arXiv:2608.11739v1 [cs.RO] 12 Aug 2026

Galaxea G0.5: One Autoregressive Stream for Robot Reasoning and Action

Galaxea Team
https://opengalaxea.github.io/G05/

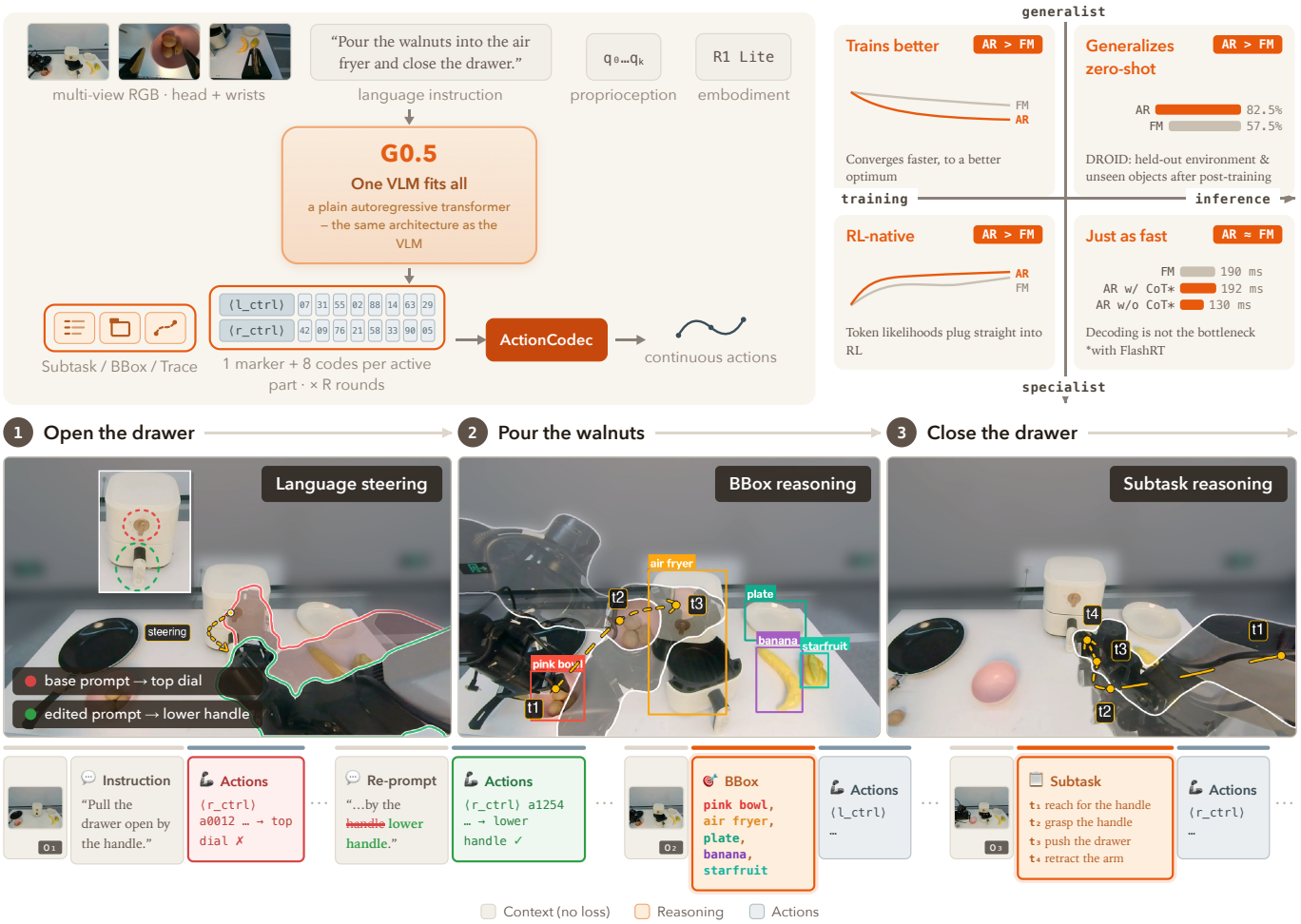


Figure 1: **G0.5: reasoning and action in one autoregressive stream. Top left:** A single VLM conditions on multi-view images, an instruction, proprioception, and an embodiment identifier, then generates optional chain-of-thought (CoT) and compact action codes under the same next-token objective. A cross-embodiment ActionCodec converts the codes into continuous motor commands, while active-part tokenization omits inactive control groups. **Top right:** Comparisons with flow-matching policies span generalist pretraining and specialist post-training, covering optimization, compatibility with reinforcement learning, transfer after DROID post-training to a held-out physical setup and previously unseen object instances, and inference latency. **Bottom:** A closed-loop rollout of *open the drawer* → *pour the walnuts* → *close the drawer*. At each stage, updated visual and proprioceptive observations condition interleaved reasoning and action generation. The examples highlight prompt-driven action steering (left), grounded bounding-box prediction (middle), and adaptive subtask reasoning (right); colors distinguish conditioning context, reasoning, and action tokens.

Abstract

The prevailing recipe for Vision-Language-Action (VLA) models couples a pretrained VLM with a separately trained flow-matching action expert. This makes the VLM a context encoder rather than a decision-maker. Instead, we argue for focusing on the VLM backbone: a unified model with a single set of weights that generates both reasoning and actions within a single autoregressive token stream.

We introduce **G0.5**, a pretrained autoregressive VLA in which a single transformer decoder emits reasoning and action tokens under a single objective. Three components make this tractable at foundation-model scale: a learnable cross-embodiment action tokenizer that maps heterogeneous robot actions into a shared vocabulary; a native chain-of-thought stream interleaving task decomposition, object grounding, and action hints with action tokens; and a visual memory module that injects multi-second history through the vision encoder. Because reasoning and action share a single set of weights, the pretrained VLM's capabilities carry over to physical behavior: the model follows instructions closely, and prompts directly steer action granularity, task horizon, and out-of-distribution scene handling without further training. Pretrained on a large collection of robot datasets together with VQA samples, **G0.5** surpasses state-of-the-art models across 7 independent regimes: real-world fine-tuning on R1-Lite/R1-Pro robots (76.7% vs. 53.3% for $\pi_{0.5}$ and 24.4% for GR00T-N1.7), the 2025 BEHAVIOR Challenge on 50 long-horizon household mobile manipulation tasks using a generalist policy (31.4% vs. 26.3% for $\pi_{0.5}$ and 26.1% for the challenge winner), DROID post-training followed by zero-shot transfer to an unseen environment and objects (82.5%), a language-following Pick-and-Place benchmark, LIBERO (98.9%), RoboTwin 2.0 (93.3%), and SimplerEnv-Bridge (87.3%).

1 Introduction

Vision-Language-Action (VLA) models have rapidly emerged as a leading paradigm for general-purpose robot control, extending large-scale vision-language pretraining from perception and language understanding to physical action [1, 2, 3]. Early VLA systems used an autoregressive interface that cast robot control as token generation: continuous actions were discretized, appended to the language vocabulary, and predicted by the VLM alongside text tokens [1, 2]. This keeps the VLM itself as the actor, but scales poorly. As control frequency, action horizon, and action dimensionality increase, per-timestep autoregressive action tokens grow rapidly, making high-frequency control slow and expensive. This bottleneck pushed the field toward VLM-as-encoder architectures, where a pretrained VLM supplies hidden states or KV cache to a separately trained flow-matching or diffusion expert that predicts continuous action chunks [3, 4, 5, 6].

This shift improves action efficiency, but changes the role of the VLM. In VLM-as-encoder models, the VLM is no longer the action generator; it becomes a vision-language condition encoder, while the final action distribution is produced by an expert with separate parameters and a separate objective. Consequently, core generative capabilities of VLMs—chain-of-thought reasoning, in-context learning, and prompt-based motion steering—can affect behavior only after passing through a compressed conditioning bottleneck, rather than as native parts of action generation [7, 8, 9]. We therefore return to the autoregressive formulation, but remove the source of its original inefficiency: excessive action tokenization. A learning-based VQ tokenizer compresses action chunks into compact discrete codes, while active degree-of-freedom prediction avoids spending tokens on robot joints that do not need to move. Together, these choices substantially reduce the decoding burden while preserving the VLM as a generative actor. As part of the pretrained backbone, we also retain a lightweight visual-memory mechanism that feeds accumulated visual context through the vision encoder, following recent memory-augmented VLA designs [10], since persistent visual context benefits long-horizon control and closed-loop replanning. More importantly, once reasoning and action share the same autoregressive stream, chain-of-thought can be trained as a native component of control: the model can

zero-shot decompose an instruction into subtasks, identify task-relevant objects and their bounding boxes, and feed these intermediate predictions directly into subsequent action generation (Fig. 1).

We introduce **G0.5**, a pretrained autoregressive VLA in which a single model reasons, plans, and acts within a unified token stream spanning images, language, reasoning traces, and actions. Our contributions are as follows:

(1) A unified heterogeneous action codec. We pretrain a learning-based action codec that maps continuous action sequences from embodiments with different degrees of freedom, control frequencies, and morphologies into a shared token vocabulary. Unlike FAST, which applies a fixed DCT-based pipeline separately to each embodiment [11], our codec is learned end-to-end and cross-embodiment by design. It allows the VLM to represent actions from different robots through a common discrete interface, making autoregressive VLA practical at foundation-model scale; Fig. 1 illustrates this on R1-Lite, where the active-token layout adapts on the fly to whichever parts are in motion and drops the idle arm's token group from the stream entirely rather than padding it. **(2) Native chain-of-thought through autoregressive training.** We construct a family of CoT templates for task decomposition, scene grounding, and sub-goal sequencing, and train the model to emit reasoning tokens before and between action tokens in the same autoregressive stream. Unlike CoT-VLA, DualCoT-VLA, and related approaches that attach reasoning modules to VLM-as-encoder backbones [7, 8, 9, 12], our CoT tokens share the decoder, context, and objective with the action tokens. Reasoning and action are therefore not separate stages, but coupled phases of one generative process (see the interleaved CoT and action segments in Fig. 1). This design yields two benefits that we evaluate separately: stronger grounding and execution under long-horizon instructions—including zero-shot execution of household tasks under stage-conditioned instructions outside the pretraining distribution (Sec. 5.6)—and improved language following beyond what the codec alone provides. **(3) Emergent prompt-driven behavior control.** Preserving the autoregressive interface keeps the VLM's in-context language capacity directly wired to action generation, in principle enabling prompt-level steering of physical behavior without retraining. In our zero-shot probes (Sec. 5.6) we see preliminary qualitative indications of this—per-stage instruction wording such as adverbial qualifiers, spatial cues, or near-synonymous verbs visibly shifts policy behavior—and leave a systematic study to future work. We suspect this capacity is partly structural to the autoregressive interface: when the VLM only conditions an external expert, prompts can shape the condition but cannot directly reshape the next-action distribution.

We evaluate **G0.5** across seven settings that probe distinct facets of a general-purpose VLA: real-world fine-tuning on the R1-Lite and R1-Pro bimanual platforms, the BEHAVIOR-1K Challenge on 50 long-horizon household mobile-manipulation tasks [13], DROID post-training with zero-shot transfer to an unseen environment and objects, a Pick-and-Place language-following benchmark, and three standardized simulation suites (LIBERO, RoboTwin 2.0, SimplerEnv-Bridge) [14, 15]. We compare against representative baselines from three model families: VLM-as-encoder models [3, 4, 5, 6], autoregressive models [2, 11], and the recently popular world action models [16]. Three findings stand out. First, on standard task success metrics, **G0.5** matches or surpasses the strongest baselines across these families—98.9% on LIBERO, 93.3% on RoboTwin 2.0, 87.3% on SimplerEnv-Bridge, 82.5% on DROID with environment- and object-level zero-shot transfer, and 76.7% on the R1-Lite and R1-Pro platforms, compared with 53.3% for $\pi_{0.5}$ and 24.4% for GR00T-N1.7—indicating that the pretrained **G0.5** backbone transfers effectively to downstream control across these heterogeneous suites. Second, on language following and multi-stage execution under stage-conditioned prompts, **G0.5** substantially outperforms VLM-as-encoder baselines on the Pick-and-Place benchmark and the BEHAVIOR-1K Challenge, where a single **G0.5** checkpoint trained for one post-training epoch already surpasses both $\pi_{0.5}$ trained for four epochs and the four-checkpoint Challenge winner. This is consistent with our argument that these capabilities are structurally weakened when the VLM is reduced to a condition encoder. Third, a small qualitative probe on two zero-shot long-horizon household tasks (Sec. 5.6) suggests that prompt wording—adverbial qualifiers, spatial cues, and verb substitutions—can shift AR+CoT rollouts without retraining; we report this as an early hook for prompt-level behavior steering rather than a quantitative claim, and defer a systematic study to future work. Taken together, our results suggest that the path forward for VLA is not to place increasingly sophisticated action experts on top of an underused VLM, but to let the VLM remain what pretraining made it: an autoregressive reasoner that can also act, remember,

