A Systematic Study of Data Modalities and Strategies for Co-training Large Behavior Models for Robot Manipulation

Fanqi Lin[†‡], Kushal Arora[†], Jean Mercat[†], Haruki Nishimura[†], Paarth Shah[†], Chen Xu[†], Mengchao Zhang[†], Mark Zolotas[†], Maya Angeles[†], Owen Pfannenstiehl[†], Andrew Beaulieu[†], Jose Barreiros[†]

[†]Toyota Research Institute, Cambridge MA and Los Altos CA, USA
[‡]Tsinghua University, Beijing, China

***Abstract*—Large behavior models (LBMs) have shown strong dexterous manipulation capabilities by extending imitation learning to large-scale training on extensive multi-task robot data, yet their generalization remains limited by the insufficient coverage of available robot data. To expand this coverage without costly additional data collection, recent work increasingly relies on *co-training*: jointly learning from target robot data and heterogeneous data modalities. However, how different co-training data modalities and training strategies affect policy performance remains poorly understood. We present a large-scale empirical study examining five co-training data modalities—standard vision-language data, dense language annotations for robot trajectories, cross-embodiment robot data, human videos, and discrete robot action tokens—across single- and multi-phase training strategies. Our study leverages 4,000 hours of robot and human manipulation data and 50M vision–language samples to train vision-language-action (VLA) policies. We evaluate 89 policies over 58,000 simulation rollouts and 2,835 real-world rollouts. Our results show that co-training with various forms of vision-language and cross-embodiment robot data substantially improves generalization to distribution shifts, unseen tasks, and language following, while discrete action token variants yield no statistically significant benefits. Furthermore, combining effective modalities produces cumulative gains and enables rapid adaptation to unseen long-horizon dexterous tasks via fine-tuning. We find that training exclusively on robot data degrades the visiolinguistic understanding of the vision-language model backbone, while co-training with the effective data modalities restores these capabilities, as measured by standard vision-language, spatial reasoning, and multimodal reasoning benchmarks. Finally, explicitly conditioning action generation on chain-of-thought traces learned from co-training data does not improve performance in our simulation benchmark. Together, these results provide a systematic understanding of co-training and practical guidance for building scalable generalist robot policies.**

Project page:https://co-training-lbm.github.io/

I. Introduction

Robot learning is increasingly moving towards generalist models that can perceive, understand, and act in the physical world. Recent efforts have focused on training large behavior models (LBMs) [1]—embodied foundation models trained on large-scale multi-task robot datasets—to produce dexterous manipulation policies. Within this family, vision-language-action models (VLAs) [37, 100, 21, 6, 33, 67, 5] are a representative subclass of LBMs that integrate visual and linguistic inputs for action generation. Despite progress, LBMs still lag behind non-embodied foundation models, such as vision-language models (VLMs) [32, 16, 65, 73], in semantic and spatial understanding and in open-world generalization. This limitation can be attributed to the significant disparity in data scale [25]: robot data is orders of magnitude smaller than the internet-scale text and image corpora used to train VLMs.

To bridge this data gap, many recent works [33, 44, 5, 39, 13, 61, 84] use *co-training*—jointly learning target robot (i.e., the deployment embodiment) data alongside heterogeneous data modalities, aiming to enhance the model's understanding of the physical world and its generalization abilities. These co-training data modalities include: standard vision-language (VL) data [33, 99, 98, 39], dense language annotations for robot trajectories [92, 44, 93, 84, 61, 11], cross-embodiment robot data [1, 50, 48, 69, 37, 38], human videos [10, 13, 88, 52, 90, 36], and discrete robot action tokens [93, 22, 35, 33]. Despite the growing interest, current studies typically evaluate only a subset of these modalities using inconsistent experimental setups, leaving the empirical effectiveness of co-training largely unexplored.

In this work, we systematically investigate how different data modalities and co-training strategies affect policy performance through large-scale experiments toward a generalist LBM. We adopt a VLA architecture consisting of a pretrained VLM backbone and an action head. Our model is trained with flow matching [46, 49] to predict continuous robot actions and next-token objectives for discrete tokens. An overview of our study is illustrated in Fig. 1.

We evaluate five major co-training data modalities (Fig. 2): **(1) Standard vision-language data**, encompassing visual question answering, object localization, and spatial reasoning tasks, which provide rich commonsense knowledge about the physical world. **(2) Dense language annotations for robot trajectories**, generated through both heuristic scripting and VLM-based captioning, offering explicit semantic labels for actions, goals, and object relationships. **(3) Cross-embodiment robot data**, which encompasses manipulation demonstrations across varied robot morphologies and environments. **(4) Large-scale egocentric human videos** that

Correspondence to: jose.barreiros@tri.global

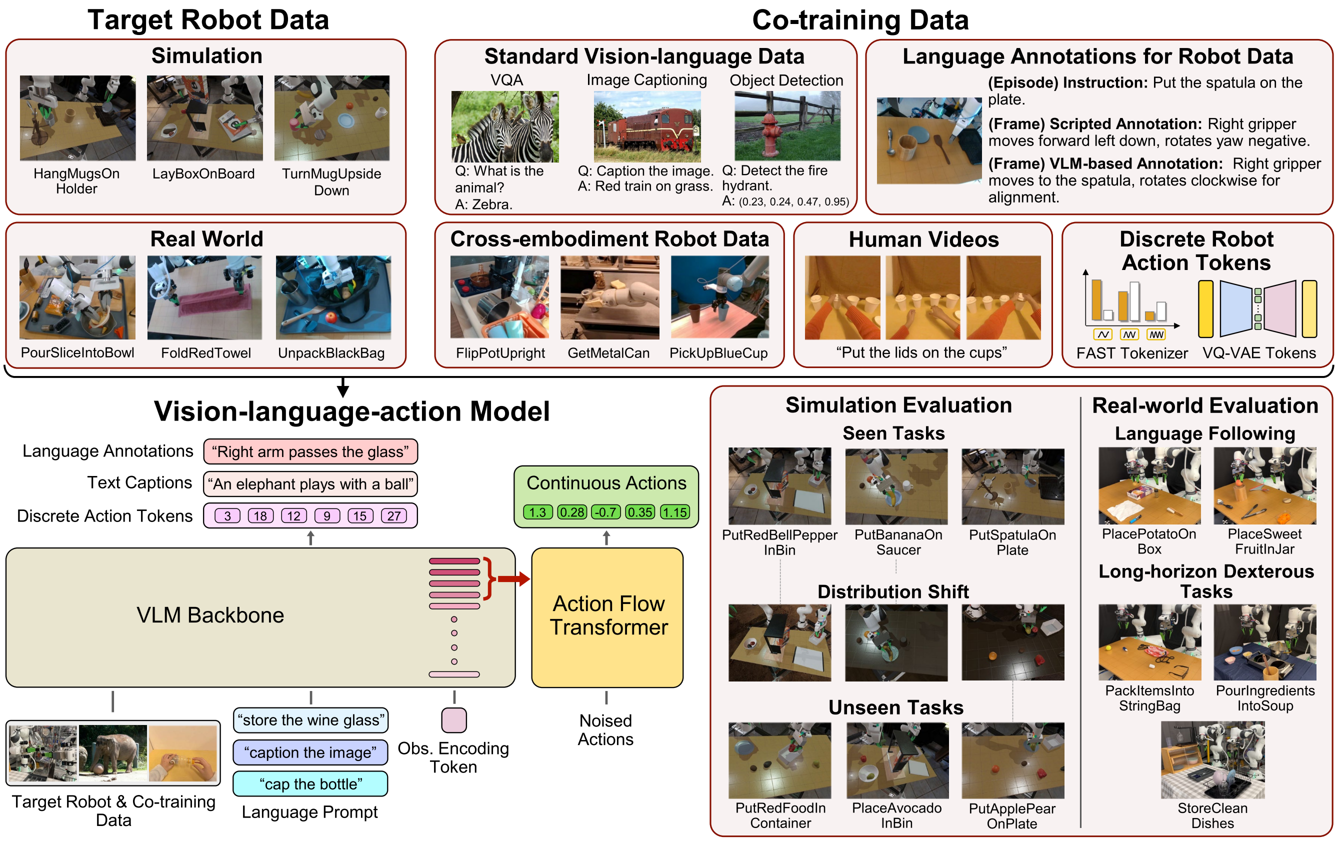


Fig. 1. **Overview of the data, model architecture, and evaluation setup.** Our policy is built on a pretrained vision-language model backbone combined with an Action Flow Transformer. It is trained on target robot data alongside heterogeneous co-training modalities, including standard vision-language data, dense language annotations for robot data, cross-embodiment robot data, human videos, and discrete robot action tokens. We evaluate policies in simulation on seen and unseen tasks, under nominal conditions and distribution shifts, and in the real-world for language following, and long-horizon dexterous manipulation.

expose models to diverse visual contexts, object interactions, and motion patterns beyond robot trajectories. We explore two approaches to leverage human videos: first, extracting discrete latent action tokens from a sequence of frames; second, generating language annotations by prompting a VLM with visual context and a task instruction to describe motions, goals, and objects on a per-frame basis. **(5) Discrete robot action tokens**, where continuous robot actions are compressed into low-dimensional discrete spaces through frequency-based methods (e.g., FAST [58]) or vector quantization-based methods (e.g., VQ-VAE [72]), probing whether such abstraction improves generalization.

We also study strategies for incorporating these modalities at different training phases (i.e., rounds of training with distinct data composition). Additionally, we examine whether combining effective co-training modalities yields cumulative performance gains. We further probe whether co-training improves representation quality by fine-tuning on unseen long-horizon, dexterous tasks. We then study how the VLM backbone is shaped by effective co-training modalities using a suite of standard vision-language benchmarks. Finally, we study explicit chain-of-thought (CoT) [77] conditioning for action generation, where the policy first produces intermediate CoT traces learned from the co-training data and then uses them to generate continuous actions.

Our policies are evaluated in both simulation and real-world settings. In total, we train and compare 89 VLA policies using about 4,000 hours of robot and human manipulation data plus 50M vision-language samples. The policies are assessed over 58,000 simulation rollouts across seen and unseen tasks, in nominal and distribution shift (DS) settings, and over 2,835 real-world rollouts covering language following and long-horizon dexterous tasks.

Our results provide practical guidance for co-training LBMs. In summary: (1) Diverse vision-language data and cross-embodiment robot data consistently improve generalization to distribution shifts, unseen tasks, and language following, while discrete action token variants provide no benefit. (2) Combining effective modalities yields cumulative gains and enables more efficient adaptation to unseen long-horizon dexterous tasks via fine-tuning. (3) Training exclusively on robot data can erode the visiolinguistic capabilities of the VLM backbone, whereas effective co-training helps preserve this understanding, as reflected by improved performance on standard vision-language benchmarks. (4) Explicitly conditioning action generation on chain-of-thought traces learned from co-training data does not improve performance in our simulation benchmark. Together, these findings constitute a controlled, large-scale empirical map of which co-training signals and strategies are most useful for building scalable, generalist robot policies.

II. Method

In this section, we first introduce our co-training framework in Section II-A, including the problem formulation, model architecture, and our co-training and inference strategies. We then describe in Section II-B how the target robot data and diverse co-training datasets are curated for this study.

A. Co-training Framework

1) Problem Formulation: Our objective is to learn a policy π_θ that can leverage diverse co-training data modalities. The policy π_θ takes as input a sequence of n images $I^{1:n}$ and a text prompt ℓ. For continuous robot actions, the model is trained with flow matching (FM) [46, 49] as the learning objective. Specifically, given an action chunk A_t, a FM timestep $\tau \in [0, 1]$, and sampled noise $\epsilon \sim N(0, \mathbf{I})$, we construct a noised action chunk as $A_t^\tau = \tau A_t + (1 - \tau)\epsilon$. The model is then trained by minimizing the loss:

$$\mathcal{L}_{\text{FM}} = \left\|\pi_\theta^a\big(I_t^{1:n}, \ell, A_t^\tau, \tau\big) - (A_t - \epsilon)\right\|^2, \tag{1}$$

where $(A_t - \epsilon)$ and $\pi_\theta^a(\cdot)$ are the ground-truth and predicted flow vector, respectively. For text tokens or discrete action tokens, the model is optimized to minimize the cross-entropy (CE) loss between the ground-truth token sequence $x_{1:M}$ and the predicted logits $\pi_\theta^\ell(\cdot)$, such that:

$$\mathcal{L}_{\text{CE}} = \mathcal{H}\big(x_{1:M}, \pi_\theta^\ell\big(I_t^{1:n}, \ell\big)\big) \tag{2}$$

When jointly optimizing for both continuous and discrete modalities, we combine the objectives into a weighted sum:

$$\mathcal{L} = M_{FM}\mathcal{L}_{FM} + w * M_{CE}\mathcal{L}_{CE}, \tag{3}$$

Here, w is the weight applied to the CE loss, M_{FM} is the FM loss mask indicating whether the model should predict continuous actions for a given sample, and M_{CE} is the mask specifying token positions used to compute the CE loss.

2) Model Architecture: We adopt a VLA architecture (Fig. 1) composed of a pretrained VLM backbone and a flow transformer action head (ActionFT). The VLM is initialized from PaliGemma2-PT (google/paligemma2-3b-pt-224 [65]). It encodes both the observed images and the language prompt describing the task, and can optionally be trained to generate text or discrete action tokens in addition to providing visiolinguistic representations. To obtain a compact representation for action generation, similarly to [45], we introduce a special observation encoding token into the backbone's vocabulary and append it to the end of the text prompt. We extract the hidden state vectors corresponding to this token from the last four layers of the VLM to form a single global conditioning embedding and feed this into the ActionFT. The ActionFT follows the diffusion transformer design introduced in [1]. It consists of 8 flow transformer layers, each conditioned on observation features and the flow-matching timestep via an adaptive layer norm (adaLN) MLP [56]. The ActionFT receives the global conditioning embedding, a noised chunk of continuous actions, and the flow matching time variable, and is trained to predict the flow vector that guides iterative denoising of the actions toward the target trajectory. Unlike prevalent methods [6], our approach uses only a single token as visiolinguistic representations rather than attention keys and values from all VLM layers. Our ablation studies (see Appendix 1A) indicate that this compact representation enhances the model's generalization ability to unseen tasks and distribution shift.

