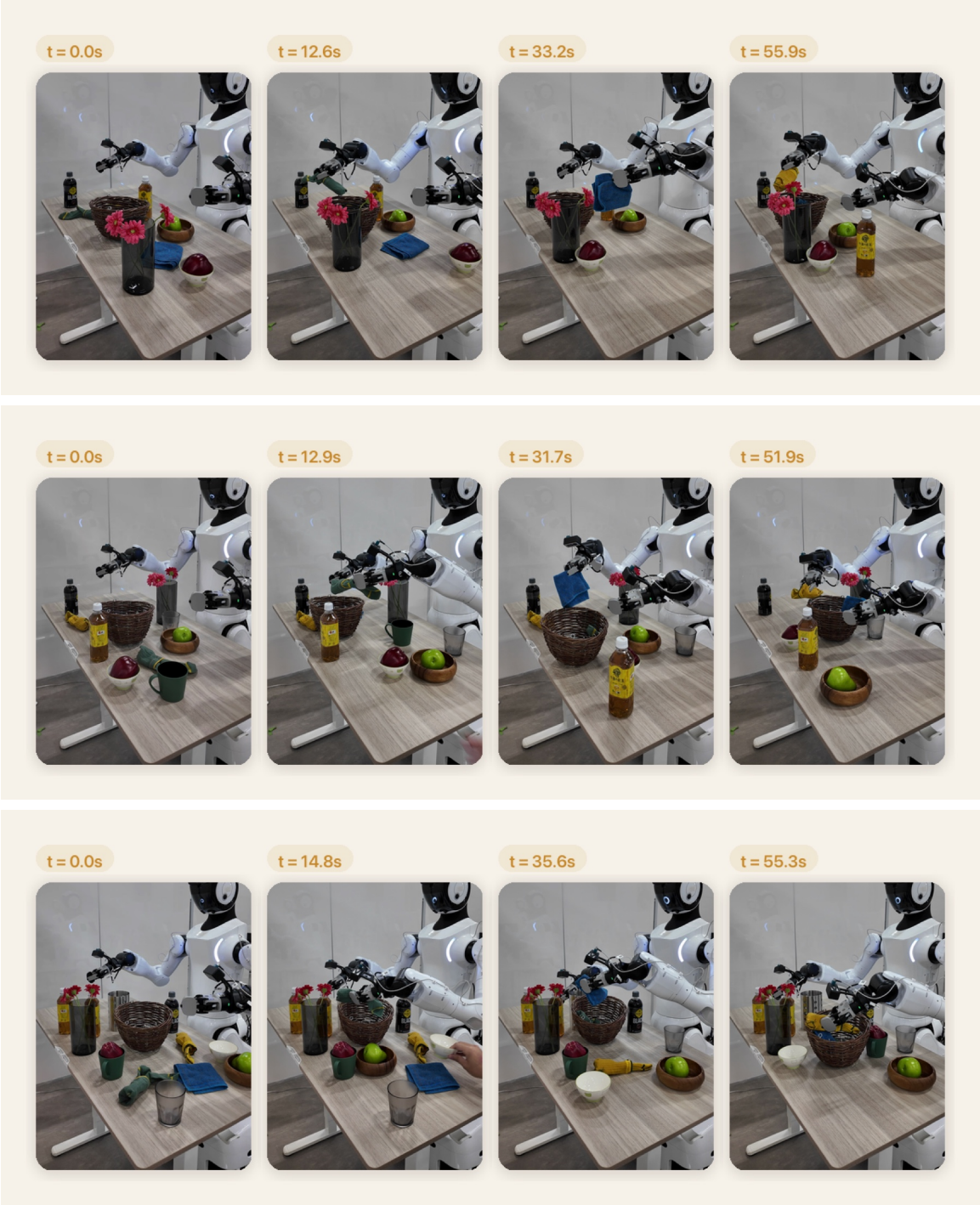


Figure 11 | Three successful TX-G2 cluttered-scene rollouts for the ordered sequence `green socks` → `handkerchief` → `yellow socks`. Each row shows a different initial clutter layout with different unseen distractors. During execution, a person also perturbs object or basket positions, forcing the policy to re-ground the current target online while preserving the required completion order.