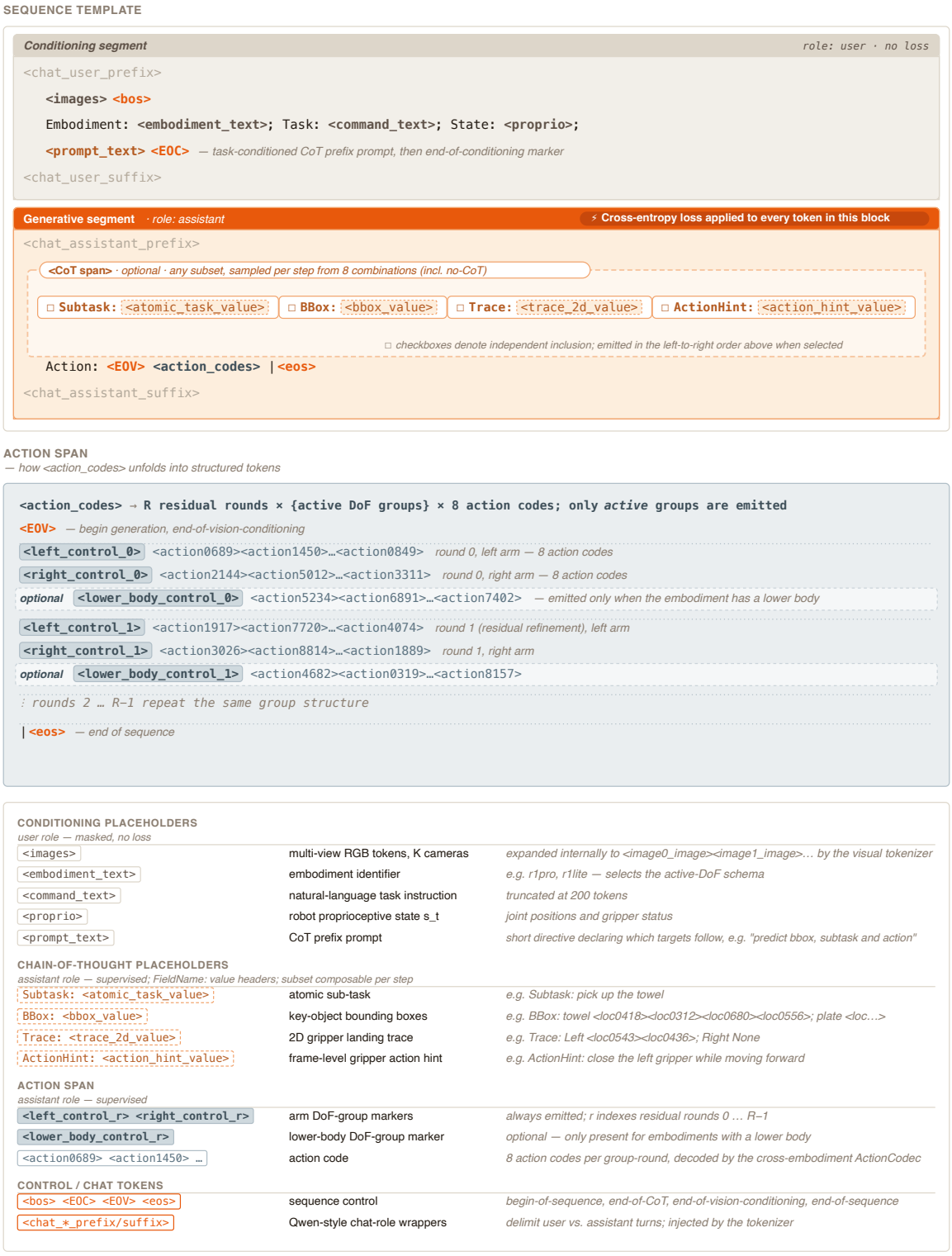


Figure 2: **Token sequence template.** All inputs and outputs are serialised into a single autoregressive sequence: a *conditioning segment* (multi-view RGB, embodiment id, task instruction, proprioceptive state—in user-side chat tokens) and a *generative segment* on which the next-token cross-entropy loss in Eq. (1) is applied. The generative segment composes an optional chain-of-thought span—any subset of four self-describing reasoning targets (`Subtask:`, `BBox:`, `Trace:`, `ActionHint:`)—followed by the action codes, which themselves expand into R residual rounds of DoF-group markers each followed by 8 action codes (Sec. 3.1).

and adapt in context. We hope this work re-establishes autoregressive modeling as a foundation for VLA and that the pretrained backbone we release provides a useful starting point for future work.

2 Related Work

2.1 VLA Architectures: from VLM-as-Encoder to VLM-as-Actor

Vision-language-action models split along one architectural axis: whether the VLM produces actions or only conditions a separate module that does. The dominant line couples a pretrained VLM with an action expert that consumes its features and emits continuous actions via diffusion or flow matching: π_0 [3] introduced a separately-parameterized expert with block-wise causal attention, and $\pi_{0.5}$ [4], GR00T-N1 / N1.5 / N1.6 [5], and SmolVLA [6] follow variants of the same template. The autoregressive line, including RT-2 [1], OpenVLA [2], and π_0-FAST [11], instead discretizes actions and predicts them with the VLM itself under next-token prediction. The two lines are typically presented as a trade-off—continuous heads for smooth high-frequency control, AR for reasoning and simplicity—but they also differ in what the VLM is *for*: in the first line, the VLM is a condition encoder whose pretrained reasoning is exercised only indirectly, while in the second, it remains the agent that acts.

A revealing thread within the VLM-as-encoder line is the anti-forgetting problem: when the action expert's gradients flow back into the VLM, the VLM's pretrained perception and language capabilities degrade [17, 18]. The mainstream remedy, Knowledge Insulation [17], stops these gradients and reintroduces AR action prediction as an auxiliary representation-learning objective for the backbone—implicitly conceding that AR action supervision is exactly the signal that protects the VLM's capabilities. Recent results push further: VLA-0 [18] shows that an unmodified VLM trained AR on actions-as-text outperforms $\pi_{0.5}$-KI, OpenVLA-OFT, and SmolVLA on LIBERO without large-scale action pretraining, providing direct evidence that the AR paradigm is not the bottleneck. Our work takes this signal seriously and commits to the AR line end-to-end, retaining a flow-matching head only as an optional inference accelerator. What remains open after VLA-0—and what the rest of this section traces—is how to scale the AR paradigm beyond a single low-frequency embodiment with closed-vocabulary tasks: through a tokenizer that respects morphological structure (Sec. 2.2) and reasoning that grounds language in action (Sec. 2.3).

2.2 Action Tokenization and Cross-Embodiment

Action tokenization for VLAs has progressed through three generations. Per-dimension, per-timestep binning, as in RT-2 [1] and OpenVLA [2], fails on high-frequency dexterous data because adjacent timesteps are strongly correlated and binning wastes capacity [11]. FAST and FAST+ [11] replace binning with DCT plus byte-pair encoding, exploiting that correlation as compressible signal, and FAST+ is trained on one million trajectories to serve as a universal tokenizer. Neural and vector-quantized variants—VQ-VLA [19], BEAST [20], and earlier VQ-BeT [21]—push reconstruction quality further at the cost of joint training and more complex pipelines.

Cross-embodiment generalization is largely orthogonal to all three. Mainstream VLAs handle morphological heterogeneity at the action-space level rather than the tokenizer: π_0 [3] pads all robots to an 18-dim union state, GR00T-N1 [5] uses per-embodiment MLP encoders and decoders, and SpatialVLA [22] unifies action spaces via adaptive grids. The closest neighbors to our work are Being-H0.5 [23], which maps heterogeneous robot controls into semantically aligned slots and even folds the MANO hand model into the same scheme, Green-VLA [24], which retargets across robots by aligning corresponding parts into a unified action space, and HEX [25], whose humanoid-aligned state representation operates on canonical body-part abstractions. All three operate at the action-vector level. Our contribution is to lift the same structural alignment into the *tokenizer itself*: a single frozen codec consumes a 5-part fixed-dimensional layout and emits a unified 27-dim action token stream, so left/right symmetry is preserved by construction and adding a new embodiment requires no new parameters in either the tokenizer or the action head.

2.3 Reasoning and Chain-of-Thought in VLAs

Two families have emerged for injecting reasoning into VLAs. *Bolt-on* CoT routes natural-language plans or 2D paths from a high-level VLM into a separate low-level controller, as in HAMSTER [26] and Fast-in-Slow style System-2-feeds-System-1 designs [5]; the reasoning is an interface between modules rather than a co-generated component of the action. *In-stream* CoT, by contrast, generates reasoning and action in the same AR sequence from the same decoder. ECoT [27] reports a 28-point absolute improvement on OpenVLA by training it to predict plans, subtasks, motions, bounding boxes, and end-effector positions before actions; CoT-VLA [7] replaces text reasoning with autoregressively-generated subgoal images; Emma-X [28] predicts look-ahead 2D gripper checkpoints; and $\pi_{0.5}$ [4] emits high-level subtask text from the VLM before invoking its flow-matching expert. Our setting is closest in spirit to ECoT in that reasoning and action share a single AR decoder, but differs along two axes that matter for our claims: we combine four reasoning primitives—object bounding boxes, atomic subtask text, 2D end-effector traces inspired by TraceVLA [29], and action hints—in one shared token vocabulary, and we expose them as *prompt-conditional* templates, letting the CoT mode be switched at inference without retraining.

3 G0.5 Model Design

We design our model around a single commitment: perception, reasoning, and action should be unified within a single autoregressive process over a shared token vocabulary. This commitment shapes every component below—the action representation, the reasoning scaffold, the visual conditioning, and the training objective—and distinguishes our design from VLM-as-encoder architectures in which action generation lives in a separate module with a separate objective.

Our model is initialized from Qwen3.5 2B [30], a pretrained vision-language model that provides a strong visual encoder, a shared multimodal token vocabulary, and an autoregressive decoder. At inference, given (i) a short temporal window of multi-view RGB observations $\{o_{t-h}^{(k)}\}$ from K cameras, (ii) an embodiment identifier e (e.g., R1-Pro), (iii) a natural-language task instruction ℓ, and (iv) a proprioceptive state s_t, the model autoregressively generates a structured output that concludes in a sequence of discrete action codes. Depending on the prompt template, the generation can optionally be preceded by chain-of-thought (CoT) segments that ground objects, decompose subtasks, or sketch gripper traces. The action codes are decoded by our cross-embodiment ActionCodec into continuous control commands in a unified action space shared across embodiments. The autoregressive VLM is self-contained and serves as the default policy in all main experiments. For comparison and optional deployment, we additionally attach a flow-matching head that follows the action-expert architecture of $\pi_{0.5}$ [4] and conditions on the autoregressive trunk's hidden states.

All inputs and outputs are serialized into a single token sequence following the template in Fig. 2. The sequence is partitioned into a *conditioning segment*—wrapping images, embodiment, task, and state in user-side chat tokens and terminated by <EOC>—and a *generative segment*—wrapping the CoT trace and action codes in assistant-side chat tokens, with <EOV> marking the boundary between reasoning and action emission. Training uses the standard next-token cross-entropy loss, computed only over the generative segment:

$$\mathcal{L}(\theta) \;=\; -\sum_{i \in \mathcal{G}} \log p_\theta\big(x_i \mid x_{<i}\big), \tag{1}$$

where $\mathcal{G}$ indexes the generative-segment tokens. Crucially, this single loss jointly supervises CoT generation and action generation: there is no auxiliary regression objective or expert distillation in pre-training. CoT traces and actions are all "just tokens" to the decoder, drawn from the same vocabulary and produced by the same forward pass.

The remainder of this section unpacks the three components of the generative segment in the order they were derived: the cross-embodiment action codec (Sec. 3.1), the chain-of-thought scaffold (Sec. 3.2), and short-term visual memory (Sec. 3.3).

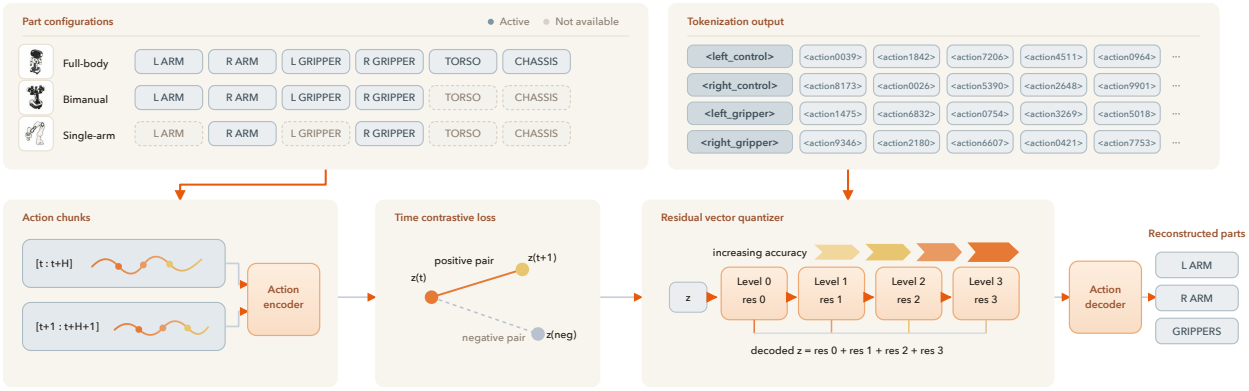


Figure 3: **Structured action tokenization.** Heterogeneous robot actions are decomposed into semantically aligned motion parts, encoded with a residual vector quantizer, and serialized as part-specific action tokens. This representation shares one action vocabulary across embodiments while allowing sparse prediction over only the activated parts.

3.1 Structured Tokenization of Heterogeneous Action Data

A key challenge is how to represent heterogeneous actions from diverse embodiments in a structured token space that VLMs can efficiently model. Existing approaches suffer from two major limitations: **(a) lack of structural decomposition.** Most methods flatten the entire action space into a single vector before discretization [11, 19, 31, 32], regardless of embodiment topology or controllable degrees of freedom (DoFs). This results in semantically entangled action tokens that transfer poorly across embodiments. In addition, token count scales directly with the total number of controllable DoFs, despite the fact that only a small subset of joints are typically active at each timestep. **(b) poor token consistency.** Discrete action spaces are usually learned without explicit structural constraints, causing semantically similar actions to map to token sequences with large Hamming distances [32]. As supervision signals for VLM training, such inconsistency introduces substantial optimization noise and reduces training efficiency.

To address these issues, we adopt the action grouping strategy of FASTer [33] together with the training recipe of ActionCodec [32]. Specifically, we decompose each robot into independent motion parts (e.g., left control, right control, lower body), and pad each part to a shared maximum dimensionality before training a residual vector quantization (RVQ) model over the grouped actions. We further introduce a temporal contrastive objective to improve token consistency across temporally adjacent motions. During tokenization, we explicitly inject structural special tokens into the sequence. Concretely, the action span shown in the generative segment of Fig. 2 unfolds into R residual rounds, each containing the currently active DoF-group markers (`<left_control_r>`, `<right_control_r>`, and optionally `<lower_body_control_r>` for embodiments with a lower body) followed by their 8 action codes. This formulation allows the model to predict only the motion parts that are actively involved in the current behavior. In practice, the proposed structured tokenization significantly improves training efficiency, enables heterogeneous embodiments to share a unified action configuration, and naturally supports sparse action prediction during inference, where inactive parts remain stationary without requiring additional token generation. We show the details in Figure 3.