3) Co-training and Inference Strategies: Co-training data can be used at different phases of model training. We explore three strategies (Table I): (1) *Single-phase co-training*: train jointly on target robot data and co-training data modalities in a single phase; (2) *Two-phase 1st-phase-only co-training*: train only on co-training modalities during the 1st-phase, then on target robot continuous actions in the 2nd-phase; (3) *Two-phase full co-training*: the same 1st-phase as strategy (2), but train on both co-training and target robot data in the 2nd-phase.

We fix the training loss weights and batch data ratios based on ablation experiment results (Appendix 1B). Full training details are provided in Appendix 2.

Beyond investigating when to incorporate co-training data, we examine ways to leverage it. While our primary approach treats co-training data solely as auxiliary supervision, recent works [44, 92, 93, 67, 11, 13] suggest that explicitly conditioning action prediction on CoT traces can boost policy performance. We therefore also test this alternative paradigm for selected co-training modalities: language annotations for robot trajectories and latent actions for videos.

When generating actions conditioned on CoT traces at inference time, the VLM backbone first produces a CoT trace (see Fig. S4). The observation token is then appended to its end, and the resulting visiolinguistic embedding (encoding images, task prompts, and CoT traces) is extracted to condition the ActionFT for continuous action prediction. During training, we introduce probabilistic CoT conditioning: with probability p, the policy is trained to generate actions conditioned on CoT traces extracted from the co-training data; with probability $1 - p$, the policy generates actions directly without CoT conditioning. Importantly, CoT conditioning differs from other co-training strategies only in that the predicted co-training tokens are used directly to form the visiolinguistic embedding, instead of just providing auxiliary supervision. We evaluate the impact of this explicit inference-time CoT approach in Section III-F.

B. Data Curation

We curate a comprehensive dataset consisting of target-robot expert demonstration data and five distinct co-training modalities. Our dataset comprises approximately 4,000 hours of manipulation data spanning both robot episodes and human videos, complemented by 50M vision-language samples that encompass standard vision-language data as well as dense annotations for both robot and human data. Fig. 2 shows an overview of the training data.

1) Target Robot Data: We adopt high-quality robot data from our previous study [1] as our primary in-distribution

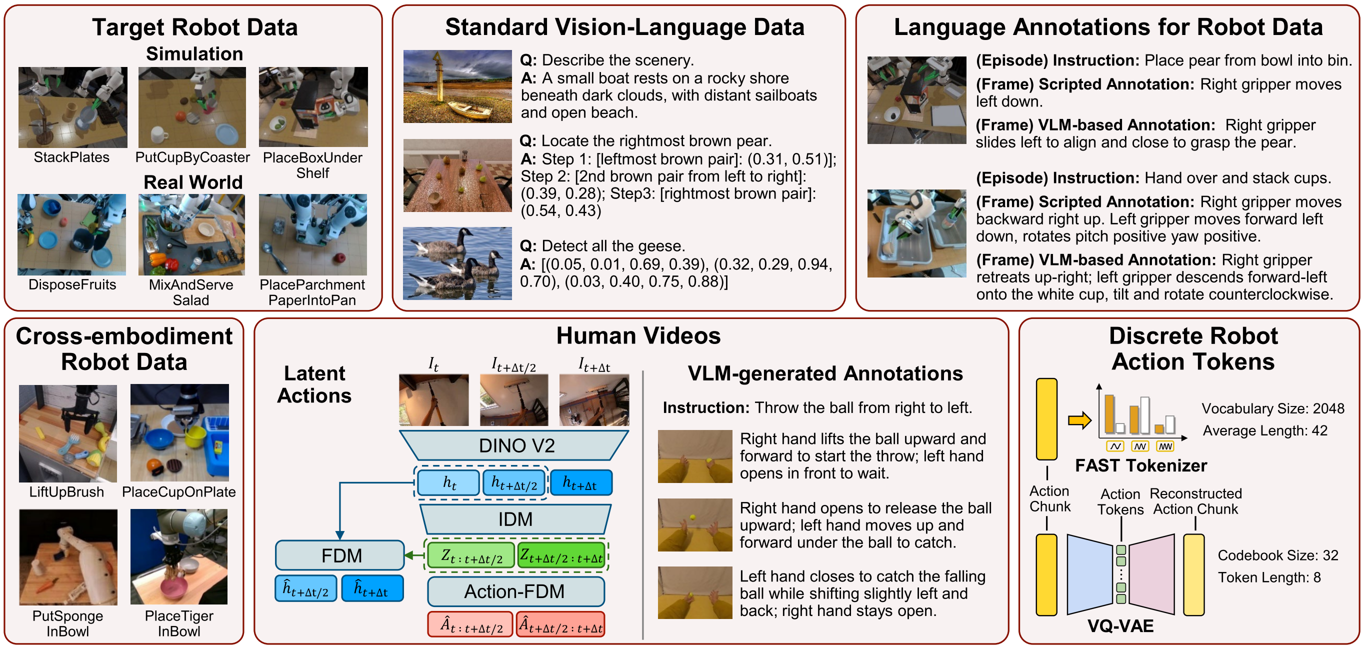

Target Robot Data
Simulation
StackPlates
PutCupByCoaster
PlaceBoxUnder Shelf
Real World
DisposeFruits
MixAndServe Salad
PlaceParchment PaperIntoPan
Standard Vision-Language Data
Q: Describe the scenery.
A: A small boat rests on a rocky shore beneath dark clouds, with distant sailboats and open beach.
Q: Locate the rightmost brown pear.
A: Step 1: [leftmost brown pair]: (0.31, 0.51)]; Step 2: [2nd brown pair from left to right]: (0.39, 0.28); Step3: [rightmost brown pair]: (0.54, 0.43)
Q: Detect all the geese.
A: [(0.05, 0.01, 0.69, 0.39), (0.32, 0.29, 0.94, 0.70), (0.03, 0.40, 0.75, 0.88)]
Language Annotations for Robot Data
(Episode) Instruction: Place pear from bowl into bin.
(Frame) Scripted Annotation: Right gripper moves left down.
(Frame) VLM-based Annotation: Right gripper slides left to align and close to grasp the pear.
(Episode) Instruction: Hand over and stack cups.
(Frame) Scripted Annotation: Right gripper moves backward right up. Left gripper moves forward left down, rotates pitch positive yaw positive.
(Frame) VLM-based Annotation: Right gripper retreats up-right; left gripper descends forward-left onto the white cup, tilt and rotate counterclockwise.
Cross-embodiment Robot Data
LiftUpBrush
PlaceCupOnPlate
PutSponge InBowl
PlaceTiger InBowl
Human Videos
Latent Actions
I_t
$I_{t+\Delta t/2}$
$I_{t+\Delta t}$
DINO V2
h_t
$h_{t+\Delta t/2}$
$h_{t+\Delta t}$
IDM
FDM
$Z_{t:t+\Delta t/2}$
$Z_{t+\Delta t/2:t+\Delta t}$
$\hat{h}_{t+\Delta t/2}$
$\hat{h}_{t+\Delta t}$
Action-FDM
$\hat{A}_{t:t+\Delta t/2}$
$\hat{A}_{t+\Delta t/2:t+\Delta t}$
VLM-generated Annotations
Instruction: Throw the ball from right to left.
Right hand lifts the ball upward and forward to start the throw; left hand opens in front to wait.
Right hand opens to release the ball upward; left hand moves up and forward under the ball to catch.
Left hand closes to catch the falling ball while shifting slightly left and back; right hand stays open.
Discrete Robot Action Tokens
Vocabulary Size: 2048
Average Length: 42
FAST Tokenizer
Action Chunk
Action Tokens
Reconstructed Action Chunk
Codebook Size: 32
Token Length: 8
VQ-VAE

Fig. 2. **Overview of the training data.** Our dataset comprises target robot data collected in both simulation and real-world, and five heterogeneous co-training data modalities: standard vision-language data for commonsense understanding, spatial reasoning, and object grounding; dense language annotations for robot data, generated via heuristic scripting and VLM-based captioning; cross-embodiment robot data capturing diverse robot morphologies and manipulation tasks; human videos, from which we derive either latent action tokens using a latent action model (LAM) or VLM-generated annotations; and discrete robot action tokens, including near-lossless FAST tokens and compact VQ-VAE tokens. These co-training modalities, together with the target robot data, constitute a unified dataset of ∼4,000 hours of manipulation data and 50M vision-language samples.

TABLE I
DATA COMPOSITION FOR DIFFERENT CO-TRAINING STRATEGIES.

Co-training strategy	Phase	Data composition	
		Target-robot continuous-action data	Co-training data
Single-phase co-training	1st-phase	✓	✓
Two-phase 1st-phase-only co-training	1st-phase	–	✓
	2nd-phase	✓	–
Two-phase full co-training	1st-phase	–	✓
	2nd-phase	✓	✓

training corpus. This dataset, which is referred to as TRI-Ramen, contains 523 hours of manipulation data spanning 403 tasks and 53,411 demonstrations. It consists of real-world (478 hours, 362 tasks, 46,063 demonstrations; namely TRI-Ramen-Real) and simulation data (45 hours, 41 tasks, 7,348 demonstrations; namely TRI-Ramen-Sim). All demonstrations are collected via teleoperation on dual Franka Panda robotic arms as described in [1].

The observation space includes i) 4 RGB images (missing images are zero-padded), and ii) a natural-language instruction. The action space includes i) end-effector poses w.r.t. the station's base frame, and ii) gripper widths. Actions are represented as relative trajectories as in [15]. The policy is conditioned only on the current observation and predicts an action chunk with a horizon of 16.

2) Standard Vision-Language Data: To enhance the model's multimodal understanding (e.g., semantics, spatial, planning), we incorporate VL datasets specifically designed for robotic scenarios: RoboPoint [91] and RefSpatial [96]. RoboPoint comprises 1.3M data samples with 8.2M question answering pairs, while RefSpatial contains 2.5M samples with 20M question answering pairs. Both datasets provide annotations tailored for spatial referring tasks, spanning qualitative visual question answering, quantitative queries on object and spatial attributes/relations, point coordinate prediction, and multi-step spatial reasoning.

3) Dense Language Annotations for Robot Trajectories: To augment TRI-Ramen robot trajectories with action-grounded textual descriptions, we employ two complementary annotation strategies: **(1) Scripted Annotation.** Following [92], we apply heuristic rules to generate per-step low-level action primitives by comparing future and current end-effector states over a 16-step horizon (corresponding to the horizon of the action chunk). These annotations capture the robot's end effector translational movement, rotational changes, and gripper state transitions. **(2) VLM-based Annotation.** While scripted annotations provide structured action descriptions, their linguistic diversity is limited and they lack contextual information about object and environment interactions. To address this, we prompt a VLM (GPT-5 [53]) to generate rich, contextually grounded descriptions. Specifically, for each robot episode, we provide the VLM with: (i) frames downsampled at 2-second intervals (approximating the 1.6-second action chunk duration), (ii) the episode-level task instruction, (iii) action hints generated by the heuristic rules described above, and (iv) a reference image depicting a world-frame coordinate to calibrate spatial directions (forward/backward, left/right, up/down). The VLM is then prompted to produce diverse frame-level action descriptions that capture interactions with objects and the environment. To achieve a higher temporal density, we process the dataset twice with a one-second offset between passes, resulting in annotations at a per-second frequency. More details are available in Appendix 3A.

4) Cross-embodiment Robot Data: We adopt the same cross-embodiment dataset used in [1], referred to as OXE-Ramen, which is a curated subset of the Open X-Embodiment dataset [55]. This collection encompasses diverse robot morphologies and manipulation scenarios, totaling 1,150 hours across 12 robot setups, 924 tasks, and 466,415 demonstrations. Observation and action spaces follow the same as in the target robot data.

5) Human Videos: To extract rich information about the motion and action (e.g, move forward and grasp) from human videos, we explore two distinct approaches: **(1) Latent Actions.** We utilize publicly available egocentric human video datasets (e.g., Ego4D [28], EgoDex [31], Something-Something V2 [27], Epic Kitchen [18], HoloAssist [75]), totaling 2,271 hours after filtering (see detailed data composition in Table S3). We train a latent action model (LAM) jointly on human videos, TRI-Ramen, and OXE-Ramen to learn a unified discrete action representation. Given consecutive frames I_t, $I_{t+\frac{\Delta t}{2}}$, $I_{t+\Delta t}$, we encode them using a pretrained DINOv2 [54] vision encoder to obtain visual features h_t, $h_{t+\frac{\Delta t}{2}}$, $h_{t+\Delta t}$. Following [13, 10], the LAM learns a quantized codebook of latent actions (codebook size C) using three modules: an inverse dynamics model (IDM), a visual forward dynamics model (FDM), and an action forward dynamics model (ActionFDM).