3.2 Native Chain-of-Thought

To preserve or further enhance the physical intelligence of VLMs, previous methods typically co-train auxiliary VQA tasks, such as sub-task or object bounding box prediction. However, these objectives are treated only as training-time supervision and never explicitly participate in the action generation process itself, making it difficult to directly assess whether such intermediate reasoning signals truly benefit downstream action prediction. In contrast, we leverage the unified autoregressive formulation of our model to naturally

integrate these auxiliary tasks into the action generation stream as native chain-of-thought (CoT) reasoning. Instead of treating reasoning-related annotations as isolated supervision targets, the model is trained to optionally perform intermediate reasoning before action prediction across four self-describing targets—task decomposition (`Subtask:`), key-object localization (`BBox:`), motion planning (`Trace:`), and action hints (`ActionHint:`)—which populate the CoT span shown in the generative segment of Fig. 2. Any subset of these targets can be emitted at each step, and we draw from 8 curated combinations (including a no-CoT baseline) per training step, all supervised within the same next-token objective.

Surprisingly, the resulting CoT capability exhibits strong zero-shot generalization. On unseen scenes and tasks, the model is able to generate accurate subtasks, proactively identify task-relevant objects together with their bounding boxes, and predict additional reasoning traces such as 2D motion trajectories and action hints. More importantly, enabling CoT reasoning consistently improves instruction-following behavior and action accuracy on complex manipulation tasks. These results suggest that intermediate reasoning is not merely an auxiliary supervision signal, but can serve as an effective test-time guidance for embodied action generation.

3.3 Visual Memory

Complex mobile manipulation tasks are inherently non-Markovian. Relying solely on single frame observations often fails during temporary occlusions from robotic arms or environmental clutter, and lacks the temporal context needed to recognize failures and formulate alternative retry strategies. However, resolving partial observability by naively stacking historical vision tokens introduces severe limitations. It scales quadratically in computational cost, causing unacceptable latency for high-frequency control, and makes the model highly susceptible to error accumulation and state drifting when encountering unseen temporal trajectories.

To overcome these challenges, we follow $\pi_{0.7}$ [34] and MEM [10] by inserting factorized spatial and temporal attention modules every four layers within the Vision Transformer. This separable mechanism efficiently fuses historical context by sequentially mixing information across time steps and spatial patches. To strictly bound computational latency, we discard all historical tokens at the final layer, and stochastically drop all historical frames during training to prevent overfitting. Finally, we replace discrete text tokenizers with continuous state embeddings to perfectly synchronize proprioceptive inputs with the corresponding visual frames.

4 G0.5 Pre-training

We pre-train **G0.5** in a single stage on a heterogeneous mixture of robot demonstrations and web-scale vision–language data. The model, tokenization, chain-of-thought (CoT) stream, and visual-memory module are described in Sec. 3; here we specify only the data composition, the sampling and supervision recipe, and the optimization setup.

Robot data mixture. The robot portion of the pre-training mixture covers **14 embodiments** across diverse real-world and simulated robot ontologies. DROID data are not part of this foundation pre-training mixture; for the evaluation in Sec. 5.1, the resulting model is subsequently post-trained on DROID data while excluding any demonstrations from the held-out evaluation environment and physical object instances. All sources are cast into a single **27-dimensional unified action space**, partitioned as

```
<left_control>(9) | <left_gripper>(1) | <right_control>(9) | <right_gripper>(1) | <lower_body>(7)
```

Slots that a given embodiment does not actuate are filled with `noop` tokens at merge time, so embodiments of differing morphology share one output head without per-robot adapters. We treat each embodiment individually for pre-processing: the action normalization mode (z-score with tail clipping, or q_{01}/q_{99} quantile scaling) and the per-channel action filters are set per source rather than mixture-wide.

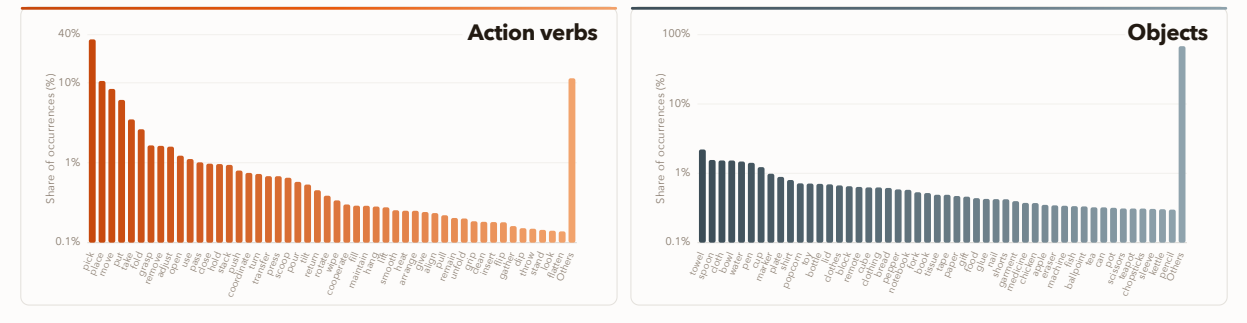


Figure 4: **Action and object concept distribution in the pre-training corpus.** We show the most frequent action verbs and object nouns extracted from the pre-training data; each bar gives the concept's share of occurrences (%) on a logarithmic axis, and the trailing *Others* bar aggregates the remaining vocabulary.

To characterize the semantic coverage of the pre-training corpus, we further analyze the frequency distribution of action and object concepts. As shown in Fig. 4, both action verbs and object nouns exhibit a clear long-tailed distribution. High-frequency actions are dominated by general manipulation primitives such as picking, placing, moving, and putting, while the object vocabulary is concentrated on common household and tabletop entities. This distribution indicates that the corpus provides broad coverage of everyday robot manipulation scenarios while retaining a diverse tail of less frequent skills and objects.

Autolabeling pipeline. To enrich the annotation signals available in large robot manipulation corpora, we build an automated multimodal labeling pipeline that converts raw episodes into multi-granularity semantic annotations, visual grounding annotations, and action-trajectory annotations. For language annotation, we first apply rule-based temporal segmentation to identify candidate action segments and keyframes, and then query multimodal model APIs such as Gemini 3 [35] and Doubao Seed 2.0 Pro [36] to generate action hints, atomic task descriptions, and episode-level instructions. These multi-granularity language annotations allow us to train and query the policy under different instruction granularities and to construct CoT pairs for training intermediate reasoning. For visual grounding, we combine multimodal foundation models followed by SAM3 tracking [37] to generate per-frame bounding boxes and segmentation masks for task-relevant objects. Finally, for 2D end-effector traces, we compute bimanual end-effector positions from robot joint poses using forward kinematics and project the resulting 3D trajectories onto the head-camera image plane.

Web and VQA co-training. To retain the VLM's general language capability and broad generalization while strengthening its spatial perception, we co-train with a large-scale vision–language mixture spanning generic web VQA [38, 39], embodied VQA [40, 41, 42], and in-house annotations generated by the autolabeling pipeline above. The in-house portion covers subtask decomposition, object bounding boxes, and general commonsense VQA over our robot scenes. During pre-training, VQA and action samples are combined in an action-heavy mixture. Both sample types are optimized with the same next-token cross-entropy loss over their target tokens, so language answers, CoT traces, and action codes are all supervised within the unified autoregressive decoder.

Chain-of-thought supervision. Each robot sample is assigned exactly one CoT format, drawn by weighted random sampling from eight candidates: a no-CoT baseline, atomic-task and high-level-task text, subtask text, subtask-with-action-hint, 2D trajectory traces, and bounding-box (object-localization) variants. The subtask-text format is assigned a higher sampling weight, while the remaining formats use the default weighting. This mirrors the increased emphasis placed on the in-domain subtask-prediction split on the VLM side, reflecting a consistent focus on subtask grounding across both modalities. Evaluation uses the fixed no-CoT format.

Implementation Details. We optimize a single cross-entropy objective over the shared vocabulary using AdamW with a warmup phase followed by a constant phase and a final cosine decay, while keeping the vision tower unfrozen throughout. The observation input provides a sparse multi-second history that includes the current frame. Historical frames are randomly dropped during training as regularization for the visual-memory module (Sec. 3.3), and the model is trained until convergence.

5 Experiments

We design our experiments to comprehensively probe the capabilities of **G0.5** along the axes that matter most for a general-purpose VLA: out-of-the-box deployability, transferability via fine-tuning, scalability to long-horizon tasks, fidelity to language, and adaptability to different contexts. Concretely, our evaluation is organized around the following research questions:

- **Q1: How well does G0.5 generalize beyond its training environments and objects?** After post-training on DROID data, we deploy **G0.5** on a DROID Franka platform whose physical environment and object instances are absent from both pre-training and post-training; we also stress-test its instruction-following capability under our Pick-and-Place Benchmark on the R1-Lite robot (Sec. 5.1, Sec. 5.5).
- **Q2: How effectively can G0.5 be adapted to out-of-domain benchmarks?** We fine-tune **G0.5** on external robot datasets, including DROID and Bridge, and evaluate it on the corresponding hardware and simulation suites. This measures how well the pretrained representation transfers when both the embodiment and the data distribution differ from our in-house platforms (Sec. 5.1, Sec. 5.2.1).
- **Q3: How does G0.5 perform on in-domain tasks after fine-tuning?** We evaluate two complementary in-domain settings: standardized simulation benchmarks (LIBERO, RoboTwin 2.0) for reproducibility and comparison with prior work, and real-world fine-tuning on R1-Lite and R1-Pro across six task-embodiment settings to measure long-horizon bimanual manipulation under matched training and evaluation conditions (Sec. 5.2.3, Sec. 5.2.2, Sec. 5.4).
- **Q4: Can G0.5 acquire long-horizon generalist mobile manipulation skills, and how do architectural choices and pre-training data distribution shape this capability?** We evaluate **G0.5** with a single policy on the 2025 BEHAVIOR Challenge, a 50-task household benchmark where each episode averages 6.6 minutes and demands coordinated navigation and bimanual manipulation. We analyze how architectural choices and pre-training data distribution shape downstream long-horizon performance (Sec. 5.3).
- **Q5: How strong is G0.5's language-following ability in cluttered scenes?** We introduce the Pick-and-Place Benchmark (PP Bench), which disentangles language grounding from low-level execution by separately reporting language following rate and task success rate across in-distribution and out-of-distribution object categories at multiple post-training scales (Sec. 5.5).
- **Q6: How do different contexts affect G0.5's behavior?** We study how augmenting the policy input with additional referring context, such as cropped object/container regions and coordinate tokens provided by an external VLM, influences language grounding and final task success, leveraging the flexible multi-image interface of **G0.5** (Sec. 5.5).
- **Q7: Does putting reasoning in the same stream as action actually pay off?** On a single pretrained checkpoint we toggle the action head (AR tokens vs. an additional flow-matching head) and the CoT stream (on/off) at inference time, across PP Bench and two new zero-shot long-horizon household tasks. We also qualitatively observe how per-stage instruction wording—e.g., adverbial qualifiers, spatial cues, or near-synonymous verbs—affects rollouts under AR+CoT (Sec. 5.6).

The remainder of this section is organized to answer each of these questions in turn.

5.1 DROID Environment- and Object-Level Zero-Shot Evaluation

We first post-train **G0.5** on the DROID dataset [43] and then deploy it on a held-out DROID robot setup. No demonstrations from the physical evaluation environment or involving the evaluation object instances

are included in either pre-training or post-training. This evaluation therefore measures environment- and object-level zero-shot generalization after DROID post-training, rather than dataset-level zero-shot transfer.

5.1.1 Evaluation Setup

Robot Platform. We use a Franka Research 3 7-DoF robot arm equipped with a Robotiq 2F-85 parallel-jaw gripper, mounted on a height-adjustable standing desk following the standard DROID hardware configuration [43]. Visual observations are provided by two RGB cameras: a right-side third-person camera offering a fixed global view of the tabletop workspace, and a wrist-mounted camera providing a close-up view for fine-grained manipulation. The policy receives both camera streams together with the natural language task instruction.

Tasks. We evaluate on 10 tabletop manipulation tasks drawn from the DROID environment [43], as illustrated in Fig. 5. Tasks span seven skill categories, each targeting a distinct manipulation challenge:

- **Move the carrot / peach into the bowl.** Requires discriminating between a soft deformable carrot plush and a rigid spherical peach, and precisely depositing the target object into a small bowl.
- **Move the block onto the green / red plate.** Requires colour-conditioned target selection and fine-grained grasping of a small block onto the correct plate.
- **Move the block into the cup.** Requires accurate vertical clearance estimation to deposit the block inside a tall cup without colliding with the rim.
- **Put the towel / pen into the open drawer.** Requires identifying the target object, localising the drawer opening, and grasping both deformable fabric and a thin rigid tool.
- **Move the bowl to the left.** Requires spatial-direction understanding and stable grasping of a wide, irregularly shaped bowl for precise lateral displacement.
- **Take out the towel from the bowl and put it on the plate.** A two-step sequential task: extract a deformable towel from a bowl, then reposition it onto a flat plate.
- **Put the block into the open drawer and close the drawer.** A long-horizon task requiring two temporally dependent sub-goals: block insertion followed by drawer closure.

Evaluation Protocol. Each task is evaluated over 10 trials. Task success is scored as a binary outcome (1 for completion, 0 otherwise), except for the sequential task *put the block into the open drawer and close the drawer*, which receives a partial score of 0.5 for completing only the insertion sub-step and a full score of 1.0 for full task completion.

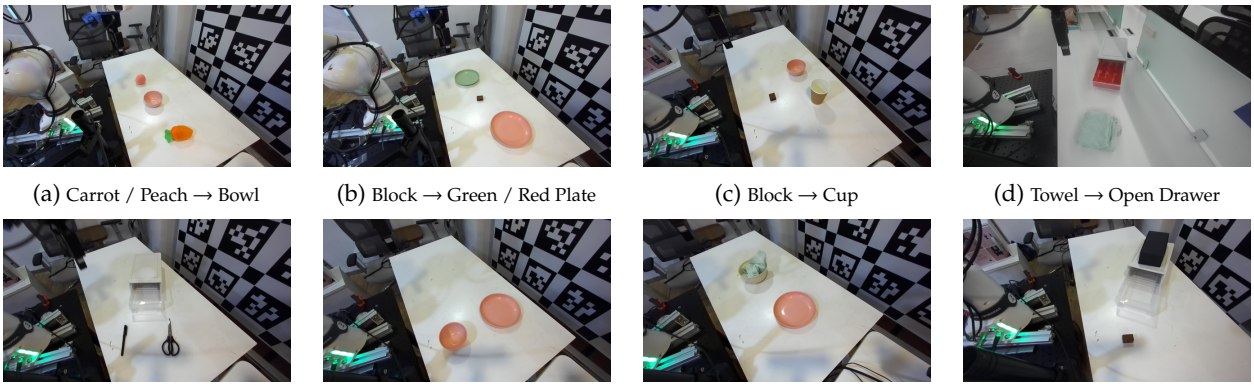
Baselines. We compare **G0.5** against two representative baselines: $\pi_{0.5}$-DROID [4], trained on the original DROID dataset with a PaliGemma backbone, and MolmoAct2-DROID [44], a generalist policy built on the Molmo vision-language model and trained with the MolmoAct2 data preprocessing pipeline.