The IDM predicts two latent action segments:

$$Z_{t:t+\frac{\Delta t}{2}},\ Z_{t+\frac{\Delta t}{2}:t+\Delta t} = \mathrm{IDM}\Big(h_t,\ h_{t+\frac{\Delta t}{2}},\ h_{t+\Delta t}\Big)\,, \tag{4}$$

while the FDM reconstructs future visual features:

$$\hat{h}_{t+\frac{\Delta t}{2}} = \mathrm{FDM}\Big(h_t,\ Z_{t:t+\frac{\Delta t}{2}}\Big)\,, \tag{5}$$

$$\hat{h}_{t+\Delta t} = \mathrm{FDM}\Big(h_{t+\frac{\Delta t}{2}},\ Z_{t+\frac{\Delta t}{2}:t+\Delta t}\Big)\,. \tag{6}$$

To encourage latent tokens to capture physical dynamics alongside visual changes, we additionally supervise on robot data (TRI-Ramen, OXE-Ramen) by reconstructing the ground-truth action chunks:

$$\hat{A}_{t:t+\frac{\Delta t}{2}} = \mathrm{ActionFDM}\Big(Z_{t:t+\frac{\Delta t}{2}}\Big)\,, \tag{7}$$

$$\hat{A}_{t+\frac{\Delta t}{2}:t+\Delta t} = \mathrm{ActionFDM}\Big(Z_{t+\frac{\Delta t}{2}:t+\Delta t}\Big)\,. \tag{8}$$

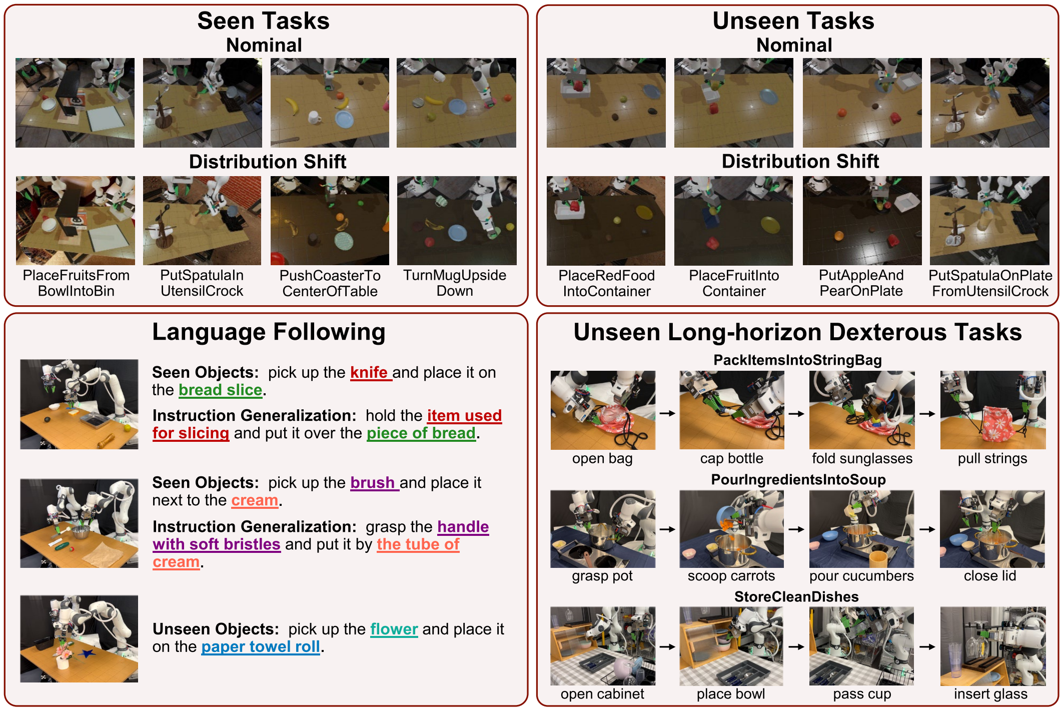
For human videos, we omit action reconstruction due to missing ground-truth actions. After training, we run inference to obtain $Z_{t:t+\frac{\Delta t}{2}}$ and $Z_{t+\frac{\Delta t}{2}:t+\Delta t}$ at each time step, concatenate them to form Z_t, and quantize each Z_t into 8 discrete tokens from a codebook of size $C = 32$. For robot videos, $\Delta t = 1.6$ seconds (matching the 16-step action chunk), while for human videos we use $\Delta t = 1.0$ seconds to account for faster motion. Implementation details are provided in Appendix 3B.1. **(2) VLM-generated Annotations.** As an alternative to latent actions, language can also serve as a unified representation across different embodiments. It captures rich semantic information about actions, goals, and objects while being naturally compatible with VLAs. Specifically, we provide GPT-5 with: (i) frames from human videos downsampled at 1-second intervals, (ii) episode-level instructions, and (iii) a reference image depicting the world-frame coordinate triad. We prompt the VLM to generate concise motion descriptions of both hands, including rich information about interactions with objects and the environment. We utilize Ego4D, EgoDex and Something-Something V2 datasets, yielding 9M annotated data samples. During training, we treat these samples as another form of VL data. More details are shown in Appendix 3B.2.

6) Discrete Robot Action Tokens: Several works suggest that co-training models on both continuous and discrete robot action representations improves sample efficiency and generalization. Motivated by this, we explore two forms of discrete robot action tokens: **(1) FAST Tokens.** We employ FAST [58] to convert continuous action chunks into a compressed, near-lossless sequence of discrete tokens. We use the off-the-shelf tokenizer without fine-tuning, as we observe no significant improvements in either average token length or reconstruction error after fine-tuning on our data (see Appendix 3C). When applied to our TRI-Ramen dataset, FAST produces sequences of average length 42.1 with a vocabulary of 2,048 tokens. **(2) VQ-VAE Discrete Action Tokens.** We compress action chunks into 8 discrete tokens with codebook size 32 using a VQ-VAE [72] on both TRI-Ramen and OXE-Ramen. Notably, these dimensions are identical to those of the latent actions learned from videos. This results in a much more compact, lower-dimensional representation compared to FAST tokens.

III. Experiments

To systematically investigate the effectiveness of co-training data modalities and strategies, we conduct large-scale experiments to address the following research questions:

1) How do different co-training data modalities, incorporated at different training phases, influence policy performance on various dimensions (in-distribution, generalization to distribution shift, unseen tasks, and language following)?
2) Does combining effective co-training modalities yield cumulative performance gains?
3) Can co-training enhance the quality of learned representations, thereby enabling rapid adaptation via fine-tuning to unseen long-horizon, dexterous tasks?
4) How do the effective co-training modalities shape the VLM backbone?



Seen Objects: pick up the knife and place it on the bread slice.

Instruction Generalization: hold the item used for slicing and put it over the piece of bread.

Seen Objects: pick up the brush and place it next to the cream.

Instruction Generalization: grasp the handle with soft bristles and put it by the tube of cream.

Unseen Objects: pick up the flower and place it on the paper towel roll.

Fig. 3. **Simulation and real-world evaluation.** Policies are evaluated in simulation on 13 seen and 8 unseen tasks under nominal and distribution shift (DS) conditions, where DS introduces appearance changes (e.g., lighting, textures, distractors, camera parameters). Real-world evaluations include language-following experiments with seen objects, instruction generalization through paraphrasing, and unseen objects, as well as adaptation to unseen long-horizon dexterous tasks via fine-tuning.

5) How does explicitly conditioning action generation on CoT learned from co-training data affect policy performance?

A. Evaluation

We evaluate policy performance across multiple important dimensions: in-distribution performance, robustness to DS, generalization to unseen tasks, and language following. We additionally assess adaptation to unseen long-horizon dexterous manipulation tasks via fine-tuning. To this end, we conduct large-scale simulation and real-world experiments (Fig. 3).

1) Simulation Benchmark: We adopt the simulation benchmark presented in [1] built on top of Drake [70]. Our benchmark includes 13 seen tasks and 8 unseen tasks (3 unseen tasks defined in [1] plus 5 newly introduced to further probe generalization; see Appendix 6). Each policy is evaluated with 50 rollouts per task under both nominal and DS conditions, and performance is measured by success rate.

The 13 seen tasks fall within the training distribution, while the 8 unseen tasks are designed to probe generalization beyond it. These unseen tasks span several challenges: (i) semantic understanding (e.g., identifying "red food" in *PlaceRedFoodIntoContainer*, or distinguishing fruits from vegetables in *PlaceFruitIntoContainer*); (ii) multi-step manipulation (e.g., sequentially placing an apple and a pear on a plate in *PutAppleAndPearOnPlate*); and (iii) compositional generalization (e.g., generalizing from training demonstrations of placing object A on C and object B on D to placing object A on D). Here, "unseen tasks" refer to skills that do not appear in the training data, although the underlying objects and environments do.

To assess the model's robustness to appearance changes, we use DS conditions of the simulation benchmark, which introduce variations in lighting, environmental backgrounds, camera parameters, objects and table textures, and colors relative to the training distribution. Nominal conditions maintain consistency with the training data across these factors.

2) Real-world Evaluation: We evaluate policies on a dual-arm Franka robot platform across the following three settings. Details of policy deployment are provided in Appendix 4.

Language Following. To evaluate the model's ability to follow natural language instructions, we design a suite of language-guided pick-and-place experiments that assess three distinct scenarios: **(i) Seen Objects:** Objects in this setting appear in the training data. Instructions follow a simple template: "pick up [object A] and place it in/on/next to [object B]", where objects are explicitly referenced by name. **(ii) Instruction Generalization:** This setting tests the model's ability to comprehend the underlying meaning behind natural language by rephrasing the instruction. Specifically, the policy must (1) understand semantic object categories (e.g., "writing tools" referring to pens), (2) recognize objects by their attributes (e.g., "the handle with soft bristles" referring to a brush), and (3) demonstrate robustness to paraphrasing (e.g., altered syntax and verb choices). **(iii) Unseen Objects:** This setting employs objects absent from the target robot training data. Instructions follow the same template as the seen objects setting.

Across all three settings, we evaluate 15 spatial layouts, each containing 6-8 tabletop objects. For each layout, we use three distinct language instructions targeting different objects, yielding a total of 45 rollouts per setting. For the instruction generalization setting, we employ identical spatial layouts and target manipulation outcomes as the seen objects setting, but vary the instruction phrasing. The full evaluation suite covers 49 seen objects and 52 unseen objects in total. We report average task completion percentage as the evaluation metric (see Appendix 7 for rubrics, experimental procedure, and rubric quality assurance (QA)).

Long-horizon Dexterous Manipulation. To investigate whether co-training facilitates rapid adaptation of the pretrained model to new challenging tasks unseen during pretraining, we design three long-horizon, dexterous manipulation tasks: *PackItemsIntoStringBag*, *PourIngredientsIntoSoup*, and *StoreCleanDishes*. On average, each task consists of 13 steps and takes 93 seconds to execute. These tasks require fine-grained manipulation skills beyond simple pick-and-place operations, such as capping a bottle, scooping food from a bowl into a pot using a spatula, or inserting a wine glass upside-down into a dish rack. For each task, we collect 200 demonstrations for fine-tuning. Each checkpoint is evaluated for 30 rollouts per task, and we report average task completion percentage as the evaluation metric (see Appendix 7 for rubrics, experimental procedure, and rubric QA).

3) Statistical Analysis Framework: We perform rigorous statistical analysis similar to recent work [1], with pairwise hypothesis tests [78, 64] and Compact Letter Display (CLD) [59] for comparison. Co-training strategies not sharing any CLD alphabet are significantly different in average performance at 5% family-wise error rate (FWER). Bayesian uncertainty estimates are reported for individual strategies, with posterior uncertainty visualized as violin plots overlaid on bar charts. Dots and horizontal lines indicate empirical and posterior means, respectively. Raw empirical distributions are reported in Appendix 5 for the task-completion results, together with more details of our statistical framework.

B. How do different co-training modalities, incorporated at different training phases, influence policy performance?

We evaluate each co-training data modality using the three strategies described in Section II-A3: *single-phase co-training*, *two-phase 1st-phase-only co-training*, and *two-phase full co-training*. We compare the policies obtained from these strategies against a baseline policy trained exclusively on continuous robot actions from TRI-Ramen (namely, *no-co-training baseline*) using the flow-matching objective (M_{CE}=0). All policies are first evaluated in simulation, with results summarized in Fig. 4. Modalities found to be effective are then evaluated in real-world language-following experiments (Fig. 5).

Standard Vision-language Data Co-training. (1) As shown in Fig. 4A and 5A, co-training with standard VL data substantially improves robustness to DS, generalization to unseen tasks, and language following, with no statistically significant change in in-distribution performance (seen tasks). (2) Fine-tuning the pretrained VLM on our curated VL data enhances its representation for robot manipulation tasks, as evidenced by gains from *two-phase 1st-phase-only co-training* over the baseline (Fig 4A, 5A). (3) Continuing to co-train with VL data during the 2nd-phase further enhances performance on unseen tasks and language following (particularly with unseen objects). We hypothesize that this continued exposure allows the model to retain the rich, generalizable knowledge from the VL corpus, which is absent in the robot data, thereby preventing catastrophic forgetting.

Dense Language Annotations for Robot Data Co-training. (1) Co-training with scripted (Fig. 4B, 5B) and VLM-based (Fig. 4C, 5C) annotations for robot data improves the model's robustness to DS, generalization to unseen tasks, and language-following, with no statistically significant change in in-distribution performance. (2) Owing to greater linguistic diversity and richer descriptions of object-environment interactions, VLM-based annotations yield more substantial improvements on unseen tasks and language following compared to scripted annotations (Fig. 4B-C, 5B-C). (3) In two-phase co-training, incorporating these annotations during the 2nd-phase yields no additional benefit, suggesting that they are most effective when used exclusively during the 1st-phase (Fig. 4B-C, 5B-C). We posit that, because these annotations describe the same physical trajectories as the robot action data, their utility primarily lies in bootstrapping language-action alignment during the 1st-phase training, rather than introducing new information during the 2nd-phase.

Cross-embodiment Robot Data Co-training. (1) As shown in Fig. 4D and 5D, co-training with cross-embodiment robot data improves robustness to DS, generalization to unseen tasks, and language following, with no statistically significant change in in-distribution performance. (2) Cross-embodiment robot data is most effective, providing the largest gains, when confined to the first phase of two-phase co-training, particularly for unseen-task generalization and robustness under DS (Fig. 4D). When included during the second phase of training in the *two-phase full co-training*, it yields negligible additional benefit for language-following. We hypothesize that the diverse morphologies and manipulation strategies from cross-embodiment data are most valuable for learning generalizable visual and behavioral representations during the 1st-phase training. In the 2nd-phase, the model benefits from specializing to the target embodiment, at which point continued exposure to other embodiments provides limited value.

Human Video Co-training. *A) Latent Actions:* In *single-phase co-training*, the model jointly learns continuous actions for TRI-Ramen and discrete latent action tokens extracted from all video data (TRI-Ramen, OXE-Ramen, and human videos). For two-phase methods, the model learns latent actions from all video data in the 1st-phase. In *two-phase full co-training*, it learns both TRI-Ramen continuous actions and latent actions from all video data during the 2nd-phase.