5.1.2 Quantitative Results

Fig. 6 presents the per-task success rates across all three models. Overall, **G0.5** consistently outperforms both baselines across the majority of tasks, achieving an average success rate of 82.5%.

G0.5-DROID vs. $\pi_{0.5}$-DROID. **G0.5**-DROID outperforms $\pi_{0.5}$-DROID on all 10 tasks, with particularly strong advantages on tasks that demand precise object discrimination and multi-step reasoning. On tasks where objects share similar appearance or require colour-conditioned target selection, **G0.5**-DROID demonstrates significantly stronger visual grounding.

G0.5-DROID vs. MolmoAct2-DROID. **G0.5**-DROID shows especially large margins on tasks involving spatial language instructions and object-category recognition, where MolmoAct2 struggles to ground instruction semantics into correct motor behaviour. Most notably, MolmoAct2-DROID completely fails on



(a) Carrot / Peach → Bowl (b) Block → Green / Red Plate (c) Block → Cup (d) Towel → Open Drawer

(e) Pen → Open Drawer (f) Bowl → Left (g) Towel: Bowl → Plate (h) Block → Drawer + Close

Figure 5: **DROID environment- and object-level zero-shot evaluation tasks.** After DROID post-training, we evaluate **G0.5** on 10 manipulation tasks across 8 held-out scene setups and previously unseen physical object instances on a Franka Research 3 robot arm. Tasks cover object placement, color-conditioned target selection, small-aperture insertion, deformable object manipulation, spatial displacement, and multi-step sequential execution.

the sequential task *put the block into the open drawer and close the drawer*, while **G0.5**-DROID succeeds on over half of the trials, demonstrating substantially stronger multi-stage task execution capability. We further observe that MolmoAct2-DROID frequently freezes or produces no motion when approaching objects such as the carrot, peach, or bowl, and often executes empty grasps when the gripper has not yet reached a valid pre-grasp pose.

Effect of visual contrast on drawer localisation. The drawer cabinet used in this evaluation features a white semi-transparent body, which provides limited visual contrast for localising the insertion aperture. To examine the impact of this visual ambiguity, we conducted a controlled comparison on the towel insertion task: in the main evaluation, orange adhesive cards were attached to the drawer's interior walls and base as explicit localisation markers, whereas the earlier experiment was run without any markers ($\pi_{0.5}$-DROID: 90%, MolmoAct2-DROID: 80%, **G0.5**-DROID: 60%). Adding the high-contrast markers dramatically improves **G0.5**-DROID's performance to 100%, while $\pi_{0.5}$-DROID remains comparatively unaffected. This indicates that **G0.5**-DROID is more sensitive to low-contrast semi-transparent surfaces and is relatively less capable of reliably localising targets without explicit high-contrast visual cues.

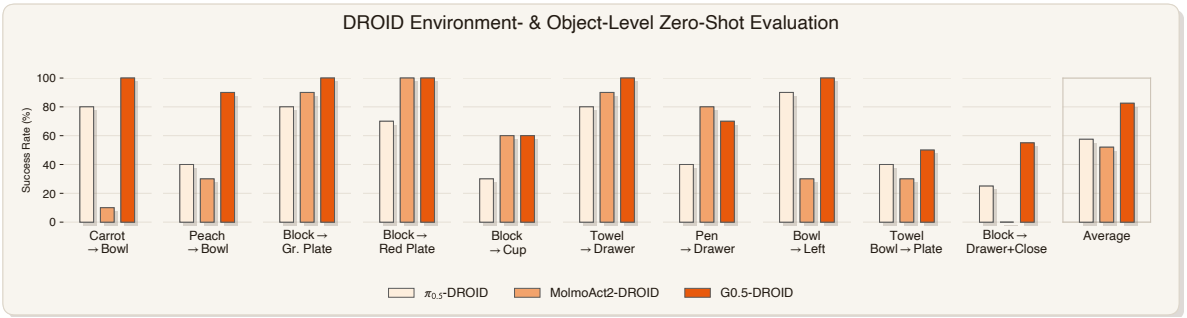


Figure 6: **DROID environment- and object-level zero-shot evaluation results.** All models are trained on DROID data, while the physical evaluation environment and object instances are held out. Per-task success rates (%) are reported for $\pi_{0.5}$-DROID, MolmoAct2-DROID, and **G0.5** across 10 manipulation tasks. **G0.5** achieves an average of 82.5%, outperforming $\pi_{0.5}$-DROID (57.5%) by 25.0 percentage points and MolmoAct2-DROID (52.0%) by 30.5 percentage points.

Table 1: Results on Bridge-SimplerEnv. All numbers are success rates (%). Since several original model papers do not report results on SimplerEnv-WidowX, we compile the corresponding baseline numbers from prior studies that evaluate these models on this benchmark. The best and second-best results are highlighted in bold and underlined, respectively.

Method	Put Spoon on Towel	Put Carrot on Plate	Stack Green Block on Yellow Block	Put Eggplant in Yellow Basket	**Average**
π_0 [3]	29.1	0.0	16.6	62.5	27.1
π_0-FAST [11]	29.1	21.9	10.8	66.6	32.1
$\pi_{0.5}$ [4]	49.3	<u>64.7</u>	44.7	69.7	57.1
GR00T-N1.5 [5]	75.3	54.3	57.0	61.3	61.9
StarVLA-GR00T [46]	83.0	59.4	18.8	**100.0**	65.3
RoboBrain2.5-8B [42]	75.0	55.5	40.1	**100.0**	67.6
MemoryVLA [47]	75.0	**75.0**	37.5	**100.0**	71.9
EO-1 [48]	63.6	54.5	<u>81.8</u>	90.9	72.7
Xiaomi-Robotics-0 [49]	<u>95.8</u>	62.5	75.0	83.3	<u>79.2</u>
G0.5 (Ours)	**97.5**	**75.0**	**83.3**	<u>93.3</u>	**87.3**

5.2 Simulation Benchmarks

5.2.1 Bridge-SimplerEnv

Bridge-SimplerEnv evaluates language-conditioned WidowX manipulation policies in SimplerEnv, a real-to-sim benchmark that instantiates the BridgeData V2/WidowX setup for scalable simulated policy evaluation [15]. We follow the official SimplerEnv WidowX+Bridge task suite and evaluate on four Bridge-style manipulation tasks: putting a spoon on a towel, putting a carrot on a plate, stacking a green cube on a yellow cube, and putting an eggplant into a yellow basket. To adapt our policy to the WidowX/Bridge embodiment before evaluation, we post-train it on BridgeData V2 demonstrations [45] for 80K gradient steps with a learning rate of 3×10^{-5}. Throughout both post-training and evaluation, we adopt a state-free policy input, where robot joint states or other proprioceptive states are not provided to the model. This yields a stricter yet comparable evaluation protocol under the same SimplerEnv task suite. We then evaluate the resulting policy in SimplerEnv without any additional simulation-domain training. Results are summarized in Tab. 1, where **G0.5** achieves the highest average success rate of 87.3% among the compared methods.

5.2.2 RoboTwin 2.0

RoboTwin 2.0 evaluates simulated bimanual manipulation across a broad suite of over 50 tasks, emphasizing behaviors that depend on coordinated dual-arm control rather than single-arm pick-and-place skills alone. We follow the multi-task training setup of [50, 51]: models are trained on a combined set of 2,500 clean-scene demonstrations and 25,000 demonstrations collected with heavy scene randomization. We finetune **G0.5** for 4 epochs using a learning rate of 4×10^{-5} and a global batch size of 1024. Success rates are averaged over 100 trials per task in both clean and randomized evaluation settings. Results are reported in Tab. 2. Per-task success rates for **G0.5** are provided in Tab. 7.

5.2.3 LIBERO

LIBERO is a Franka robot arm simulation benchmark comprising four task suites—Goal, Spatial, Object, and Long—which evaluate instruction following, spatial reasoning, object recognition, and long-horizon manipulation, respectively. Each suite contains 10 tasks with 50 demonstrations per task. We finetune **G0.5** for 100K steps using a learning rate of 1×10^{-5} and a weight decay of 1×10^{-2}. Following the standard LIBERO protocol, we evaluate the model over 50 rollout trials per task using the benchmark-defined initial states. Results are summarized in Tab. 3. **G0.5** achieves state-of-the-art overall performance among recent

Table 2: Results on RoboTwin 2.0. We report success rates under clean and randomized evaluation settings, together with their average. The best and second-best results are highlighted in bold and underlined, respectively.

Method	Clean	Rand.	**Average**
π_0 [3]	65.9	58.4	62.2
$\pi_{0.5}$ [4]	82.7	76.8	79.8
LingBot-VLA [52]	86.5	85.3	85.9
Motus [50]	88.7	87.0	87.8
StarVLA [46]	88.7	87.8	88.3
LingBot-VA [51]	<u>92.9</u>	91.5	<u>92.2</u>
Fast-WAM [16]	91.9	<u>91.8</u>	91.8
G0.5 (Ours)	**93.7**	**92.8**	**93.3**

Table 3: Results on LIBERO. The best and second-best results are highlighted in bold and underlined, respectively.

Method	Spatial	Object	Goal	Long	**Average**
Qwen-VLA-Base [53]	–	–	–	–	90.8
π_0 [3]	98.0	96.8	94.4	88.4	94.4
InternVLA-M1 [54]	98.0	99.0	93.8	92.6	95.9
Wall-OSS-0.5 [55]	–	–	–	–	96.5
$\pi_{0.5}$ [4]	98.8	98.2	98.0	92.4	96.9
GR00T-N1.7 [5]	97.7	98.5	97.5	94.4	97.0
OpenVLA-OFT [56]	97.6	98.4	97.9	94.5	97.1
Fast-WAM [16]	98.2	**100.0**	97.0	95.2	97.6
Being-H0.5 [23]	<u>99.2</u>	98.2	<u>99.0</u>	96.2	98.2
Motus [50]	96.8	<u>99.8</u>	96.6	97.6	97.7
Qwen-VLA-Instruct [53]	–	–	–	–	97.9
EO1 [48]	**99.7**	<u>99.8</u>	**99.2**	94.8	98.4
Cosmos Policy [57]	98.1	**100.0**	98.2	97.6	98.5
LingBot-VA [51]	98.5	99.6	97.2	<u>98.5</u>	98.5
Xiaomi-Robotics-0 [49]	98.8	**100.0**	98.8	97.2	<u>98.7</u>
G0.5 (Ours)	98.4	**100.0**	98.6	**98.6**	**98.9**

VLA models, attaining an average success rate of 98.9%. Notably, it delivers the strongest performance on the challenging Long suite.

5.3 Long-horizon Tasks

The 2025 BEHAVIOR Challenge, built on the BEHAVIOR-1K benchmark [13] and the photo-realistic OmniGibson simulator powered by NVIDIA Isaac Sim, presents a demanding testbed for **long-horizon mobile manipulation**. The challenge selects 50 full-length household tasks from the 1,000 activity collection, covering diverse activities like rearrangement, cooking, cleaning, and installation. To support training, it provides 10,000 teleoperated expert demonstrations totaling over 1,100 hours, where each demonstration episode averages 6.6 minutes and spans up to 14 minutes. To accomplish these tasks, a policy must control an **R1-Pro** robot to simultaneously process RGB observations from the head and dual-wrist cameras, navigate through house-scale environments, and perform dexterous bimanual manipulation using two 7-DOF arms equipped with parallel-jaw grippers.

During evaluation, policies are tested over 10 episodes per task with systematically randomized initial object states and robot poses. Because executing these complex household chores is significantly more demanding than short horizon table top tasks, overall performance is quantified by a Task Success Score, which serves as the primary ranking metric for the challenge. This metric measures how much of a goal condition a policy satisfies by computing the proportion of completed BDDL goal predicates and selecting the best matched

goal clause at the end of the episode. By awarding partial credit, it ensures that policies making meaningful progress score higher even without full task completion. Consequently, this granular scoring mechanism provides a smoother and more reliable way to evaluate incremental progress and compare policies across BEHAVIOR tasks than a traditional binary success rate.

5.3.1 Implementation Details

For our evaluation, we adopt the official standard track and the default low-resolution RGB rendering setting. To ensure a fair comparison, we follow the first place solution [58] and use single frame observations during post training, but omit their explicit stage head. Notably, to validate the general mobile manipulation capabilities of our pre-trained model and to measure the comprehensive performance of a single policy across 50 diverse household tasks, we jointly co-train all 10,000 episodes from the 50 tasks during the post-training phase. Consequently, the test results reported below for both $\pi_{0.5}$ and **G0.5** are evaluated using only a single checkpoint.

Table 4: Overall results on the 2025 BEHAVIOR Challenge (50 tasks, 10 instances each). *Task Success Score* is the challenge ranking metric (task progress). The first place solution by the Robot Learning Collective (RLC) [58] uses a set of 4 checkpoints; $\pi_{0.5}$ [4] (4 epochs) and **G0.5** each use a single checkpoint, averaged over two eval runs. Best/second best in **bold**/underline.