Single-phase co-training with latent actions yields no im-

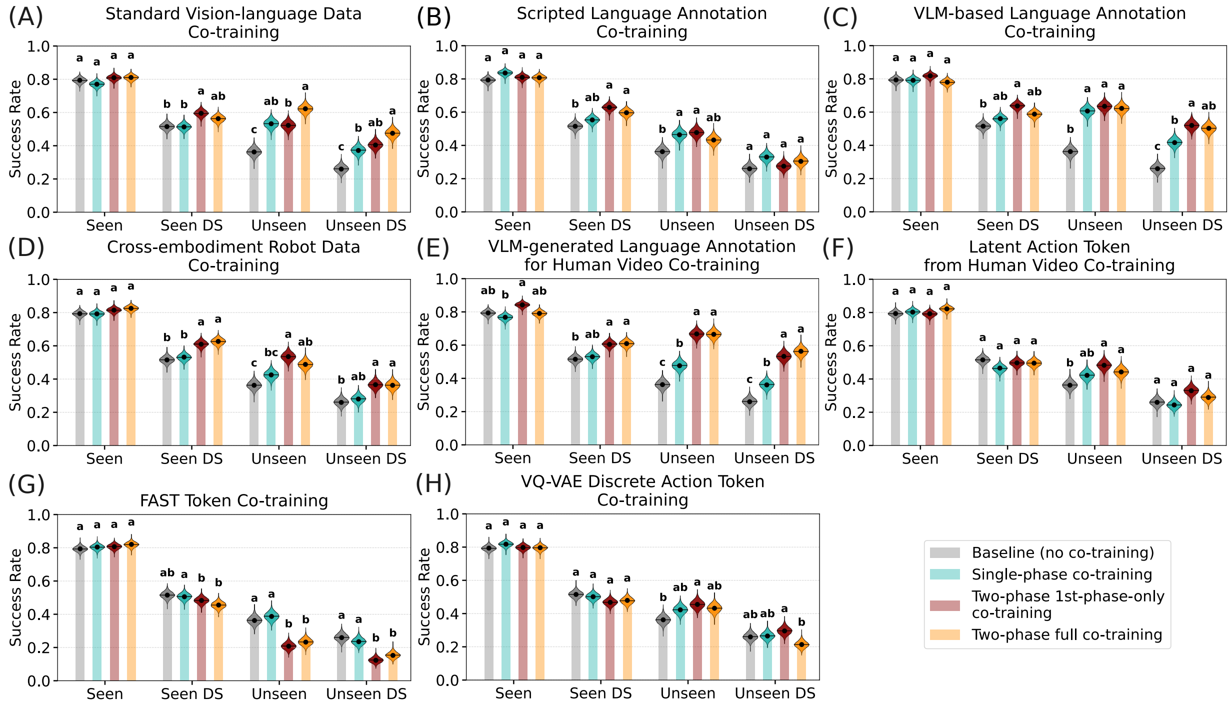


Fig. 4. **Simulation ablation of co-training data and strategies.** Comparison of the no-co-training baseline with policies co-trained on a single data modality across sequential training phases. Policies are evaluated on seen and unseen tasks under nominal and distribution shift conditions (A–H denote data modalities).

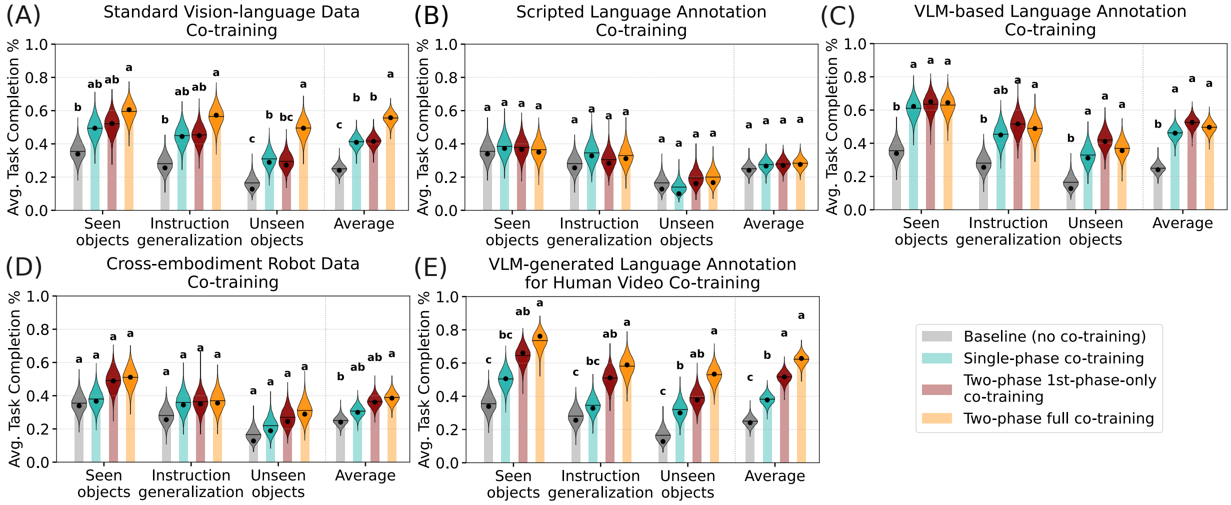


Fig. 5. **Real-world ablation of co-training data and strategies.** Performance of the no-co-training baseline and policies co-trained with a single data modality across training phases, evaluated at language-following with seen objects, instruction generalization, and unseen objects (A–E denote data modalities).

provement over the baseline (Fig. 4F), whereas *single-phase co-training* with other effective modalities (e.g., standard vision language data) consistently improves performance. On the other hand, for two-phase co-training: (1) latent action 1st-phase training improves performance on unseen tasks, and (2) incorporating latent actions during the 2nd-phase provides no additional benefit. These results suggest that the benefits of latent action 1st-phase training out of two phases may stem from increased computation rather than genuine knowledge transfer. To investigate this, we design a data- and compute-rich setting, comparing two policies: (a) Cross-embodiment co-training baseline: 1st-phase trained on all robot data (TRI-Ramen and OXE-Ramen), and 2nd-phase on TRI-Ramen; this baseline is equivalent to the *two-phase 1st-phase-only co-training* with cross-embodiment data without latent action co-training. (b) Latent action three-phase co-training: (i) train on latent actions from all video data, (ii) train on all robot data for continuous actions, and (iii) train on TRI-Ramen. Notably, the only difference between these two methods is that the latter includes an additional initial training phase on all video data. As shown in Fig. 6, the added initial latent action training phase provides no benefit in this data- and compute-

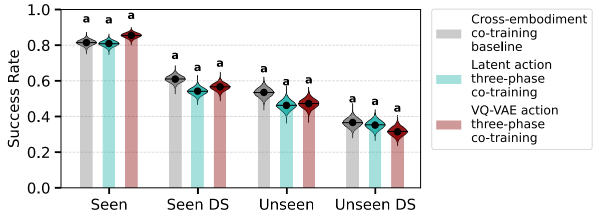


Fig. 6. **Ablation of latent action and VQ-VAE action co-training under data and compute-rich settings.** Simulation results indicate that these modalities show no performance gains over the cross-embodiment baseline.

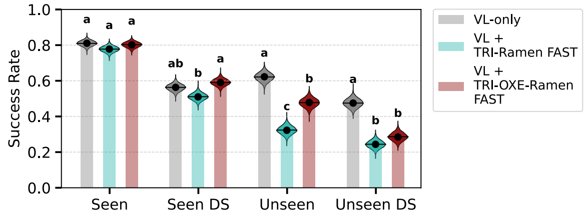


Fig. 7. **Ablation of FAST token co-training on a broad mix of robot and standard vision-language (VL) data.** FAST co-training along with VL data fails to improve overall performance and degrades generalization on unseen tasks compared to the VL-only co-training baseline.

rich setting.

Given that prior works [10, 5, 13] highlight the utility of latent action pretraining in a low target-robot-data regime, we further explore its effectiveness across varying scales of robot data, ranging from single-task to the full robot dataset. Fig. 8 shows that latent action 1st-phase training improves performance in the low target-robot-data regime, but these benefits diminish as the quantity of fine-tuning robot data increases.

B) VLM-generated Annotations: (1) As shown in Fig. 4E and 5E, co-training with VLM-generated annotations for human videos improves robustness to DS, generalization to unseen tasks, and language following, with no statistically significant change in in-distribution performance. (2) In two-phase co-training (Fig. 5E), continuing to incorporate these annotations during the 2nd-phase enhances language-following performance, particularly for unseen objects. We attribute this benefit to the rich diversity of motions, objects, and environments in human videos, which is absent from TRI-Ramen. Joint training during the 2nd-phase allows the model to maintain this broader world knowledge, rather than narrowing to the distribution of target robot data.

Discrete Robot Action Tokens Co-training. *A) FAST Tokens:* Our results suggest that FAST token co-training fails to improve performance across all dimensions and degrades generalization on unseen tasks (Fig. 4G). Prior works [33, 22] show that FAST token co-training could improve performance when pretrained on a broad mix of robot and standard VL data. To examine this claim, we compare three approaches: (a) VL + TRI-OXE-Ramen FAST: 1st-phase training on all robot data using FAST tokens, alongside VL data. (b) VL + TRI-Ramen FAST: 1st-phase training only on TRI-Ramen using FAST tokens, alongside VL data. (c) VL-only: 1st-phase training with VL data only. All methods employ an identical 2nd-phase: continuous action learning on TRI-Ramen through flow matching, with standard VL data co-training.

As illustrated in Fig. 7, FAST token co-training fails to improve overall performance and degrades generalization on unseen tasks. However, including OXE-Ramen during the 1st-phase significantly outperforms training on TRI-Ramen alone, indicating that FAST co-training might prove beneficial when scaled to substantially larger robot datasets, though it remains ineffective at our current data scale. We attribute this to the nature of FAST tokens as near-lossless action representations: co-training with FAST tokens may bias the VLM backbone toward learning precise action mappings rather than generalizable features.

B) VQ-VAE Discrete Action Tokens: (1) VQ-VAE discrete action token co-training yields marginal improvements on unseen tasks but slightly degrades performance under DS (Fig. 4H). (2) Incorporating VQ-VAE tokens during the 2nd-phase of two-phase co-training provides no additional benefit.

Similar to latent action co-training, we examine its effectiveness in a data- and compute-rich setting by comparing a cross-embodiment co-training baseline against VQ-VAE discrete action three-phase co-training policy that begins with learning VQ-VAE discrete action tokens on all robot data during the 1st-phase. As illustrated in Fig. 6, VQ-VAE co-training yields no improvement in this regime.

Summary. (1) Co-training with diverse VL data and cross-embodiment robot data substantially enhances the model's generalization to DS, unseen tasks, and language-following capabilities. Notably, owing to their information richness, co-training with standard VL data and language annotations for human videos benefits both 1st-phase and 2nd-phase co-training, whereas language annotations for robot trajectories and cross-embodiment data are primarily effective during 1st-phase in two-phase co-training. (2) Across all the effective co-training data modalities, standard VL data, VLM-based language annotations for robot data, and language annotations for human videos are the most beneficial. These three specific modalities are all in the form of diverse VL data, suggesting that strengthening VL understanding of the VLM backbone translates into better robot policies. (3) Discrete action tokens (including latent actions extracted from videos, FAST tokens, and action tokens learned from VQ-VAE) co-training yield no statistically significant performance improvements in our experiments. Specifically, co-training with FAST tokens decreases generalization, while latent actions from videos only provide benefits in the low target-robot-data regime, with benefits diminishing as the proportion of robot data increases. (4) Across all co-training modalities examined, we observe no statistically significant impact on in-distribution performance.

Fig. 9 compares the best training strategies for each useful co-training modality. Specifically, for standard VL and VLM-generated language annotations for human videos, the best models correspond to *two-phase full co-training*. For scripted and VLM-based language annotation and cross-embodiment robot data, the best models correspond to *two-phase 1st-phase-only co-training*. Co-training with standard VL data, VLM-

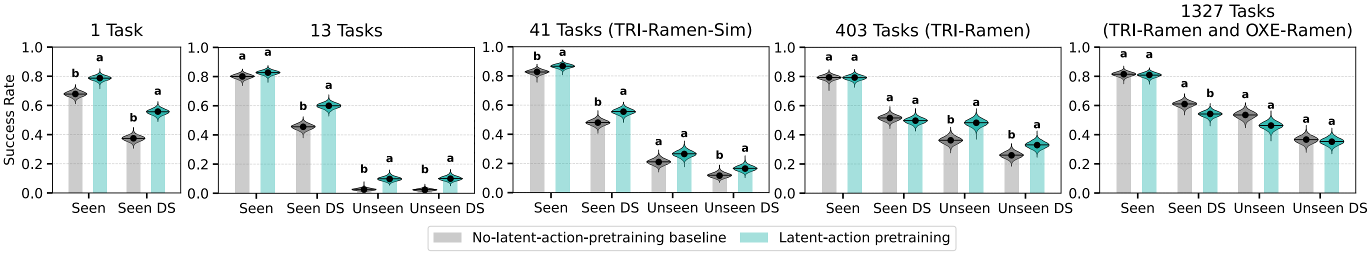


Fig. 8. **Ablation of latent action co-training on increasing data quantity.** Latent action 1st-phase in a multi-phase training yields performance gains in the low target-robot data regime, with diminishing returns as the quantity of fine-tuning robot data increases.

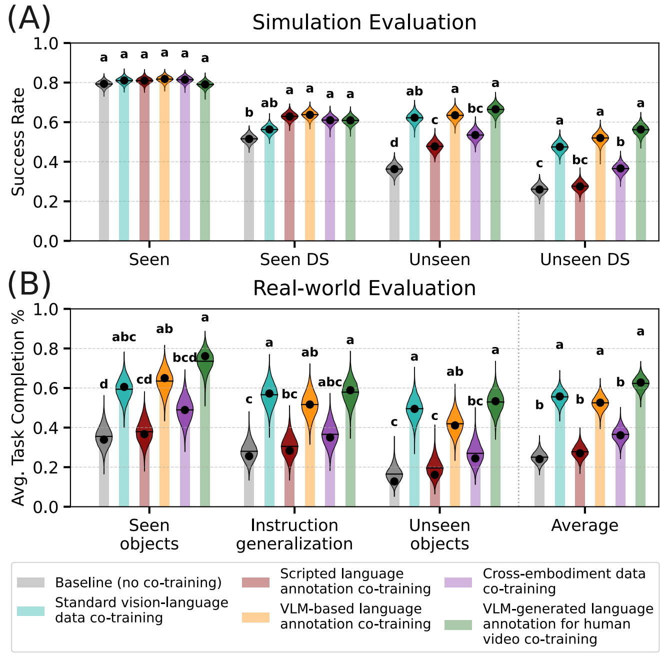


Fig. 9. **Performance of policies trained with the best co-training strategies for effective co-training data modalities.** A) Simulation results on seen and unseen tasks under nominal and distribution-shift conditions. B) Real-world language-following performance on seen objects, instruction generalization, and unseen objects. Diverse vision-language and cross-embodiment robot data substantially enhance the model's generalization to distribution shifts, unseen tasks, and language-following without affecting in-distribution performance.