Method	Task Success Score (Ranking Metric) ↑
RLC (1st place) [58]	0.2605
Comet (2nd place) [59]	0.1830
$\pi_{0.5}$ (4 epochs) [4]	0.2626
G0.5 (Ours, 1 epoch)	<u>0.2904</u>
G0.5 (Ours, 4 epochs)	**0.3136**

5.3.2 Key Findings and Analysis

We evaluate **G0.5** and $\pi_{0.5}$ directly to establish a strictly fair comparison of the pre-trained model weights using a single policy, whereas other baseline scores represent public leaderboard submissions utilizing multiple policies. From Table 4 and the detailed results in Table 6, we highlight the following key findings:

- **Training Efficiency.** With only a single epoch of post-training, **G0.5** already surpasses $\pi_{0.5}$ trained for four epochs by +10.6% in the primary Task Success Score. With four epochs, this advantage widens to +19.4%, demonstrating that our model continues to improve with additional training. This result directly demonstrates the superior representational capacity of the **G0.5** pre-trained backbone: a stronger foundation model can extract task-relevant knowledge from the same downstream data far more efficiently, requiring significantly fewer gradient steps to acquire complex household manipulation skills.
- **Single-Policy Generalization.** Across the entire suite of 50 tasks, **G0.5** (4 epochs) outperforms the first-place solution by +20.4% using only a single checkpoint, whereas the competition winner relies on a set of four distinct checkpoints to cover different task distributions. Even with just 1 epoch, **G0.5** already exceeds the first-place result by +11.5%. This confirms that **G0.5** learns a sufficiently general whole-body control prior during pre-training, eliminating the need for task-specific checkpoint selection at evaluation time.

We attribute these results primarily to three aspects of **G0.5**'s design:

Structured Action Decomposition Benefits Mobile Manipulation. As described in Section 3.1, our structured tokenization decomposes the robot's action space into independent motion parts (e.g., left control, right control, lower body). This decomposition is particularly beneficial for mobile manipulation tasks, as it explicitly decouples navigation from manipulation in the token space. Rather than learning from a flat, entangled action representation, the model acquires a factored whole-body control prior where each motion

group can be independently predicted. This advantage is clearly reflected in our results: **G0.5** achieves strong performance on long-horizon tasks that interleave navigation and object manipulation, such as *moving boxes to storage* (+0.35 vs. $\pi_{0.5}$), *picking up trash* (+0.30), and *loading the car* (+0.28), where the robot must navigate to different locations, grasp objects, and place them at target positions in sequence.

Pre-Training Distribution Shapes Downstream Strengths. Our per-task analysis reveals a clear alignment between pre-training data composition and downstream task performance. As shown in Figure 4, the real-robot pre-training data for **G0.5** is predominantly composed of pick-and-place behaviors. Correspondingly, **G0.5** demonstrates strong advantages on *open-space pick-and-place* tasks, where the robot picks up objects and places them at target locations across diverse spatial configurations. Examples include *setting mousetraps* (+0.46), *assembling gift baskets* (+0.26), and *putting shoes on rack* (+0.20), all of which primarily require robust grasping, accurate placement, and coordinated navigation across diverse spatial configurations.

Conversely, $\pi_{0.5}$ outperforms **G0.5** on *container-interaction* tasks that involve appliance or cabinet manipulation (e.g., *make microwave popcorn*: 0.95 vs. 0.55; *cook hot dogs*: 0.93 vs. 0.90). These skills are severely underrepresented in our pre-training data; however, the gap narrows substantially with more training: *cook hot dogs* improves from 0.45 (1 epoch) to 0.90 (4 epochs), approaching $\pi_{0.5}$'s 0.93.

Despite this distributional gap, **G0.5** leads on 29 out of 50 evaluated household tasks (58%) while $\pi_{0.5}$ leads on only 15 (30%), with 6 tasks being comparable. This broad coverage underscores the generality of the pre-trained representations. Moreover, it suggests a clear path forward: enriching the pre-training data with container-interaction skills could further narrow the remaining gap.

Visual Memory Pre-Training Improves Long-Horizon Performance. As described in Section 3.3, **G0.5** is pre-trained with factorized spatial-temporal attention that processes multi-frame visual context. Although we use single-frame input during post-training for fair comparison, the benefits of temporal pre-training are clearly evident in the downstream results. The gains are most pronounced on navigation-intensive, long-horizon tasks: *moving boxes to storage*, *loading the car*, *bringing in wood*, and *tidying bedroom*. These tasks require the robot to repeatedly traverse between distant locations while tracking which objects have been moved and where they were placed.

We attribute this advantage to the visual dynamics inherent in mobile manipulation pre-training data: even in standard pick-and-place episodes, the robot frequently moves its base between grasp and place locations, causing consecutive frames from each camera to exhibit large visual changes including scene layout shifts and object appearance transitions. The factorized temporal attention in the vision encoder, operating within each camera view across time steps is well-suited to capture these sequential visual dynamics, encouraging each per-camera representation to encode not just the current observation but also an implicit understanding of how the scene evolves over time. Additionally, during pre-training we stochastically drop all historical frames with 30% probability to prevent the model from overfitting to historical context. Together, these design choices yield single-frame representations that are more spatially informed, which explains why **G0.5** generalizes well to long-horizon mobile manipulation even when post-trained with single-frame input.

5.4 Real-World Fine-Tuning Evaluation

We evaluate **G0.5** through real-world fine-tuning experiments on two robot embodiments, R1-Lite and R1-Pro. This evaluation focuses on whether a policy can be adapted to robot embodiments with different kinematic structures and execute long-horizon bimanual manipulation tasks under matched training and evaluation conditions.

Robot Embodiments. R1-Lite is a mobile dual-arm manipulation platform, comprising two 6-DoF arms, a 3-DoF torso, and a mobile omnidirectional base. The torso provides vertical and pitching motion to extend the manipulation workspace and improve operational flexibility.

R1-Pro is a humanoid upper-body mobile manipulation platform, comprising two 7-DoF arms, a 4-DoF torso, and a mobile omnidirectional base. Compared with R1-Lite, the additional arm and torso degrees of freedom

provide enhanced dexterity and greater flexibility for complex bimanual manipulation tasks. Evaluating on both embodiments allows us to test whether the learned policy remains effective under different workspace, dexterity, and whole-body coordination requirements.

Baselines and Fine-Tuning Protocol. We compare **G0.5** with two representative open-source VLA baselines, $\pi_{0.5}$ [4] and GR00T-N1.7 [5]. For each evaluation setting, all models are fine-tuned on the same training data with an aligned compute budget. Specifically, each model is trained using 16 H20 GPUs for the same wall-clock duration within the same setting, ranging from 4 to 10 hours depending on task complexity. All models are adapted to both robot embodiments and evaluated with the same observation space, action space, inference procedure, and low-level control settings.

Evaluation Protocol. We use *task* to denote the semantic task objective, such as towel folding or carton folding, and *setting* to denote a specific task-embodiment pair, such as towel folding on R1-Lite or towel folding on R1-Pro. Under this definition, the real-world fine-tuning evaluation contains four tasks instantiated as six evaluation settings. The R1-Lite settings include towel folding, carton folding, and pencil-case packing, while the R1-Pro settings include towel folding, carton folding, and box transfer and stacking. Towel folding and carton folding are evaluated on both embodiments, enabling direct comparison across different robot configurations under the same task objectives.

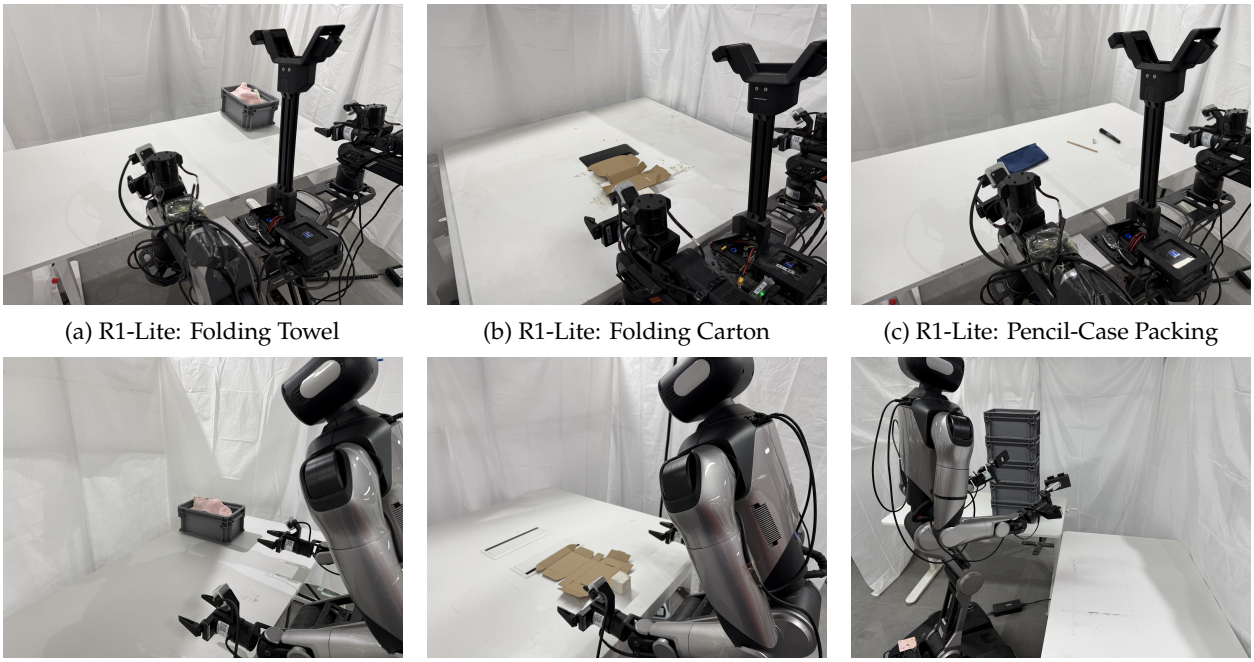
Each setting is evaluated over 15 real-world episodes. We report both task success rate and process score. The success rate measures the fraction of episodes in which the full task is completed, while the process score evaluates intermediate task progress based on predefined stage-wise criteria. The detailed stage definitions and scoring rules are provided in the appendix. To reduce the influence of uncontrolled environmental factors, such as lighting changes and robot hardware state variations, we evaluate the models in an interleaved order within each setting rather than evaluating one model exhaustively before the next.

Observation Setup. Both platforms are equipped with three RGB cameras for visual observation. Two wrist-mounted cameras provide close-up views for fine-grained gripper manipulation, while one external camera provides a global view of the scene for spatial understanding and long-horizon planning.

5.4.1 Task Definitions and Evaluation Metrics

Fig. 7 shows the six real-world evaluation settings across the two robot embodiments. The four tasks cover deformable object manipulation, contact-rich assembly, sequential object interaction, and whole-body bimanual coordination.

- **Folding Towel.** This task is evaluated on both R1-Lite and R1-Pro. The robot is required to 1) grasp a towel from a basket, 2) lift and unfold the towel through coordinated bimanual motion, 3) flatten the towel on the tabletop, 4) fold the towel into a predefined configuration, and 5) place the folded towel into a designated target area. This task is challenging because towels exhibit highly deformable and unstable geometric states during manipulation. Small errors in grasping or tension control can accumulate throughout the folding process, leading to misalignment, incomplete folds, or entanglement. Successful execution therefore requires accurate dual-arm coordination, continuous shape regulation, and long-horizon manipulation of deformable objects.
- **Folding Carton.** This task is evaluated on both R1-Lite and R1-Pro. The robot is required to transform a flat carton into a complete box structure through a sequence of predefined folding operations. The task involves multiple stages of coordinated bimanual interaction, including edge alignment, surface folding, and structure stabilization. Since the carton is non-rigid and sensitive to manipulation errors, minor inaccuracies during intermediate folding stages may damage the structure or prevent successful assembly. The task therefore demands precise dual-arm coordination, accurate contact control, and stable sequential execution.
- **Box Transfer and Stacking.** This task is evaluated on R1-Pro. The robot is required to sequentially transfer five boxes from one table to another and stack them into a stable configuration. During placement, each



(a) R1-Lite: Folding Towel (b) R1-Lite: Folding Carton (c) R1-Lite: Pencil-Case Packing

(d) R1-Pro: Folding Towel (e) R1-Pro: Folding Carton (f) R1-Pro: Box Transfer and Stacking

Figure 7: **Overview of real-world fine-tuning evaluation settings.** We evaluate four manipulation tasks instantiated as six task-embodiment settings across R1-Pro and R1-Lite. The R1-Pro settings include towel folding, carton folding, and box transfer and stacking, while the R1-Lite settings include towel folding, carton folding, and pencil-case packing. Towel folding and carton folding are shared across both embodiments, enabling cross-embodiment comparison under the same task objectives.

box must be accurately aligned with the grooves of the box below. Unlike tabletop-only manipulation tasks, this task requires coordinated control of the upper body, including both arms and the torso, to achieve sufficient reachability and stable motion during transportation and placement. The main challenge lies in precise spatial alignment during stacking, as small positioning errors can cause instability or collapse of the stack.

- **Pencil-Case Packing.** This task is evaluated on R1-Lite. The robot is required to 1) unzip a pencil case from its closed state, 2) identify and sequentially place designated stationery items into the pencil case, and 3) close the zipper after all target objects have been inserted. This task combines deformable object manipulation with fine-grained tool interaction. The zipper is small and requires accurate manipulation to operate reliably, while the deformable structure of the pencil case introduces additional geometric uncertainty during interaction. In addition, the robot must identify target objects and perform sequential pick-and-place operations under cluttered tabletop conditions.

For all tasks, we report both task success rate and process score. The process score evaluates intermediate task progress based on predefined stage-wise completion criteria, enabling finer-grained comparison between different policies when a task is only partially completed.

5.4.2 Quantitative Results

Fig. 8 summarizes the quantitative results across the six real-world evaluation settings. Overall, **G0.5** achieves the highest average performance among the three models. Across all six settings, **G0.5** obtains an average success rate of 76.7%, compared with 53.3% for $\pi_{0.5}$ and 24.4% for GR00T-N1.7. **G0.5** also achieves an average process score of 129.2, compared with 105.2 for $\pi_{0.5}$ and 68.9 for GR00T-N1.7.

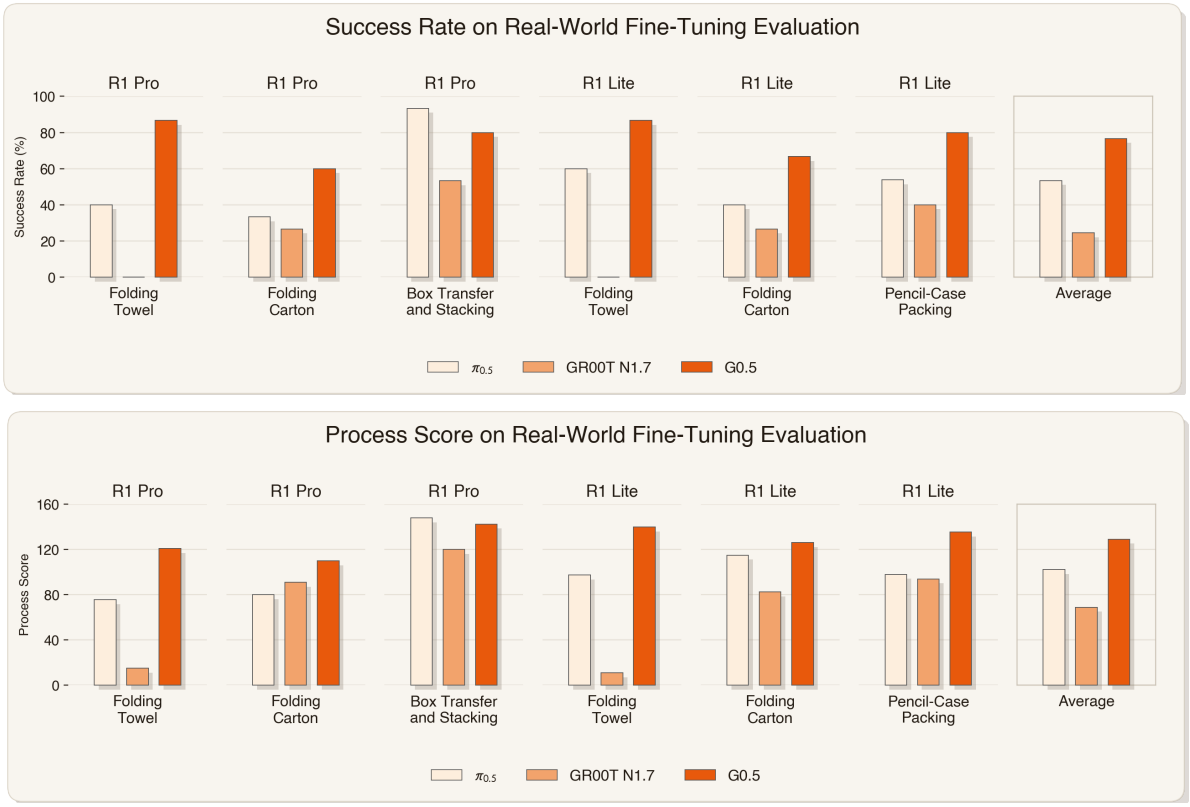


Figure 8: **Performance on real-world fine-tuning evaluation.** We evaluate **G0.5** against $\pi_{0.5}$ and GR00T-N1.7 on four manipulation tasks instantiated as six task-embodiment settings across R1-Pro and R1-Lite. **G0.5** achieves strong overall performance in both task success rate and process score, demonstrating robust long-horizon manipulation capability across different embodiments.

G0.5 achieves the highest success rate in five out of the six settings. The only exception is the R1-Pro box transfer and stacking setting, where $\pi_{0.5}$ achieves a higher final success rate of 93.3%, while **G0.5** achieves 80.0%. However, **G0.5** remains competitive in this setting, achieving a process score of 142.5 compared with 148.0 for $\pi_{0.5}$, and outperforming GR00T-N1.7 in both success rate and process score.

The two shared tasks, towel folding and carton folding, allow direct comparison across the two robot embodiments. On these four shared task-embodiment settings, **G0.5** achieves an average success rate of 75.0%, outperforming $\pi_{0.5}$ at 43.3% and GR00T-N1.7 at 13.3%. For process score on the same shared settings, **G0.5** achieves an average score of 124.3, compared with 92.0 for $\pi_{0.5}$ and 49.9 for GR00T-N1.7. These results indicate that **G0.5** maintains strong performance not only on embodiment-specific tasks, but also on the same task objectives executed by different robot configurations.

The performance of **G0.5** is also balanced across embodiments. On R1-Pro, **G0.5** achieves an average success rate of 75.6% and an average process score of 124.5. On R1-Lite, **G0.5** achieves an average success rate of 77.8% and an average process score of 133.8. This suggests that **G0.5** adapts effectively to both the 6-DoF dual-arm embodiment of R1-Lite and the 7-DoF humanoid upper-body embodiment of R1-Pro under the same fine-tuning and evaluation protocol.

5.5 Pick-and-Place Benchmark

Large-scale pretraining is expected to improve not only low-level action generation, but also language following in visually cluttered scenes. In real-world manipulation, a policy must first identify the object and target specified by the instruction before executing primitive skills such as picking and placing. This

distinction is critical because failures may arise either from incorrect language following, where the robot interacts with the wrong object, or from low-level execution errors after the correct target has been selected. To disentangle these factors, we introduce the Pick-and-Place Benchmark (PP Bench), which separately reports language following rate and final task success rate.

Dataset and Post-training Setup. We collect 50 hours of tabletop manipulation data using the R1-Lite robot. Each scene contains 5–20 objects randomly placed on the table with varying positions and orientations. The robot is instructed to pick up a specified object and place it into a specified container. Each instruction explicitly specifies both the target object and the target container, following the same format used during pretraining, e.g., *"Task: Pick up the yellow utility knife with the left hand and place it into the white basket."*

To evaluate the effect of post-training scale, we construct three nested subsets from the 50-hour dataset: 1H, 10H, and 50H, where the 1H subset is sampled from the 10H subset and the 10H subset is sampled from the full dataset. The models are trained for 12, 8, and 4 epochs on the 1H, 10H, and 50H subsets, respectively. This setup allows us to compare different data scales while keeping the data distribution consistent across subsets.

Evaluation Setting. The test set includes both in-distribution and out-of-distribution object categories. We randomly sample 48 objects from the 50H subset and additionally include 16 categories absent from all post-training data for out-of-distribution evaluation. The full set of benchmark objects and containers is shown in Fig. 9. During evaluation, each tabletop scene contains 16 randomly placed objects and containers, following the same setup as data collection. In each trial, the robot receives a language instruction and is required to place the specified object into the specified container. Each model is evaluated over 64 real-world trials to reduce variance. For fair comparison, different models are evaluated on the same instruction under identical object layouts, container placements, and robot initial states. This paired evaluation protocol reduces variance from scene configuration and isolates the effect of the policy.

We report two metrics: language following rate and task success rate. Language following rate measures whether the robot selects the object specified by the instruction among distractors. A trial is counted as language-following success if the robot moves toward the target object and attempts to grasp it. Task success rate measures whether the robot completes the full instruction by successfully grasping the specified object and placing it into the specified container.

Results and Analysis. Using PP Bench, we evaluate **G0.5** from four perspectives: zero-shot capability, the effect of post-training, comparison with $\pi_{0.5}$, and the benefit of additional context (Fig. 10).

Zero-shot capability. **G0.5** exhibits strong zero-shot language following. Without any PP-specific post-training data, it achieves a language following rate of 65.6% and a task success rate of 59.4%. This indicates that large-scale pretraining provides transferable instruction-following ability and basic pick-and-place action priors.

Effect of post-training. Post-training further improves both semantic grounding and execution reliability. **G0.5** reaches language following rates of 62.5%, 71.9%, and 84.4% under the 1H, 10H, and 50H settings, respectively, with corresponding task success rates of 57.8%, 65.6%, and 75.0%. The consistent improvement from 10H to 50H suggests that additional post-training data strengthens grounding for in-distribution objects and improves action reliability on the R1-Lite embodiment.

Comparison with $\pi_{0.5}$. Compared with $\pi_{0.5}$, **G0.5** achieves higher language following and task success across all post-training scales. The gap is most pronounced in the zero-shot and 1H settings, where $\pi_{0.5}$ shows limited transfer to the R1-Lite setup. With 50H post-training, $\pi_{0.5}$ improves to 68.8% in language following and 65.6% in task success, indicating that target-domain post-training helps adapt the model to the instruction format, object-container distribution, and R1-Lite action interface. Nevertheless, under the same 50H setting, **G0.5** still outperforms $\pi_{0.5}$ by 15.6 percentage points in language following and 9.4 percentage points in task success. We attribute this advantage to large-scale web-data co-training, which improves open-vocabulary



Figure 9: **Pick-and-Place Benchmark setting.** The evaluation set contains 64 object categories and 3 container categories. Each trial presents 16 randomly arranged objects and containers, requiring the robot to identify the instructed target among distractors and place it into the specified container. Each policy is evaluated across all object categories in real-world trials to reduce variance.

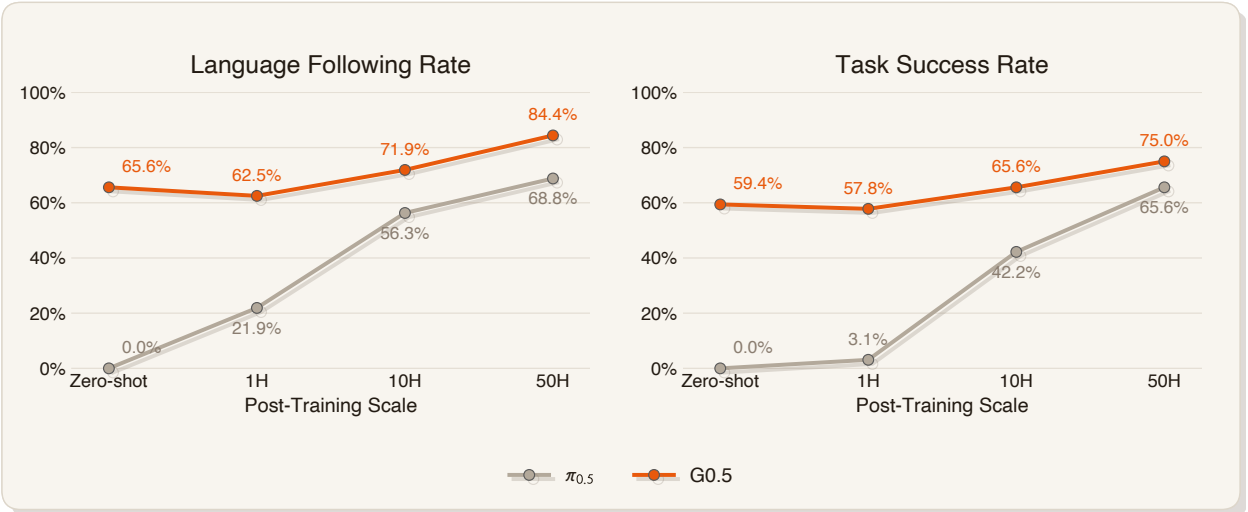


Figure 10: **PP Bench evaluation results.** Left: language following rate. Right: task success rate. We compare **G0.5** and $\pi_{0.5}$ across zero-shot, 1H, 10H, and 50H post-training settings.

Table 5: **Ablation on referring context for PP Bench.** We evaluate **G0.5** with progressively richer *referring context*, i.e., auxiliary inputs appended to the standard instruction that help the policy identify the language-specified target object beyond its category name.

Metric	Name-only	+ Box Coord.	+ Target Visual
Language Following Rate	84.4%	85.9%	98.4%
Task Success Rate	75.0%	76.6%	84.4%

semantic understanding, and to robot pretraining that includes the R1-Lite embodiment, which provides stronger action priors and execution quality.

Additional referring context. Failure analysis shows that many remaining language-following errors occur on long-tail objects that are visually ambiguous, partially occluded, or difficult to identify from language alone. We therefore evaluate several referring-context variants in Table 5. *Name-only* uses the standard instruction format, where the prompt specifies only the target object and container names. *+ Box Coord.* augments the instruction with textual coordinate tokens for the target object and container boxes. Despite providing explicit spatial cues, this setting does not improve over the name-only baseline. We hypothesize that directly injecting coordinate tokens introduces an instruction-format shift from the name-only robot pretraining distribution, and that the post-training data may be insufficient for the action head to reliably exploit this new conditioning signal. *+ Target Visual* further augments the input with cropped visual regions of the target object and container. These visual contexts are appended to the original camera views and encoded with the same visual encoder as the standard observations. Compared with box coordinates alone, the target visual context provides fine-grained local appearance cues, such as texture, shape, and category-specific visual details. These cues are particularly useful for targets that are difficult to disambiguate from language alone or belong to long-tail categories with limited linguistic exposure but distinctive visual appearance, such as a Chinese chess piece labeled "horse". This setting significantly improves language following to 98.4% and task success to 84.4%.

5.6 Zero-Shot Probe of CoT and Action Head

We isolate the contributions of chain-of-thought and the action decoder by taking a single pretrained **G0.5** checkpoint and varying *only the inference-time configuration*; no parameters are fine-tuned or adapted, and the same weights are reused across all cells. The probe spans three tasks of increasing horizon on the R1-Lite platform: PP Bench (single stage), and two new long-horizon zero-shot tasks—*Air Fryer* and *Cook Bacon*—each decomposed into five sequential stages (*Air Fryer:* approach → open door → grasp bread → place inside → close door; *Bacon:* approach → grasp bacon → place in pan → turn on stove → flip). At runtime the policy receives a per-stage natural-language instruction for the current sub-goal. We toggle (i) the decoder—*AR* for autoregressive action tokens versus *FM* for an additional flow-matching head (optional at inference)—and (ii) the chain-of-thought stream, where the model optionally emits subtask and bounding-box reasoning before each action. When CoT is on, both decoders read from the post-CoT hidden state, so the comparison isolates the decoding interface rather than the conditioning input. We report the language-following rate and, for the long-horizon tasks, a progress score equal to the number of completed sub-stages (0–5). PP Bench follows the zero-shot evaluation protocol of Sec. 5.5 (64 rollouts); the long-horizon tasks (Fig. 11) use n=5 rollouts per cell.