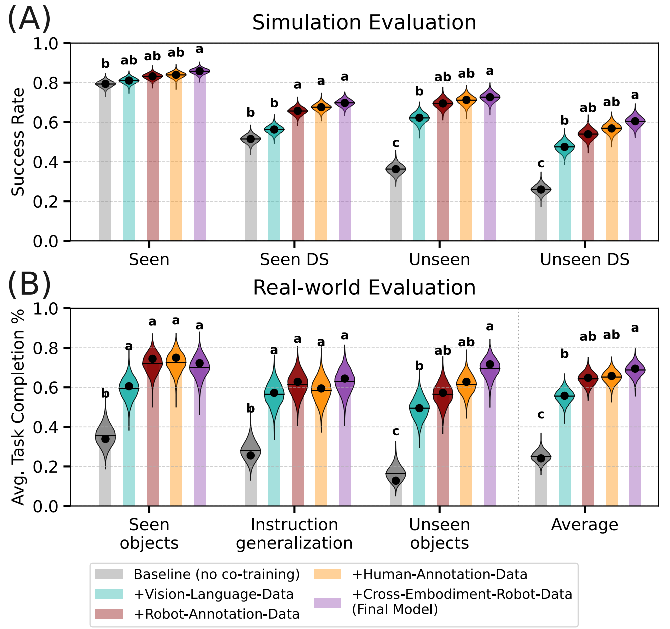


Fig. 10. **Performance of policies co-trained with the effective data modalities additively combined.** A) Simulation results on seen and unseen tasks under nominal and distribution shift conditions. B) Real-world evaluations for language-following performance across seen objects, instruction generalization, and unseen objects settings. Combining the effective co-training data modalities yields cumulative gains in policy performance.

based annotations for robot data, and annotations for human videos most substantially improves performance on unseen tasks and language following. Furthermore, benefiting from the rich information absent in robot demonstrations, co-training with standard VL data and annotations for human videos more effectively enables the model to recognize unseen objects compared to VLM-based annotations for robot data.

C. Does combining effective co-training modalities yield cumulative performance gains?

Having identified effective co-training data modalities along with their optimal training strategies, we further investigate whether combining these modalities yields cumulative benefits. We conduct an ablation study where we incrementally add each effective data source (training details are provided in Table S1): (1) Baseline without co-training, (2) +Vision-Language-Data: Standard VL data co-training only, (3) +Robot-Annotation-Data: Adding dense language annotations (both scripted and VLM-based) for robot data, (4) +Human-Video-Annotation-Data: Adding VLM-generated annotations for human videos, (5) +Cross-Embodiment-Robot-Data (namely, Final Model): Adding cross-embodiment robot data.

As shown in Fig. 10, combining effective co-training modalities yields consistent cumulative performance gains across all evaluation dimensions. Our Final Model achieves strong performance across all settings, attaining a 72.6% empirical success rate on simulation unseen tasks (36.4% improvement over baseline) and 69.4% empirical average task completion on real-world language following (45.3% improvement over baseline[1]).

[1] Improvement rates are calculated using the empirical means.

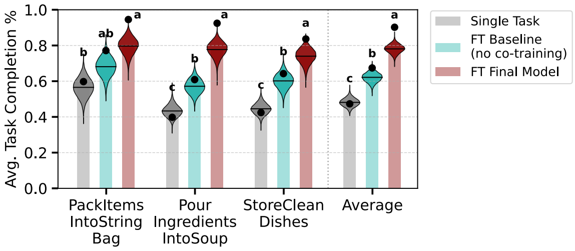


Fig. 11. **Adaptation to unseen long-horizon, dexterous tasks via fine-tuning.** Real-world evaluation of a Single Task policy trained from scratch, a fine-tuned (FT) Baseline pretrained only on robot data, and our FT Final Model pretrained with all effective co-training modalities. The FT Final Model achieves higher average task completion, highlighting the role of co-training in learning transferable, high-quality representations.

D. Can co-training enhance the quality of learned representations, thereby enabling rapid adaptation to unseen long-horizon, dexterous tasks?

Prior works [1, 6, 33] have demonstrated that high-quality pretraining enables policies to rapidly adapt to downstream unseen tasks through fine-tuning. We investigate whether co-training enhances learned representation quality by fine-tuning (FT) on our suite of unseen long-horizon, dexterous manipulation tasks. We compare three approaches: (1) FT Final Model: fine-tune our Final Model from Section III-C (trained with all effective co-training modalities), (2) FT Baseline: fine-tune the no-co-training baseline model (trained only on TRI-Ramen), (3) Single Task: a single-task policy without pretraining.

As shown in Fig. 11, benefiting from our curated co-training data and co-training strategies, our Final Model rapidly acquires new skills through fine-tuning, achieving 90.2% average task completion with only 200 demonstrations—a 22.8% improvement over the FT Baseline and a 42.9% improvement over the Single Task policy. This demonstrates that effective co-training substantially enhances representation quality, thereby enabling more fine-grained action learning in downstream tasks. We observe that failures in the FT Baseline and the Single Task policy predominantly stem from insufficient precision in manipulation: they frequently fail to align and secure the cap onto the bottle in *PackItemsIntoStringBag*, misalign the spatula for grasping in *PourIngredientsIntoSoup*, and struggle to grasp the transparent cup in *StoreCleanDishes*. In contrast, the FT Final Model consistently executes these fine-grained manipulations with high precision.

E. How do the effective co-training modalities shape the VLM backbone?

To investigate how the VLM backbone is shaped during co-training, we benchmark the VLMs extracted from our trained VLA policies on a suite of standard vision-language benchmarks using VLMEvalKit [23]. We evaluate policies trained with individual effective co-training modalities as well as the policy trained with all effective modalities additively combined across three complementary axes: semantic understanding and reasoning (MMBench, MME, SeedBench), spatial reasoning (RealWorldQA, GQA, SpatialEval), and planning and long-term reasoning (LEGO). We additionally report results for PaliGemma2-PT, the pretrained VLM used to initialize our policies, and PaliGemma2-Mix, a version of the backbone further fine-tuned for instruction following. These baselines contextualize how VLA training and co-training modify the backbone relative to its pretrained and instruction-tuned counterparts. Fig. 12 shows normalized performance across benchmarks, and we present unnormalized scores in Table S4. Several trends emerge: (1) The no-co-training baseline performs poorly across nearly all benchmarks. Compared to PaliGemma2-PT and PaliGemma2-Mix, it exhibits substantial degradation, losing all ability to generate language, indicating that training exclusively on robot data erodes the VLM backbone visiolinguistic understanding inherited from pretraining. (2) Co-training with standard vision-language data leads to strong improvements over the PaliGemma2-PT baseline across most benchmarks, particularly in spatial reasoning and real-world question answering. In contrast, other individual modalities yield no gains when used in isolation. (3) When effective co-training modalities are combined additively, the VLM backbone exhibits consistent, across-the-board improvements, outperforming both the no-co-training baseline and the pretrained PaliGemma2-PT model. The combined model achieves balanced gains across spatial, reasoning, and perception benchmarks, approaching or matching the performance of PaliGemma2-Mix, indicating a more robust and well-rounded multimodal representation. (4) The no-co-training baseline shows both degraded VLM benchmark performance and the weakest generalization in robot tasks, whereas policies co-trained with effective co-training modalities combined improve VLM benchmark scores and also generalize better under DS and to unseen tasks.

F. How does explicitly conditioning action generation on Chain-of-Thought (CoT) learned from co-training data affect policy performance?

We examine whether generating intermediate CoT traces to condition action generation provides any advantage over standard training. Specifically, we evaluate three co-training data types that naturally yield CoT-like intermediate contents: (1) scripted annotations for robot data, (2) VLM-based annotations for robot data, and (3) latent actions from videos.

For each co-training data type, we train three additional policies that differ in their CoT conditioning strategies. All three methods follow the same 1st-phase training procedure on the corresponding co-training data as described in Section II-A3. During the 2nd-phase, all methods jointly learn from both continuous robot actions and co-training data. Specifically: (1) 50%-CoT Training-Only: During the 1st-phase, the model conditions action generation on CoT with 50% probability. At inference time, actions are generated directly without CoT. (2) 50%-CoT with Inference: Training procedure is identical to (1). At inference time, the model first generates CoT, then conditions subsequent action generation on it. (3) 100%-CoT: In the 2nd-phase, the model always conditions action generation on CoT (100% probability). At inference time, the model generates CoT and conditions action generation on it.

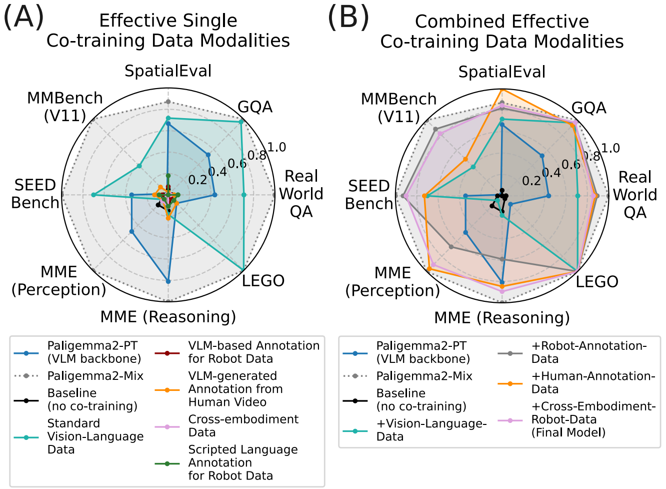


Fig. 12. **VLM backbone benchmarking for policies trained with the effective co-training data modalities.** A) Normalized performance of VLM backbones co-trained with individual effective data modalities compared to the no-co-training baseline, the pretrained PaliGemma2-PT used as backbone of all our policies, and the instruction-tuned PaliGemma2-Mix model, evaluated on standard vision-language, spatial, reasoning, and perception benchmarks. B) Performance of the VLM backbone when effective co-training data modalities are combined additively. Combining modalities yields consistent, cumulative improvements across all benchmarks, mirroring gains in downstream robot policy generalization.

We compare these explicit CoT strategies against the implicit two-phase co-training approaches discussed in Section II-A3, and the baseline policy trained without co-training. As shown in Fig. 13, while explicit CoT conditioning demonstrates improvements over the baseline in certain settings (e.g., co-training with VLM-based annotations for robot data on unseen tasks), it consistently fails to improve upon the implicit two-phase co-training approaches across all settings, with discernible performance degradation when using VLM-based annotations and latent actions as CoT sources. This lack of improvement can be attributed to the nature of our evaluation tasks. Prior works [44, 93, 67] demonstrating benefits from explicit CoT generation typically evaluate on tasks requiring multi-step planning or complex semantic reasoning. In contrast, our simulation benchmark focuses on manipulation tasks with clear objectives and immediate visual feedback, where the mapping from observation to action is relatively direct. In such settings, the implicit reasoning learned during co-training appears sufficient, making explicit CoT generation redundant.

The observed performance degradation, particularly pronounced with VLM-based annotations and latent actions, likely stems from two factors. First, these co-training data sources may contain inherent inaccuracies—VLM-generated annotations can misinterpret visual scenes or actions, while latent action models may capture irrelevant visual changes such as background variations rather than true action semantics. Second, these two sources produce richer and more complex CoT content compared to scripted annotations. When explicitly conditioning action generation on such CoT, any errors or imprecision in the generated CoT directly propagate to the subsequent action prediction, compounding the initial inaccuracies and leading to less precise manipulation behavior.

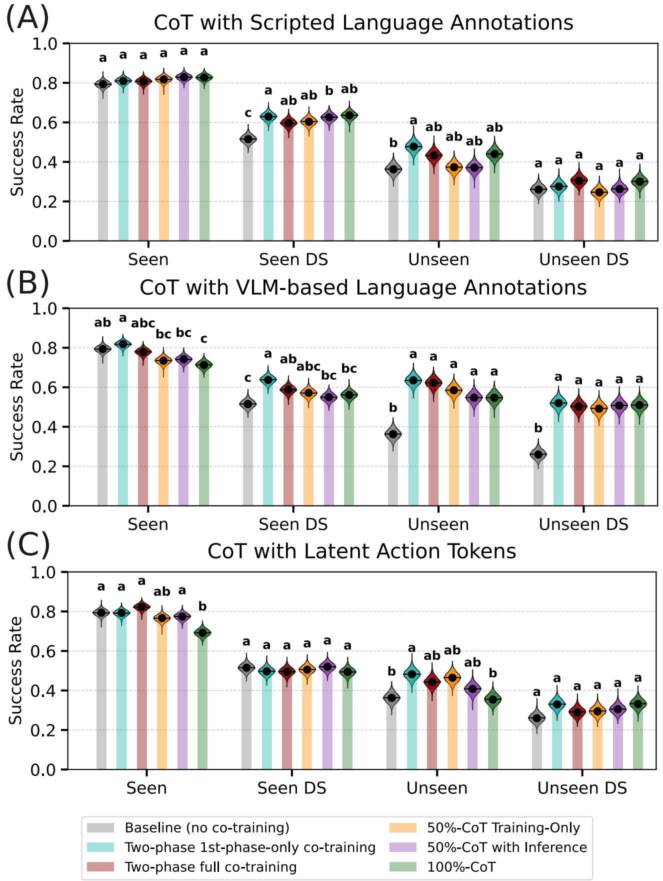


Fig. 13. **Simulation ablation of CoT-conditioned action generation.** Evaluation of policies trained to explicitly condition action generation on chain-of-thought (CoT) traces derived from A) scripted language annotations for robot data, B) VLM-based language annotations for robot data, and C) latent action tokens from videos. Action generation conditioned on CoT traces from co-training data yields no improvement in our simulation benchmark.

IV. Related Work

A. Large Behavior Models

Developing a general-purpose policy capable of perceiving, understanding, and acting in the physical world is a central objective in robotics. Early robot learning systems rely on task-specific policies trained on limited robot demonstrations [42, 24, 83], which constrain their ability to generalize beyond narrow training distributions. In contrast, LBMs scale imitation learning, both in model capacity and dataset size, and have shown impressive performance on dexterous tasks [1]. A prominent class within LBMs is VLAs [37, 100, 21, 6, 33, 22, 68, 67, 5, 80, 79], which integrate pretrained VLM [4, 65, 47, 73] backbones. Representative examples include VLMs paired with an action head as in [6, 33, 63] and fully autoregressive models [39, 58, 26]. Despite data scaling efforts, VLAs exhibit a limited generalization [1, 94, 97] to new objects, environments, and instructions compared to non-embodied foundation models like VLMs. This generalization gap primarily stems from the vast difference in training data availability [25, 55]: robot datasets remain orders of magnitude smaller than the internet-scale text and image corpora used for VLMs.

B. Co-training for Robot Learning

To bridge the gap between limited robot data and internet-scale multimodal resources, numerous studies have employed co-training with diverse data modalities.

Public VL datasets [14, 71, 89, 19, 91, 96], which are rich in commonsense knowledge and naturally compatible with VLA architectures, are widely adopted for co-training robot policies [33, 99, 98, 39]. Beyond public datasets, recent works [44, 84, 61, 93] construct embodied reasoning VL datasets directly on robot trajectories, incorporating rich planning and spatial information. These efforts show evidence that co-training with diverse vision-language data enhances generalization [33, 44] and improves VLM's learned representations [11, 84] for manipulation tasks.

Several works have explored using cross-embodiment robot data for co-training. Some efforts [2, 20, 85, 86, 7] train a single policy directly on multiple robot embodiment data, enabling a unified model to operate with diverse morphologies. [6, 33, 22, 50, 48] incorporate cross-embodiment data during pretraining to learn generalizable, embodiment-agnostic representations that are subsequently adapted to a specific target robot. [69, 1, 37, 38, 62] leverage the Open X-Embodiment dataset [55], a large-scale aggregation of demonstrations from many robot platforms, aiming to improve robustness and generalization.

Compared to robot data that typically requires teleoperation for collection, human videos offer a more scalable data source. A number of works [41, 52, 36, 40, 51] explicitly extract action labels (e.g., hand poses) from videos for policy co-training; however, obtaining accurate labels often requires additional sensing modalities, such as VR devices [60, 90] or wearable exoskeletons [81, 66]. Another line of research [9, 88, 13, 8, 12, 5] explores latent action representations by extracting discrete action tokens from video frames with methods such as VQ-VAE [72], which can serve as a unified representation across different embodiments that encode motion information. However, these approaches have been validated only in low target robot data regimes.

Discretizing continuous robot actions into tokens allows policies to treat action generation as a sequence modeling problem. Beyond the naive per-dimension binning approach [37, 100], advanced methods [58, 76] employ frequency-based techniques (e.g., FAST tokenizer) or vector quantization (e.g., VQ-VAE) to compress the action space. Notably, relying on discrete tokens for low-level control often results in limited precision [22] and slow inference [58, 39]. To mitigate this, recent approaches [33, 22, 35, 93] utilize these tokens solely for pre-training or co-training objectives, while retaining an action head for continuous action generation. This strategy has been shown to enhance sample efficiency and generalization.

C. Chain-of-thought for Robot Control

Inspired by the substantial benefits that CoT has brought to language models when dealing with complex tasks [29, 34, 77, 95], recent works [92, 33, 11, 43] have explored adapting CoT for robot control. Specifically, these approaches first generate intermediate content before conditioning action generation on it. This intermediate content can be linguistic, such as subtask decomposition or visual grounding information (e.g., object locations) [44, 93, 67, 30, 87], or control-centric, such as end-effector movements [39, 3] and latent actions [10, 13]. While CoT has been shown to be beneficial for long-horizon tasks or those requiring complex reasoning [44, 93, 67], there is limited empirical evidence comparing (i) explicitly conditioning the policy on predicted CoT traces, versus (ii) using the same CoT content solely as an auxiliary co-training objective. This evidence is especially insufficient for manipulation tasks with clear and well-defined goals.

V. Discussion, Limitations, And Future Work

We present a large-scale empirical study that systematically dissects the impact of diverse co-training data and strategies on the performance of LBMs. Our findings reveal that co-training with vision-language data and cross-embodiment robot data substantially enhances generalization to DS, unseen tasks, and language following capabilities, while discrete action token variants yield no statistically significant benefits. Furthermore, we show that combining effective modalities produces cumulative performance gains and enables rapid adaptation to dexterous, long-horizon tasks via fine-tuning.

Notably, among all useful co-training modalities, diverse vision-language data—including standard datasets and rich annotations for robot and human videos—demonstrate the most substantial improvements. This observation resonates with the Good Regulator Theorem [17], which states that a system must incorporate an internal model (implicit or explicit) of its operating world to effectively regulate it. In our setting, strong foundation models (such as VLMs) provide precisely such internal models that have rich semantic and spatial understanding of the physical world. Our results suggest that progress towards truly generalist robot policies is intrinsically linked to advances in these foundation models. Specifically, the VLM benchmarking results corroborate this interpretation: co-training with effective data modalities not only improves downstream robot performance but also preserves the visiolinguistic reasoning, spatial understanding, and perception capabilities of the VLM backbone itself. While this pattern is consistent across the models we evaluate, further investigation is needed to more rigorously characterize the relationship between backbone visiolinguistic understanding and policy generalization. Interestingly, we find that explicitly conditioning action generation on CoT learned from co-training data provides no benefit for our manipulation tasks, which have clear immediate objectives, suggesting that implicit reasoning learned during co-training suffices for such settings.

While our study provides promising insights, several limitations should be acknowledged. First, while we examine various sources of vision-language data, we do not systematically analyze their impact by task taxonomy (e.g., visual question answering, image captioning, object detection, spatial reasoning). Understanding how different vision-language task categories affect specific policy capabilities would enable more targeted and sample-efficient data curation. Second, we explore only coarse-grained representations for human videos through latent actions and language annotations. As hand pose estimation techniques advance and dexterous robotic hands continue to evolve, explicitly extracting fine-grained dexterous motions from human video may become a valuable co-training signal. Third, our exploration of CoT is limited to forms naturally arising from our co-training data—primarily low-level action abstractions lacking high-level planning or complex reasoning. Future work could investigate richer CoT formulations, such as history and reflection traces or hierarchical plans for tasks requiring complex decision-making. Finally, our study focuses exclusively on imitation learning; exploring co-training within alternative learning paradigms, such as world modeling or reinforcement learning, remains an open frontier for developing scalable generalist policies.

Acknowledgments

We thank Muhammad Zubair Irshad, Vitor Campagnolo Guizilini, Sedrick Keh, and Swati Gupta for the manuscript reviews; Kosei Tanada, Takuto Ito, Phoebe Horgan, Gordon Richardson, and Aykut Onol for evaluation quality assurance; Sam Shereda, Mariah Smith-Jones, and Matthew Tran for evaluation and data collection; Alejandro Castro and Joseph Masterjohn for simulation support, and to the TRI's PROPS team for assistance with experimental setup.

References

[1] Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *arXiv preprint arXiv:2507.05331*, 2025.

[2] Erik Bauer, Elvis Nava, and Robert K Katzschmann. Latent action diffusion for cross-embodiment manipulation. *arXiv preprint arXiv:2506.14608*, 2025.

[3] Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quon Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. Rt-h: Action hierarchies using language. *arXiv preprint arXiv:2403.01823*, 2024.

[4] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[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] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

[7] Nico Bohlinger, Grzegorz Czechmanowski, Maciej Krupka, Piotr Kicki, Krzysztof Walas, Jan Peters, and Davide Tateo. One policy to run them all: an end-to-end learning approach to multi-embodiment locomotion. *arXiv preprint arXiv:2409.06366*, 2024.

[8] Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

[9] Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2502.14420*, 2025.

[10] Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

[11] William Chen, Suneel Belkhale, Suvir Mirchandani, Oier Mees, Danny Driess, Karl Pertsch, and Sergey Levine. Training strategies for efficient embodied reasoning. *arXiv preprint arXiv:2505.08243*, 2025.

[12] Xiaoyu Chen, Junliang Guo, Tianyu He, Chuheng Zhang, Pushi Zhang, Derek Cathera Yang, Li Zhao, and Jiang Bian. Igor: Image-goal representations are the atomic control units for foundation models in embodied ai. *arXiv preprint arXiv:2411.00785*, 2024.

[13] Xiaoyu Chen, Hangxing Wei, Pushi Zhang, Chuheng Zhang, Kaixin Wang, Yanjiang Guo, Rushuai Yang, Yucen Wang, Xinquan Xiao, Li Zhao, et al. Villa-x: enhancing latent action modeling in vision-language-action models. *arXiv preprint arXiv:2507.23682*, 2025.

[14] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

[15] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau,

Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

[16] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[17] Roger C Conant and W Ross Ashby. Every good regulator of a system must be a model of that system. *International journal of systems science*, 1(2):89–97, 1970.

[18] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. The epic-kitchens dataset: Collection, challenges and baselines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11): 4125–4141, 2020.

[19] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. *arXiv e-prints*, pages arXiv–2409, 2024.

[20] Ria Doshi, Homer Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. *arXiv preprint arXiv:2408.11812*, 2024.

[21] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. *ICML23: Proceedings of the 40th International Conference on Machine Learning*, 2023.

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

[23] Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM international conference on multimedia*, pages 11198–11201, 2024.

[24] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference on robot learning*, pages 357–368. PMLR, 2017.

[25] Ken Goldberg. Good old-fashioned engineering can close the 100,000-year "data gap" in robotics, 2025.

[26] 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.

[27] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The "something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017.

[28] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18995–19012, 2022.

[29] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[30] Asher J Hancock, Xindi Wu, Lihan Zha, Olga Russakovsky, and Anirudha Majumdar. Actions as language: Fine-tuning vlms into vlas without catastrophic forgetting. *arXiv preprint arXiv:2509.22195*, 2025.

[31] Ryan Hoque, Peide Huang, David J Yoon, Mouli Sivapurapu, and Jian Zhang. Egodex: Learning dexterous manipulation from large-scale egocentric video. *arXiv preprint arXiv:2505.11709*, 2025.

[32] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[33] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

[34] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[35] Tao Jiang, Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Jianning Cui, Xiao Liu, Shuiqi Cheng, Jiyang Gao, Huazhe Xu, and Hang Zhao. Galaxea open-world dataset and g0 dual-system vla model. *arXiv preprint*

arXiv:2509.00576, 2025.

[36] Simar Kareer, Dhruv Patel, Ryan Punamiya, Pranay Mathur, Shuo Cheng, Chen Wang, Judy Hoffman, and Danfei Xu. Egomimic: Scaling imitation learning via egocentric video. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13226–13233. IEEE, 2025.

[37] 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.

[38] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025. *URL https://arxiv.org/abs/2502.19645*, 2025.

[39] Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, et al. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.

[40] Marion Lepert, Jiaying Fang, and Jeannette Bohg. Masquerade: Learning from in-the-wild human videos using data-editing. *arXiv preprint arXiv:2508.09976*, 2025.

[41] Marion Lepert, Jiaying Fang, and Jeannette Bohg. Phantom: Training robots without robots using only human videos. *arXiv preprint arXiv:2503.00779*, 2025.

[42] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39): 1–40, 2016.

[43] Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Raymond Yu, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, et al. Hamster: Hierarchical action models for open-world robot manipulation. *arXiv preprint arXiv:2502.05485*, 2025.

[44] Fanqi Lin, Ruiqian Nai, Yingdong Hu, Jiacheng You, Junming Zhao, and Yang Gao. Onetwovla: A unified vision-language-action model with adaptive reasoning. *arXiv preprint arXiv:2505.11917*, 2025.

[45] Juyi Lin, Amir Taherin, Arash Akbari, Arman Akbari, Lei Lu, Guangyu Chen, Taskin Padir, Xiaomeng Yang, Weiwei Chen, Yiqian Li, Xue Lin, David Kaeli, Pu Zhao, and Yanzhi Wang. Vote: Vision-language-action optimization with trajectory ensemble voting. *arXiv preprint arXiv:2507.05116*, 2025. URL https://arxiv.org/abs/2507.05116.

[46] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[47] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26296–26306, 2024.

[48] Jiaming Liu, Hao Chen, Pengju An, Zhuoyang Liu, Renrui Zhang, Chenyang Gu, Xiaoqi Li, Ziyu Guo, Sixiang Chen, Mengzhen Liu, et al. Hybridvla: Collaborative diffusion and autoregression in a unified vision-language-action model. *arXiv preprint arXiv:2503.10631*, 2025.

[49] Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022.

[50] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

[51] Yangcen Liu, Woo Chul Shin, Yunhai Han, Zhenyang Chen, Harish Ravichandar, and Danfei Xu. Immimic: Cross-domain imitation from human videos via mapping and interpolation. *arXiv preprint arXiv:2509.10952*, 2025.

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

[53] OpenAI. Gpt-5 is here, 2025. URL https://openai.com/gpt-5/.

[54] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[55] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[56] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[57] Baoqi Pei, Yifei Huang, Jilan Xu, Guo Chen, Yuping He, Lijin Yang, Yali Wang, Weidi Xie, Yu Qiao, Fei Wu, et al. Modeling fine-grained hand-object dynamics for egocentric video representation learning. *arXiv preprint arXiv:2503.00986*, 2025.

[58] 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.