Finding 1: CoT improves grounding and execution within long-horizon stage-conditioned rollouts. On the single-stage PP Bench, CoT brings essentially no change to language following: AR moves from 65.6 to 67.2 and FM from 59.4 to 60.9 over 64 rollouts—at most ~1.6 percentage points for either decoder. With only one grounding event per rollout, there is little room for per-stage reasoning to help. On the five-stage Air Fryer and Bacon tasks the picture is different: with per-stage sub-goals supplied at runtime, CoT lets the policy ground the relevant object and execute each sub-step more reliably. With this structure the AR head's progress score lifts from 2.4 to 3.8 on Air Fryer and from 1.5 to 3.4 on Bacon, with the language-following rate

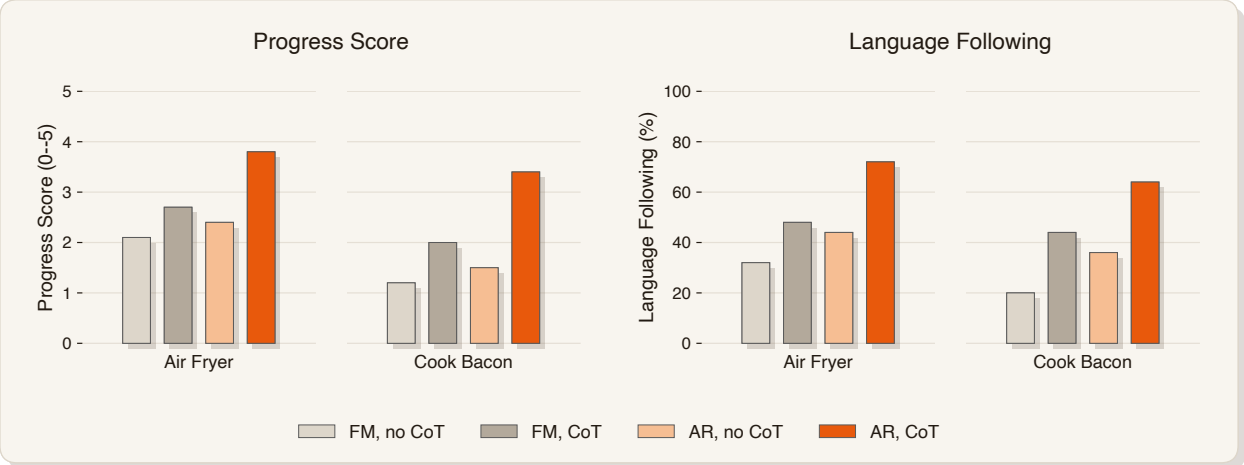


Figure 11: Long-horizon zero-shot probe on *Air Fryer* and *Cook Bacon* (same pretrained **G0.5** checkpoint with only inference-time switching). Progress score (0–5, number of completed sub-stages) and language-following rate across the four decoder × CoT cells: both metrics show the same ordering, with AR+CoT clearly leading on each task.

rising in step (Fig. 11). The benefit thus appears where the task is presented as a sequence of stage-conditioned sub-goals. Air Fryer and Bacon are household manipulation scenes that *do not appear in the pretraining data*; CoT improves per-stage grounding and execution on them without any retraining.

Qualitative observation: instruction wording. Beyond toggling CoT, we informally observed that the exact wording of the per-stage instruction affects behavior on these tasks. Two patterns recurred. First, adding adverbial or spatial qualifiers (e.g., *"push it in hard"*, *"vertically"*) tended to bring the executed motion closer to the intended sub-goal. Second, a single physical action often admits several near-synonymous verbs—closing the air-fryer drawer can be phrased as *press*, *push in*, or *close*—and substituting among them changed which rollouts succeeded in our small set. We report these only as qualitative observations. This probe is AR-only.

Finding 2: AR appears to follow the CoT more closely than the FM head. Under matched CoT, the AR head benefits more than the FM head (Fig. 11): on Air Fryer, CoT lifts AR's progress score from 2.4 to 3.8 while FM moves only from 2.1 to 2.7; on Bacon, 1.5 → 3.4 for AR versus 1.2 → 2.0 for FM. The language-following rate echoes this gap under matched CoT (Air Fryer 72 vs. 48; Bacon 64 vs. 44). We *hypothesize* that this reflects the decoding interface rather than the reasoning content: the autoregressive action tokens are emitted in the same stream as the CoT and can attend to it directly, whereas the FM head conditions on a pooled summary of the hidden state. We do not directly probe this mechanism and leave its verification to future work.

CoT correctness. The CoT is generated autoregressively by the VLA; once it terminates, the action is either continued autoregressively (AR) or sampled by the FM head. AR and FM are therefore run as two separate rollouts whose CoT traces, while produced by the same mechanism, are not identical, since the rollouts diverge once actions are executed. We hand-scored subtask text and bounding-box correctness on the CoT-on rollouts for both heads and found comparable quality—roughly 90% on PP Bench, 85% on Air Fryer, and 80% on Bacon, with no systematic AR–FM difference. The progress-score gap between the two heads under matched CoT therefore does not appear to come from differences in reasoning quality, which is consistent with the decoding-interface hypothesis in Finding 2.

5.7 G0.5 Fine-Tunes with RL Out of the Box

An autoregressive policy emits actions as tokens, so it exposes exact token-level log-probabilities—precisely the quantity that ratio-based RL algorithms consume. A flow-matching head offers no such likelihood

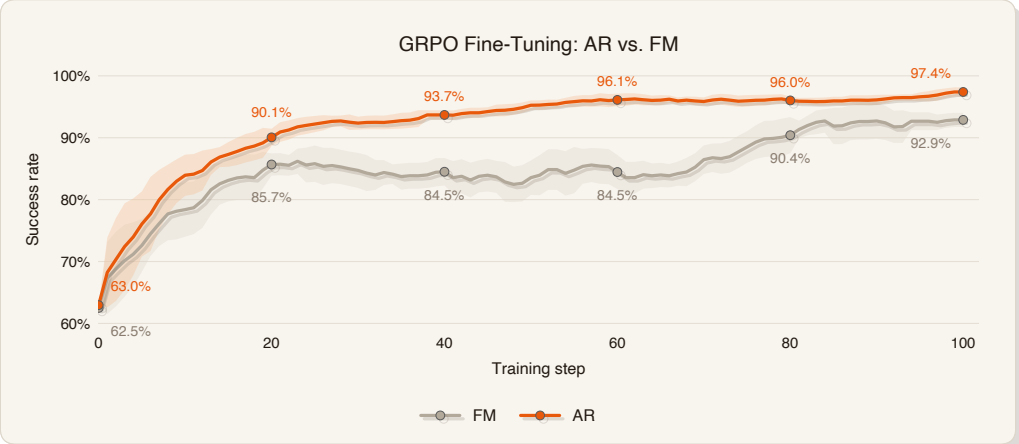


Figure 12: **GRPO fine-tuning: AR vs. FM.** Average success rate over four LIBERO tasks during GRPO fine-tuning from a single demonstration per task (mean ± std over seeds). Starting from AR and FM policies with comparable initial success rates, the AR policy converges faster, attains a higher final success rate, and shows lower run-to-run variance.

and must be reformulated before the same algorithms apply. We therefore ask whether the autoregressive interface is the easier one to optimize with RL, and test this in a low-data regime where post-training gains come almost entirely from exploration rather than imitation. We first jointly post-trained the AR and FM policies using a single demonstration trajectory per task on LIBERO [14]. We then selected four tasks on which the two policies reached comparable initial success rates and applied Group Relative Policy Optimization (GRPO) [60]. For the AR policy, actions were sampled with a temperature to encourage stochastic exploration while retaining tractable token-level log-probabilities. For the FM policy, we introduced an SDE into the denoising procedure and treated the denoising trajectory as a Markov process in order to approximate the policy log-probability, following RLinf [61].

As shown in Fig. 12, the AR policy converges substantially faster, reaches a higher final success rate, and trains more stably, with lower variance, than the FM policy. We hypothesize that this advantage stems from the AR policy's native and direct likelihood parameterization, which makes policy probability ratios straightforward to compute. Applying ratio-based policy optimization to the FM policy, in contrast, requires an auxiliary stochastic reformulation, a discretization of the denoising dynamics, and additional noise-schedule choices; these components may introduce greater optimization sensitivity and gradient variance.

6 Conclusion

We have argued, and empirically supported, that the path forward for VLA models is to let the VLM be what it was pretrained to be—an autoregressive reasoner that now also acts, remembers, and adapts in-context—rather than to design ever more sophisticated action experts on top of an underutilized backbone. **G0.5** instantiates this commitment with a single cross-entropy objective over a shared vocabulary, supported by a cross-embodiment action codec, a native chain-of-thought stream, and a multi-second visual memory module.

Three observations point to a structural rather than incidental advantage of the autoregressive route. First, on Pick-and-Place benchmark, the zero-shot language-following rate of **G0.5** exceeds the post-trained $\pi_{0.5}$ baseline, suggesting that AR action supervision protects—rather than degrades—the VLM's instruction-following ability. Second, on the 2025 BEHAVIOR Challenge, a single **G0.5** checkpoint trained for only one post-training epoch surpasses both $\pi_{0.5}$ post-trained for four epochs and the four-checkpoint winner, indicating that the pretrained representations carry generalist mobile manipulation priors. Third, when fine-tuned under matched compute and an identical protocol on the R1-Lite and R1-Pro platforms, **G0.5**

beats $\pi_{0.5}$ and GR00T-N1.7, indicating that the advantage over VLM-as-encoder architectures persists under apples-to-apples conditions rather than reflecting differences in training budget.

G0.5 inherits two acknowledged failure modes that future work should address. Drawer-insertion and semi-transparent cabinet tasks remain weak across both **G0.5** variants, pointing to a sensing limit not closed by AR alone; and our visual memory captures only seconds of history, leaving long-horizon memory open to research. Lower-body actuation is represented in the unified action space, but is not evaluated separately in this work. More broadly, the prompt-level controllability our zero-shot probe begins to surface—where per-stage instruction wording shifts AR+CoT rollouts on out-of-distribution household tasks—deserves a systematic empirical study of its own. We hope the released pretrained backbone serves as a starting point for further work in these directions.

7 Contributors

Data engineering: Tao Jiang, Ke Dong, Xiaoshu Ren, Chenru Wu, Xiao Liu, Tianyuan Yuan, Zibin Dong, Zihan Guo, Zijie Zhao

Annotation & supplemental data: Tianyuan Yuan, Tao Jiang, Changxun Pan, Xinlei Zhang, Chenru Wu, Haonan Liu, Haodong Yang, Bowen Zhang

Policy training & research: Yicheng Liu, Zibin Dong, Tianyuan Yuan, Baijun Ye, Shicheng Cao, Shaoting Zhu, Xiao Liu

Policy evaluation & benchmarking: Anqi Yang, Zihan Guo, Baijun Ye, Zibin Dong, Tao Jiang, Yue Sun, Tianyuan Yuan, Tailai Cheng, Changxun Pan, Jianning Cui, Shicheng Cao, Haonan Liu, Shaoting Zhu, Zijie Zhao, Haoyu Zhang, Jiahui Niu, Shiduo Zhang

Writing & illustration: Yicheng Liu, Zibin Dong, Baijun Ye, Shicheng Cao, Haonan Liu, Tailai Cheng, Tao Jiang, Zihan Guo, Anqi Yang, Yue Sun, Tianyuan Yuan, Shiduo Zhang, Hang Zhao

Project lead: Yicheng Liu • **Project PI:** Hang Zhao

References

[1] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[2] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[3] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[4] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[5] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[6] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[7] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1702–1713, 2025.

[8] Zhide Zhong, Junfeng Li, Junjie He, Haodong Yan, Xin Gong, Guanyi Zhao, Yingjie Cai, Jiantao Gao, Xu Yan, Bingbing Liu, et al. Dualcot-vla: Visual-linguistic chain of thought via parallel reasoning for vision-language-action models. *arXiv preprint arXiv:2603.22280*, 2026.

[9] Quanxin Shou, Fangqi Zhu, Shawn Chen, Puxin Yan, Zhengyang Yan, Yikun Miao, Xiaoyi Pang, Zicong Hong, Ruikai Shi, Hao Huang, et al. Halo: A unified vision-language-action model for embodied multimodal chain-of-thought reasoning. *arXiv preprint arXiv:2602.21157*, 2026.

[10] Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, et al. Mem: Multi-scale embodied memory for vision language action models. *arXiv preprint arXiv:2603.03596*, 2026.

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

[12] Zhide Zhong, Haodong Yan, Junfeng Li, Xiangchen Liu, Xin Gong, Tianran Zhang, Wenxuan Song, Jiayi Chen, Xinhu Zheng, Hesheng Wang, et al. Flowvla: Visual chain of thought-based motion reasoning for vision-language-action models. *arXiv preprint arXiv:2508.18269*, 2025.

[13] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. BEHAVIOR-1K: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pages 80–93. PMLR, 2023.

[14] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[15] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[16] Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

[17] Danny Driess, Jost Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *Advances in Neural Information Processing Systems*, 38:102867–102888, 2026.

[18] Ankit Goyal, Hugo Hadfield, Xuning Yang, Valts Blukis, and Fabio Ramos. Vla-0: Building state-of-the-art vlas with zero modification. *arXiv preprint arXiv:2510.13054*, 2025.

[19] Yating Wang, Haoyi Zhu, Mingyu Liu, Jiange Yang, Hao-Shu Fang, and Tong He. Vq-vla: Improving vision-language-action models via scaling vector-quantized action tokenizers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11089–11099, 2025.

[20] Hongyi Zhou, Weiran Liao, Xi Huang, Yucheng Tang, Fabian Otto, Xiaogang Jia, Xinkai Jiang, Simon Hilber, Ge Li, Qian Wang, et al. Beast: Efficient tokenization of b-splines encoded action sequences for imitation learning. *Advances in Neural Information Processing Systems*, 38:172934–172959, 2026.

[21] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

[22] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

[23] Hao Luo, Ye Wang, Wanpeng Zhang, Sipeng Zheng, Ziheng Xi, Chaoyi Xu, Haiweng Xu, Haoqi Yuan, Chi Zhang, Yiqing Wang, et al. Being-h0. 5: Scaling human-centric robot learning for cross-embodiment generalization. *arXiv preprint arXiv:2601.12993*, 2026.

[24] I Apanasevich, M Artemyev, R Babakyan, P Fedotova, D Grankin, E Kupryashin, A Misailidi, D Nerus, A Nutalapati, G Sidorov, et al. Green-vla: Staged vision-language-action model for generalist robots. *arXiv preprint arXiv:2602.00919*, 2026.