[59] Hans-Peter Piepho. An algorithm for a letter-based representation of all-pairwise comparisons. *Journal*

of Computational and Graphical Statistics, 13(2):456–466, 2004.

[60] Ri-Zhao Qiu, Shiqi Yang, Xuxin Cheng, Chaitanya Chawla, Jialong Li, Tairan He, Ge Yan, David J Yoon, Ryan Hoque, Lars Paulsen, et al. Humanoid policy˜ human policy. *arXiv preprint arXiv:2503.13441*, 2025.

[61] Delin Qu, Haoming Song, Qizhi Chen, Zhaoqing Chen, Xianqiang Gao, Xinyi Ye, Qi Lv, Modi Shi, Guanghui Ren, Cheng Ruan, et al. Eo-1: Interleaved vision-text-action pretraining for general robot control. *arXiv preprint arXiv:2508.21112*, 2025.

[62] Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025.

[63] 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.

[64] David Snyder, Asher James Hancock, Apurva Badithela, Emma Dixon, Patrick Miller, Rares Andrei Ambrus, Anirudha Majumdar, Masha Itkina, and Haruki Nishimura. Is your imitation learning policy better than mine? policy comparison with near-optimal stopping. In *Proceedings of the Robotics: Science and Systems Conference (RSS) XXI*, 2025.

[65] Andreas Steiner, André Susano Pinto, Michael Tschannen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, Anthony Sherbondy, Shangbang Long, et al. Paligemma 2: A family of versatile vlms for transfer. *arXiv preprint arXiv:2412.03555*, 2024.

[66] Tony Tao, Mohan Kumar Srirama, Jason Jingzhou Liu, Kenneth Shaw, and Deepak Pathak. Dexwild: Dexterous human interactions for in-the-wild robot policies. *arXiv preprint arXiv:2505.07813*, 2025.

[67] Gemini Robotics Team, Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025.

[68] Gemini Robotics Team, S Abeyruwan, J Ainslie, JB Alayrac, MG Arenas, T Armstrong, A Balakrishna, R Baruch, M Bauza, M Blokzijl, et al. Gemini robotics: Bringing ai into the physical world, 2025. *URL https://arxiv. org/abs/2503.20020*, 2025.

[69] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[70] Russ Tedrake et al. Drake: Model-based design and verification for robotics, 2019.

[71] Peter Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Adithya Jairam Vedagiri IYER, Sai Charitha Akula, Shusheng Yang, Jihan Yang, Manoj Middepogu, Ziteng Wang, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *Advances in Neural Information Processing Systems*, 37:87310–87356, 2024.

[72] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[73] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[74] Xiaofeng Wang, Kang Zhao, Feng Liu, Jiayu Wang, Guosheng Zhao, Xiaoyi Bao, Zheng Zhu, Yingya Zhang, and Xingang Wang. Egovid-5m: A large-scale video-action dataset for egocentric video generation. *arXiv preprint arXiv:2411.08380*, 2024.

[75] Xin Wang, Taein Kwon, Mahdi Rad, Bowen Pan, Ishani Chakraborty, Sean Andrist, Dan Bohus, Ashley Feniello, Bugra Tekin, Felipe Vieira Frujeri, et al. Holoassist: an egocentric human interaction dataset for interactive ai assistants in the real world. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20270–20281, 2023.

[76] 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. *arXiv preprint arXiv:2507.01016*, 2025.

[77] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[78] Bernard L Welch. The generalization of ‘student's’problem when several different population varlances are involved. *Biometrika*, 34(1-2):28–35, 1947.

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

[80] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.

[81] Mengda Xu, Han Zhang, Yifan Hou, Zhenjia Xu, Linxi Fan, Manuela Veloso, and Shuran Song. Dexumi:

Using human hand as the universal manipulation interface for dexterous manipulation. *arXiv preprint arXiv:2505.21864*, 2025.

[82] Mingxing Xu, Wenrui Dai, Chunmiao Liu, Xing Gao, Weiyao Lin, Guo-Jun Qi, and Hongkai Xiong. Spatial-temporal transformer networks for traffic flow forecasting. *arXiv preprint arXiv:2001.02908*, 2020.

[83] Zhenjia Xu, Zhou Xian, Xingyu Lin, Cheng Chi, Zhiao Huang, Chuang Gan, and Shuran Song. Roboninja: Learning an adaptive cutting policy for multi-material objects. *arXiv preprint arXiv:2302.11553*, 2023.

[84] Ganlin Yang, Tianyi Zhang, Haoran Hao, Weiyun Wang, Yibin Liu, Dehui Wang, Guanzhou Chen, Zijian Cai, Junting Chen, Weijie Su, et al. Vlaser: Vision-language-action model with synergistic embodied reasoning. *arXiv preprint arXiv:2510.11027*, 2025.

[85] Jonathan Yang, Dorsa Sadigh, and Chelsea Finn. Polybot: Training one policy across robots while embracing variability. *arXiv preprint arXiv:2307.03719*, 2023.

[86] Jonathan Yang, Catherine Glossop, Arjun Bhorkar, Dhruv Shah, Quan Vuong, Chelsea Finn, Dorsa Sadigh, and Sergey Levine. Pushing the limits of cross-embodiment learning for manipulation and navigation. *arXiv preprint arXiv:2402.19432*, 2024.

[87] Shuai Yang, Hao Li, Yilun Chen, Bin Wang, Yang Tian, Tai Wang, Hanqing Wang, Feng Zhao, Yiyi Liao, and Jiangmiao Pang. Instructvla: Vision-language-action instruction tuning from understanding to manipulation. *arXiv preprint arXiv:2507.17520*, 2025.

[88] Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. *arXiv preprint arXiv:2410.11758*, 2024.

[89] Qiying Yu, Quan Sun, Xiaosong Zhang, Yufeng Cui, Fan Zhang, Yue Cao, Xinlong Wang, and Jingjing Liu. Capsfusion: Rethinking image-text data at scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14022–14032, 2024.

[90] Chengbo Yuan, Rui Zhou, Mengzhen Liu, Yingdong Hu, Shengjie Wang, Li Yi, Chuan Wen, Shanghang Zhang, and Yang Gao. Motiontrans: Human vr data enable motion-level learning for robotic manipulation policies. *arXiv preprint arXiv:2509.17759*, 2025.

[91] 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. *arXiv preprint arXiv:2406.10721*, 2024.

[92] 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.

[93] Andy Zhai, Brae Liu, Bruno Fang, Chalse Cai, Ellie Ma, Ethan Yin, Hao Wang, Hugo Zhou, James Wang, Lights Shi, et al. Igniting vlms toward the embodied space. *arXiv preprint arXiv:2509.11766*, 2025.

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

[95] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022.

[96] Enshen Zhou, Jingkun An, Cheng Chi, Yi Han, Shanyu Rong, Chi Zhang, Pengwei Wang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, et al. Roborefer: Towards spatial referring with reasoning in vision-language models for robotics. *arXiv preprint arXiv:2506.04308*, 2025.

[97] Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.

[98] Zhongyi Zhou, Yichen Zhu, Junjie Wen, Chaomin Shen, and Yi Xu. Chatvla-2: Vision-language-action model with open-world embodied reasoning from pretrained knowledge. *arXiv preprint arXiv:2505.21906*, 3, 2025.

[99] Zhongyi Zhou, Yichen Zhu, Minjie Zhu, Junjie Wen, Ning Liu, Zhiyuan Xu, Weibin Meng, Yaxin Peng, Chaomin Shen, Feifei Feng, et al. Chatvla: Unified multimodal understanding and robot control with vision-language-action model. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5377–5395, 2025.

[100] 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.

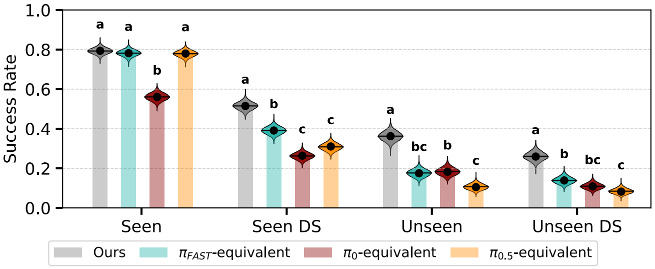


Fig. S1. Architecture ablation in simulation comparing our model to π_0-, $\pi_{0.5}$-, and π_{FAST}-equivalent variants across seen and unseen tasks, with and without distribution shift.

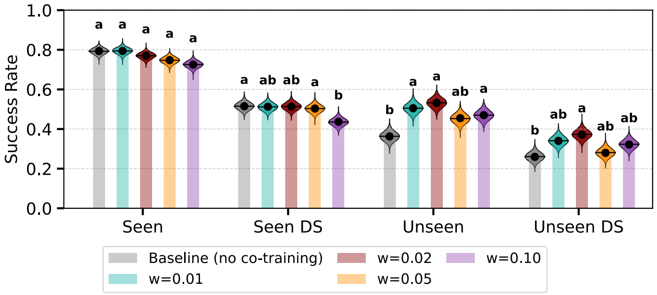


Fig. S2. Ablation of the cross-entropy loss weight w under *single-phase co-training* with standard vision-language data.

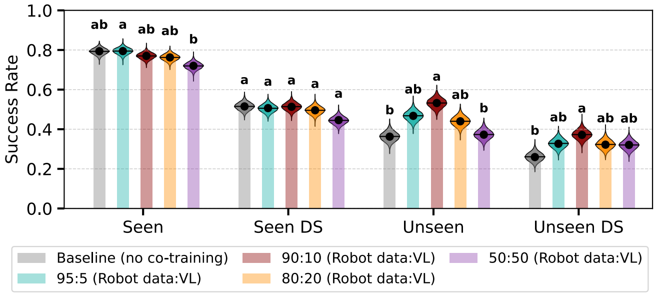


Fig. S3. Ablation of the robot-to-co-training data ratio per batch under *single-phase co-training* with standard vision-language data.

1. Model Architecture and Hyperparameter Ablation

A. *Model Architecture Ablation.*

We present ablation results comparing four model architectures: (1) Ours: Comprises a VLM and a flow transformer action head (ActionFT) (2) π_{FAST}-equivalent [58]: Consists solely of a VLM that auto-regressively generates FAST tokens, which are further decoded into continuous robot actions. (3) π_0-equivalent: Comprises a VLM and an action expert as formulated in [6]. All attention keys and values from all tokens across VLM layers are used as visual-linguistic representations for the action expert. (4) $\pi_{0.5}$-equivalent [33]: This is similar to π_0-equivalent with VLM and action expert, but the flow-matching timestep embedding is injected at each layer of the action expert, whereas in π_0-equivalent, the timestep embedding is injected only at the first layer together with the noised action. All four architectures use the same pretrained VLM (PaliGemma2-PT) and are trained on TRI-Ramen data with identical hyperparameters, except that π_0-equivalent and $\pi_{0.5}$-equivalent use batch size of 112 for 230k steps due to higher memory consumption, while Ours and π_{FAST}-equivalent use batch size of 128 for 200k steps. As shown in Fig. S1, while π_{FAST}-equivalent and $\pi_{0.5}$-equivalent achieve comparable in-distribution performance to Ours, they exhibit significantly degraded performance on distribution shifts and unseen tasks, demonstrating that our more compact representation enhances generalization. Additionally, π_0-equivalent substantially underperforms on in-distribution tasks, indicating the importance of injecting flow-matching timestep embeddings at each layer of the action network.

B. *Hyperparameter Ablation.*

We study two hyperparameters that are empirically critical for effective training: the weighting factor used to balance the flow-matching loss and the cross-entropy loss, and the ratio of target robot data to co-training data within each batch.

We analyze these hyperparameters under the vision-language data *single-phase co-training* setting. As shown in Figs. S2 and S3, increasing either the weighting factor or the co-training data ratio degrades in-distribution performance, while setting these parameters too low diminishes the generalization benefits from co-training. We adopt $w = 0.02$ and a 9:1 ratio of robot data to co-training data for co-training with modalities whose samples are not paired one-to-one with target robot data (i.e., standard vision-language data, cross-embodiment robot data, and human videos). This configuration preserves in-distribution performance while maximizing generalization improvements.

For modalities paired with target robot data, such as language annotations or discrete action tokens for robot data, the model learns from both the continuous actions and the corresponding co-training sample. Cross-embodiment robot data constitutes a special case, as the learning objective is still flow matching over robot actions; in this setting, target robot and cross-embodiment data are mixed during the 1st-phase training using a 6:4 batch ratio.

2. Training Details

Each phase of multi-task VLA policy training runs for 200k steps with a default batch size of 128, using a cosine learning rate schedule that decays from 2e-5 to 2e-6 over the last 60k steps. Each training phase requires approximately 64 hours on 16 H100 GPUs in this default setting. The ratios of robot data to co-training data follow Appendix 1B. Exceptions include: (1) the latent action 1st-phase training on all video data uses a batch size of 256 due to the large video volume, (2) for multi-modality co-training in Section III-C, to ensure the effective sample count per modality remains consistent with single-modality co-training policies, we increase batch size in certain training phases. The data modalities, data ratios, and batch sizes used at each training phase for these special VLA policies are listed in Table S1. For VLA policies trained or fine-tuned

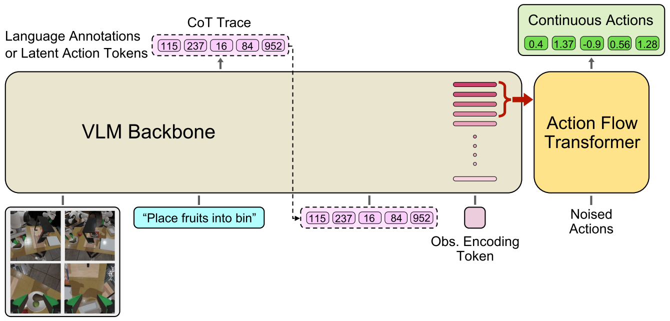


Fig. S4. Depiction of the training and inference process for the CoT-conditioned policy.

on a single task, we train for 30k steps and decay the learning rate from 2e-5 to 2e-6 with a cosine schedule over the last 24k steps. All other shared training hyperparameters are listed in Table S2.