[25] Shuanghao Bai, Meng Li, Xinyuan Lv, Jiawei Wang, Xinhua Wang, Fei Liao, Chengkai Hou, Langzhe Gu, Wanqi Zhou, Kun Wu, et al. Hex: Humanoid-aligned experts for cross-embodiment whole-body manipulation. *arXiv preprint arXiv:2604.07993*, 2026.

[26] Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Caelan Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, et al. Hamster: Hierarchical action models for open-world robot manipulation. In *International Conference on Learning Representations*, volume 2025, pages 24040–24068, 2025.

[27] Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.

[28] Qi Sun, Pengfei Hong, Tej Deep Pala, Vernon Toh, U-Xuan Tan, Deepanway Ghosal, and Soujanya Poria. Emma-x: An embodied multimodal action model with grounded chain of thought and look-ahead spatial reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14199–14214, 2025.

[29] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. In *International Conference on Learning Representations*, volume 2025, pages 54277–54296, 2025.

[30] Qwen Team. Qwen3. 5: Towards native multimodal agents, february 2026. *URL https://qwen. ai/blog*, 2026.

[31] Suneel Belkhale and Dorsa Sadigh. Minivla: A better vla with a smaller footprint, 2024. URL https://github.com/Stanford-ILIAD/openvla-mini.

[32] Zibin Dong, Yicheng Liu, Shiduo Zhang, Baijun Ye, Yifu Yuan, Fei Ni, Jingjing Gong, Xipeng Qiu, Hang Zhao, Yinchuan Li, and Jianye Hao. Actioncodec: What makes for good action tokenizers. *arXiv preprint arXiv:2602.15397*, 2026.

[33] Yicheng Liu, Shiduo Zhang, Zibin Dong, Baijun Ye, Tianyuan Yuan, Xiaopeng Yu, Linqi Yin, Chenhao Lu, Junhao Shi, Luca Jiang-Tao Yu, Liangtao Zheng, Jingjing Gong, Tao Jiang, Xipeng Qiu, and Hang Zhao. FASTer: Toward powerful and efficient autoregressive vision–language–action models with learnable action tokenizer and block-wise decoding. In *The Fourteenth International Conference on Learning Representations, ICLR*, 2026.

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

[35] Google DeepMind. Gemini 3 pro model card, 2026. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf. Model card, last updated May 2026.

[36] ByteDance Seed Team. Seed 2.0 official launch, 2026. URL https://research.doubao.com/en/blog/seed-2-0-official-launch.

[37] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Radle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollar, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. Sam 3: Segment anything with concepts, 2025. URL https://arxiv.org/abs/2511.16719.

[38] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024. URL https://llava-vl.github.io/blog/2024-01-30-llava-next/.

[39] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer, 2024. URL https://arxiv.org/abs/2408.03326.

[40] 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 for robotics, 2024. URL https://arxiv.org/abs/2406.10721.

[41] Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmoact: Action reasoning models that can reason in space, 2025. URL https://arxiv.org/abs/2508.07917.

[42] Yuheng Ji, Huajie Tan, Jiayu Shi, Xiaoshuai Hao, Yuan Zhang, Hengyuan Zhang, Pengwei Wang, Mengdi Zhao, Yao Mu, Pengju An, Xinda Xue, Qinghang Su, Huaihai Lyu, Xiaolong Zheng, Jiaming Liu, Zhongyuan Wang, and Shanghang Zhang. Robobrain: A unified brain model for robotic manipulation from abstract to concrete, 2025. URL https://arxiv.org/abs/2502.21257.

[43] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. DROID: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[44] Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, Shanli Xing, Jaemin Cho, Jae Sung Park, Ainaz Eftekhar, Peter Sushko, Karen Farley, Angad Wadhwa, Cole Harrison, Winson Han, Ying-Chun Lee, Eli VanderBilt, Rose Hendrix, Suveen Ellawela, Lucas Ngoo, Joyce Chai, Zhongzheng Ren, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmoact2: Action reasoning models for real-world deployment, 2026. URL https://arxiv.org/abs/2605.02881.

[45] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[46] StarVLA Community and Von Neumann Institute, HKUST. Starvla: A lego-like codebase for vision-language-action model developing, 2026.

[47] Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation, 2025.

[48] Delin Qu, Haoming Song, Qizhi Chen, Zhaoqing Chen, Xianqiang Gao, Dong Wang, Xinyi Ye, Qi Lv, Modi Shi, Guanghui Ren, Cheng Ruan, Maoqing Yao, Haoran Yang, Jiacheng Bao, Bin Zhao, and Xuelong Li. Eo-1: An open unified embodied foundation model for general robot control. *arXiv preprint arXiv:2508.21112*, 2026.

[49] Rui Cai, Jun Guo, Xinze He, Piaopiao Jin, Jie Li, Bingxuan Lin, Futeng Liu, Wei Liu, Fei Ma, Kun Ma, Feng Qiu, Heng Qu, Yifei Su, Qiao Sun, Dong Wang, Donghao Wang, Yunhong Wang, Rujie Wu, Diyun Xiang, Yu Yang, Hangjun Ye, Yuan Zhang, and Quanyun Zhou. Xiaomi-robotics-0: An open-sourced vision-language-action model with real-time execution. *arXiv preprint arXiv:2602.12684*, 2026.

[50] Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model. *arXiv preprint arXiv:2512.13030*, 2025.

[51] Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.

[52] Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, Yiyu Ren, Kejia Zhang, Hui Yu, Jingmei Zhao, Shuai Zhou, Zhenqi Qiu, Houlong Xiong, Ziyu Wang, Zechen Wang, Ran Cheng, Yong-Lu Li, Yongtao Huang, Xing Zhu, Yujun Shen, and Kecheng Zheng. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*, 2026.

[53] Qwen Team. Qwen-vla: Unifying vision-language-action modeling across tasks, environments, and robot embodiments. *arXiv preprint arXiv:2605.30280*, 2026.

[54] Xinyi Chen, Yilun Chen, Yanwei Fu, Ning Gao, Jiaya Jia, Weiyang Jin, Hao Li, Yao Mu, Jiangmiao Pang, Yu Qiao, Yang Tian, Bin Wang, Bolun Wang, Fangjing Wang, Hanqing Wang, Tai Wang, Ziqin Wang, Xueyuan Wei, Chao Wu, Shuai Yang, Jinhui Ye, Junqiu Yu, Jia Zeng, Jingjing Zhang, Jinyu Zhang, Shi Zhang, Feng Zheng, Bowen Zhou, and Yangkun Zhu. Internvla-m1: A spatially guided vision-language-action framework for generalist robot policy. *arXiv preprint arXiv:2510.13778*, 2025.

[55] Andy Zhai, Brae Liu, Bruno Fang, Chalse Cai, Ellie Ma, Ethan Yin, Hao Wang, Hugo Zhou, James Wang, Lights Shi, Lucy Liang, Make Wang, Qian Wang, Roy Gan, Ryan Yu, Shalfun Li, Starrick Liu, Sylas Chen, Vincent Chen, and Zach Xu. Igniting vlms toward the embodied space. *arXiv preprint arXiv:2509.11766*, 2025.

[56] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[57] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

[58] Ilia Larchenko, Gleb Zarin, and Akash Karnatak. Task adaptation of vision-language-action model: 1st place solution for the 2025 behavior challenge. *arXiv preprint arXiv:2512.06951*, 2025.

[59] Junjie Bai, Yu-Wei Chao, Qizhi Chen, Jinwei Gu, Moo Jin Kim, Zhaoshuo Li, Xuan Li, Tsung-Yi Lin, Ming-Yu Liu, Nic Ma, et al. Openpi comet: Competition solution for 2025 behavior challenge. *arXiv preprint arXiv:2512.10071*, 2025.

[60] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[61] Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, et al. RLinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation. *arXiv preprint arXiv:2509.15965*, 2025.

A Appendix / supplemental material

Table 6: Detailed results on the 2025 BEHAVIOR Challenge (50 tasks, 10 instances each). *Task Success Score* is the challenge ranking metric (task progress). The first place solution by the Robot Learning Collective (RLC) [58] uses a set of 4 checkpoints; $\pi_{0.5}$ [4] (4 epochs) and **G0.5** each use a single checkpoint, averaged over two eval runs. Best/second best in **bold**/underline.

Task	RLC [58]	Comet [59]	$\pi_{0.5}$ (4 epochs) [4]	**G0.5** (1 epoch)	**G0.5** (4 epochs)
assembling gift baskets	0.2125	0.0000	0.2312	**0.5188**	0.4938
attach camera to tripod	0.0000	0.0000	0.0000	0.0000	0.0000
boxing books for storage	0.0000	0.0000	0.0000	0.0000	0.0000
bringing in wood	0.0667	**0.5000**	0.1667	0.3333	0.3500
bringing water	0.2667	**0.9000**	0.6500	0.8333	0.7000
can meat	0.0000	0.0000	0.0333	0.0111	**0.0444**
canning food	0.0100	0.0000	0.0550	0.0800	**0.1100**
carrying in groceries	0.1500	0.0000	**0.1750**	0.0750	0.1500
chop an onion	0.3000	0.0000	0.1750	**0.4125**	0.4000
chopping wood	0.1000	0.0000	0.0875	**0.2375**	0.2000
clean a patio	0.0000	0.0000	0.0000	0.0000	0.0000
clean a trumpet	0.0000	0.0000	0.0000	0.0000	0.0000
clean boxing gloves	0.2000	0.0000	0.0000	**0.2250**	0.1750
clean desk	0.1273	0.0000	0.2227	**0.2864**	0.2591
clean plates and food	0.1857	0.0000	**0.3857**	0.1786	0.1929
clear food to fridge	0.0800	0.0000	0.0800	0.1800	**0.2700**
collect childrens toys	0.4333	0.0000	0.4214	0.5857	**0.5929**
cook bacon	**0.7571**	0.0000	0.4214	0.0714	0.3214
cook cabbage	0.0000	0.0000	0.0500	0.1500	**0.2000**
cook hot dogs	0.8500	**1.0000**	0.9250	0.4500	0.9000
freeze pies	0.0143	**0.1571**	0.1357	0.0571	0.0429
get organized for work	0.0200	0.0000	0.0200	0.1050	**0.1100**
hanging pictures	0.0000	**0.2000**	0.0000	0.0000	0.0000
hiding Easter eggs	0.1444	**0.2444**	0.0778	0.0500	0.0500
loading the car	0.2000	0.0000	0.0000	0.2333	**0.2833**
make microwave popcorn	0.9000	0.7000	**0.9500**	0.1500	0.5500
make pizza	0.0000	0.0000	0.0000	0.0000	0.0000
move boxes to storage	0.6500	**1.0000**	0.2000	0.6250	0.5500
outfit basic toolbox	0.2571	0.1000	**0.3429**	0.1429	0.2500
pick up toys	0.3000	0.0000	0.1833	0.3167	**0.4083**
pick up trash	0.6667	0.7667	0.5500	0.8167	**0.8500**
prepare lunch box	0.5167	0.0000	0.5417	**0.5667**	**0.5667**
put away Halloween decor.	0.2000	0.5000	0.3714	0.4857	**0.5286**
put dishes away	0.2714	0.0000	**0.5250**	0.1393	0.2464
put shoes on rack	0.5000	0.5400	0.3650	**0.6450**	0.5650
put up Christmas decor.	0.4333	0.0000	**0.5611**	0.4667	0.3889
rearrange kitchen furn.	0.3000	0.3750	**0.3875**	0.3500	0.3625
set up coffee station	0.1500	0.2167	**0.2833**	0.2500	0.1917
setting mousetraps	0.3333	0.0000	0.1083	0.5083	**0.5667**
setting the fire	**0.3250**	0.2000	0.0750	0.3125	0.1250
slicing vegetables	0.1889	0.0000	0.2278	**0.4444**	0.2611
sorting household items	0.0625	0.0000	0.1938	0.1437	**0.2062**
sorting vegetables	0.4769	0.0000	0.6000	0.5231	**0.6231**
spraying for bugs	**0.2500**	0.1000	0.1000	0.2000	0.1500
spraying fruit trees	0.3000	**0.3500**	0.2000	0.1250	0.1500
storing food	0.3750	0.0000	0.5750	0.4938	**0.6625**
tidying bedroom	0.4000	0.0000	0.3500	0.5667	**0.6167**
turning on radio	0.6000	**1.0000**	0.1500	0.3000	0.0500
wash a baseball cap	0.4500	0.3000	0.6000	0.6500	**0.7500**
wash dog toys	0.0000	0.0000	**0.3750**	0.2250	0.2167
Overall	0.2605	0.1830	0.2626	0.2904	**0.3136**

Table 7: Per-task success rates of **G0.5** on RoboTwin 2.0 under clean and randomized evaluation settings.

Task	Clean	Rand.
Adjust Bottle	100	100
Beat Block Hammer	100	96
Blocks Ranking RGB	100	100
Blocks Ranking Size	96	96
Click Alarmclock	100	100
Click Bell	100	100
Dump Bin Bigbin	94	96
Grab Roller	100	100
Handover Block	98	84
Handover Mic	98	100
Hanging Mug	40	56
Lift Pot	100	100
Move Can Pot	98	94
Move Pillbottle Pad	96	96
Move Playingcard Away	100	98
Move Stapler Pad	80	70
Open Laptop	100	98
Open Microwave	92	88
Pick Diverse Bottles	84	86
Pick Dual Bottles	96	88
Place A2B Left	94	94
Place A2B Right	94	90
Place Bread Basket	98	98
Place Bread Skillet	96	90
Place Burger Fries	100	98
Place Can Basket	68	72
Place Cans Plasticbox	96	100
Place Container Plate	100	98
Place Dual Shoes	88	94
Place Empty Cup	100	100
Place Fan	100	94
Place Mouse Pad	84	84
Place Object Basket	90	92
Place Object Scale	96	94
Place Object Stand	100	96
Place Phone Stand	94	98
Place Shoe	98	94
Press Stapler	92	92
Put Bottles Dustbin	90	90
Put Object Cabinet	90	90
Rotate QRcode	96	98
Scan Object	100	92
Shake Bottle	100	100
Shake Bottle Horizontally	100	100
Stack Blocks Three	98	100
Stack Blocks Two	100	100
Stack Bowls Three	94	86
Stack Bowls Two	96	90
Stamp Seal	88	92
Turn Switch	74	80
Average	**93.72**	**92.84**