3. Data Processing and Curation Details

A. *Dense Language Annotations for Robot Trajectories*

We derive scripted annotations by comparing the robot's end effector states at the current step and 16 steps into the future (matching the action chunk horizon). This comparison yields, for each gripper, the movement along the xyz axes, rotational changes in roll, pitch, and yaw, and variations in gripper width. We design the following template for both left and right grippers: [right/left] gripper moves [forward/backward] [right/left] [up/down], rotates [roll positive/negative] [pitch positive/negative] [yaw positive/negative], [opens/closes] We set thresholds of 2.5 cm for xyz movement, 20 degrees for rotation, and 1.3 cm for gripper width. When a change does not exceed its corresponding threshold, we omit the associated descriptor block from the annotation. For VLM-based annotations, we present the complete prompt provided to GPT-5 [53] in Fig. S5. Furthermore, Fig. S6 illustrates examples of scripted annotations and VLM-based annotations.

B. *Human Videos*

1) Latent actions: **Human Video Data Curation.** We utilize five publicly available egocentric human video datasets: Ego4D [28], EgoDex [31], Something-Something V2 [27], Epic Kitchen [18], and HoloAssist [75]. For Ego4D, Epic Kitchen, and HoloAssist, we filter videos using the provided language annotations with heuristic rules (e.g., maximum episode duration and removal of manipulation-irrelevant episodes by action verbs such as *look*, *walk*, *laugh*). For EgoDex and Something-Something V2, which are already structured as high-quality short clips with precise instructions, we use them directly without additional filtering. We also utilize reverse-playback annotations available for a subset of EgoDex videos. The total hours for each dataset are shown in Table S3.

Latent Action Model (LAM) Implementation.

The IDM is a 12-layer spatial-temporal transformer [82]. A Vector Quantization module (codebook size 32) maps continuous IDM outputs to 8 discrete latent tokens per time step. The visual FDM is a 12-layer spatial transformer [82]. ActionFDM is a lightweight convolutional decoder from tokens to robot action chunks. We train the LAM with batch size 1024 on 16 H100 GPUs for 300k steps (approximately 68 hours), using learning rate 1e-4 with 1k-step linear warmup and 20k-step cosine decay.

2) VLM-generated annotations: We utilize three human video datasets for annotation: Ego4D, EgoDex, and Something-Something V2. Representative annotation examples are shown in Fig. S7, with the number of annotated data samples summarized in Table S3. For Ego4D, we first filter the dataset by computing the intersection of video unique identifiers (UIDs) from two filtered subsets: EgoVid [74] and EgoHOD [57], ensuring that only videos retained by both filtering pipelines are included. We then segment the filtered videos into 4-second clips following EgoVid's splits, and downsample each clip at 1-second intervals. Moreover, Ego4D provides fine-grained action descriptions, which we pair with each frame as action priors. Consequently, our prompt to GPT-5 includes, for each clip: (i) the frames downsampled at 1-second intervals, (ii) action priors for each frame, (iii) instructions, and (iv) a reference image depicting the world-frame coordinate triad. The complete prompt is provided in Fig. S8. This yields a set of frame-level pairs, with one language annotation for each frame. To obtain accurate clip-level instructions, we perform a second prompting step where we prompt GPT-5 with the first and last frames of each clip, along with the complete list of generated annotations, to produce a clear and precise summary of the clip (prompt shown in Fig. S9).

For EgoDex and Something-Something V2, since neither dataset provides fine-grained action descriptions, we omit the frame-level action priors from the first prompting step. Additionally, both datasets already include high-quality clip-level instructions for each video, so we skip the second prompting step.

C. *Discrete Robot Action Tokens*

FAST Tokens. We directly apply the off-the-shelf FAST tokenizer [58] to TRI-Ramen without fine-tuning, which yields an average token length of 42.1 and a reconstruction error of 2.26e-4. We also experimented with fine-tuning the tokenizer on TRI-Ramen, which results in an average token length of 36.8 and a reconstruction error of 2.26e-4. Fine-tuning does not show substantial improvements. Hence, given that the off-the-shelf tokenizer is trained on a larger and more diverse robot dataset, which provides a more generalizable representation, we opt to use it without fine-tuning.

VQ-VAE Discrete Action Tokens. We employ a lightweight VQ-VAE architecture to compress action chunks into discrete tokens. The encoder and decoder are both compact networks: the encoder uses a sequence of 1D convolutional layers with residual blocks to map the action chunk

TABLE S1
DATA MODALITIES, BATCH DATA RATIOS, AND BATCH SIZES (BS) USED AT EACH TRAINING PHASE FOR THE SPECIAL POLICIES.

Policy	1st-phase	2nd-phase	3rd-phase
Three-phase latent action co-training	TRI-Ramen : OXE-Ramen : Human videos = 3 : 3 : 4; BS = 256	TRI-Ramen : OXE-Ramen = 6 : 4; BS = 128	TRI-Ramen : OXE-Ramen = 9 : 1; BS = 128
VL + TRI-OXE-Ramen FAST	TRI-Ramen (FAST) : Standard vision-language data : OXE-Ramen (FAST) = 5 : 2 : 3; BS = 128	TRI-Ramen : Standard vision-language data = 9 : 1; BS = 128	—
+Robot-Annotation-Data	Standard vision-language data : Language annotations for TRI-Ramen = 1 : 1; BS = 256	TRI-Ramen : Standard vision-language data = 9 : 1; BS = 128	—
+Human-Annotation-Data	Standard vision-language data : Language annotations for TRI-Ramen : Language annotations for human videos = 1 : 1 : 1; BS = 384	TRI-Ramen : Standard vision-language data : Language annotations for human videos = 9 : 0.5 : 0.5; BS = 128	—
+Cross-Embodiment-Robot-Data (Final Model)	Standard vision-language data : Language annotations for TRI-Ramen : Language annotations for human videos = 1 : 1 : 1; BS = 384	TRI-Ramen : Standard vision-language data : OXE-Ramen : Language annotations for human videos = 4 : 1 : 4 : 1; BS = 256	TRI-Ramen : Standard vision-language data : Language annotations for human videos = 9 : 0.5 : 0.5; BS = 128

TABLE S2
TRAINING HYPERPARAMETERS SHARED ACROSS ALL POLICIES.

Hyperparameter	Value
Image observation horizon	1
Camera number (TRI-Ramen)	4 (2 scene cameras, 1 wrist camera per robotic arm)
Image augmentation	Random crop (for TRI-Ramen, $256 \times 342 \rightarrow 224 \times 224$); color jitter (brightness $= 0.3$, contrast $= 0.4$, saturation $= 0.5$, hue $= 0.05$)
Learning rate warm-up steps	1000
Learning rate warm-up scheduler	linear
Learning rate decay scheduler	cosine decay

TABLE S3
SUMMARY OF THE HOURS AND NUMBER OF VLM-GENERATED ANNOTATION DATA SAMPLES FOR HUMAN VIDEO DATA.

Dataset	Video Hours	Annotation Data Samples
Ego4D	774.5 hours	5.2M
EgoDex	744.4 hours	3.0M
EgoDex (reversed)	455.7 hours	–
Something-Something V2	155.8 hours	0.8M
Epic Kitchen	60.4 hours	–
HoloAssist	80.8 hours	–
Total	**2271.6 hours**	**9.0M**

(horizon 16, dimension 20) into a lower-dimensional latent space, while the decoder mirrors the encoder's architecture to reconstruct the original action chunk from quantized embeddings. The vector quantizer maps each latent vector to its nearest entry in a learned codebook of size 32, discretizing the continuous action space into 8 tokens per action chunk. The model is trained using a combination of reconstruction loss, quantization loss, and commitment loss as described in [72].

4. POLICY DEPLOYMENT

We employ the same dual-arm Franka robot platform as in [1]. For simulation rollouts, the policy predicts a 16-step action chunk at each inference step and executes the first 8 steps in an open-loop manner before recomputing actions. During real-world deployment, we observe that discontinuities between predicted action chunks lead to jerky motions. To address this issue, we adopt a temporal ensemble technique. Specifically, the policy performs continuous inference (with an average latency of 0.146 seconds) to generate temporally overlapping action chunks. At each timestep, the actual action to be executed is computed as a uniform average of the corresponding actions from the four most recently predicted action chunks.

5. STATISTICAL ANALYSIS

To mitigate the risk of over-interpreting small empirical differences, we adopt the statistical analysis framework of [1] to (1) rigorously compare multiple co-training strategies via A/B testing and (2) quantify epistemic uncertainty using Bayesian analysis. For comparisons based on average task completion percentage, we use Welch's t-test [78]. For success rate comparisons, we use the STEP test [64], a state-of-the-art method for A/B testing of binary outcomes.

When comparing k strategies, we perform all $k(k-1)/2$ pairwise tests and control the FWER at 5% using Bonferroni correction. Final results are summarized using CLD [59], a label-based representation of A/B tests. Two CLD representations that do not share an English alphabet are statistically separated with at least 95% confidence. For example, if some strategy A, B, and C respectively have a, ab, and b as the corresponding CLDs, only the pair (A, C) is distinguished with a significant difference in performance. Lexicographical ordering of CLDs often aligns with empirical performance (e.g., A $>$ B $>$ C in this example), but this is not guaranteed when some pairwise differences are not statistically significant. See Fig. 4G ("Seen DS" column) for an illustrative case.

In addition to hypothesis testing, we report Bayesian uncertainty estimates for individual strategies. For success rate, we compute posterior distributions using a uniform Beta prior and observed success counts. For average task completion percentage, we use a uniform Dirichlet prior over task progress values. Posterior uncertainty is visualized as violin plots overlaid on bar charts in the main text (with dots and horizontal lines indicating empirical and posterior means, respectively), while raw empirical distributions are reported in Fig. S10, S11, S12, S13 for completeness (with dots only for empirical means).

6. Simulation Experiment Details

We visualize the 13 seen tasks and 8 unseen tasks in our simulation benchmark under both nominal and distribution shift conditions in Fig. S14, with each task represented by one snapshot showing the initial configuration. For both initial conditions and success criteria, we adopt the same specifications as in [1]. Specifically, for initial conditions, each rollout samples from a predefined distribution deterministically based on the simulation seed. We evaluate all the policies using identical seed sets to ensure fair comparison. As evident from the snapshots (Fig. S14), the scenarios are visually ambiguous; hence, policies cannot determine the task based purely on visual appearance and need language to disambiguate. For success criteria, each task is assigned a fixed time budget, and during execution, we monitor a set of task-specific predicates over the simulator state to determine whether the policy has successfully completed the task within the allocated time budget.

7. Real-world Experiment Details

A. Language Following.

We visualize all the initial conditions and corresponding instructions for the Seen Objects, Instruction Generalization, and Unseen Objects settings in Fig. S15 and Fig. S16. Task completion is evaluated based on four milestones:

1) The robot moves towards the target object for picking and reaches it within 10 cm.
2) The robot successfully grasps the target object for picking.
3) The robot moves the picked object to within 10 cm of the target destination.
4) The picked object is correctly positioned at the destination according to the instruction:
   - **In/On:** The picked object rests inside or on top of the destination.
   - **Next to:** The picked object is positioned within 10 cm of the destination.

B. Long-horizon Dexterous Manipulation.

We visualize the sample initial conditions for the three tasks in Fig. S17. A collage of video frames showing the task execution is displayed in Fig. S18, S19, S20. The task completion milestones for each task are defined as follows:

PackItemsIntoStringBag

- The left arm grasps the rim of the drawstring bag.
- The right arm grasps the opposite rim and opens the bag.
- The robot picks up the bottle.
- The robot picks up the bottle cap.
- The robot securely caps the bottle.
- The robot places the bottle inside the bag.
- The robot picks up the ball.
- The robot places the ball inside the bag.
- The robot picks up the sunglasses.
- The robot folds the first temple of the sunglasses.
- The robot folds the second temple of the sunglasses.
- The robot places the folded sunglasses inside the bag.
- The robot grasps the two drawstrings.
- The robot closes the bag by pulling the drawstrings.

PourIngredientsIntoSoup

- The robot grasps both handles of the soup pot.
- The robot places the pot onto the left burner of the stove.
- The right arm picks up the lid.
- The right arm places the lid on the table.
- The left arm picks up the carrot bowl and moves it on top of the pot.
- The right arm grasps the spatula.
- The robot pours the carrots into the pot using the spatula.
- The right arm places the spatula on the table.
- The left arm places the empty bowl on the table.
- The left arm picks up the mushroom bowl and moves it on top of the pot.
- The left arm pours the mushrooms into the pot and places the empty bowl on the table.
- The left arm picks up the cucumber bowl and moves it on top of the pot.
- The left arm pours the cucumbers into the pot and places the empty bowl on the table.
- The right arm picks up the lid.
- The right arm places the lid onto the pot.

StoreCleanDishes

- The robot opens the cabinet door.
- The robot picks up the bowl.
- The robot stacks the bowl onto the bowl inside the cabinet.
- The robot picks up the plate.
- The robot stacks the plate onto the plate inside the cabinet.
- The robot picks up the cup.
- The right arm passes the cup to the left arm.
- The robot stacks the cup into the cup on the cabinet.
- The robot picks up the glass.
- The right arm passes the glass to the left arm.
- The robot inserts the glass upside-down into the rack.

C. Experimental Procedure

All real-world evaluations follow a standardized and controlled experimental procedure to ensure fair comparison across policies. For each evaluation setting, we first generate a list of policy checkpoints to be evaluated. Checkpoints are selected sequentially from this list. For a given evaluation session, an initial condition (IC), including the set of objects and their poses, is sampled according to the task specification. The table and surrounding workspace are then manually configured to match this IC as closely as possible.

The selected policy checkpoint is executed once on the physical robot, after which the experimenter records task outcomes by following the rubrics. The evaluator then proceeds to the next checkpoint in the list, using the same initial condition. This process is repeated until all checkpoints have been evaluated on that IC, after which a new IC is sampled and the process restarts.

By evaluating all checkpoints under the same initial condition within a short time window, this protocol ensures that differences in performance are attributable to policy behavior rather than variations in hardware, lighting, or environmental setup. In effect, all checkpoints are evaluated under as nearly identical physical conditions as possible. Moreover, any slow and imperceptible changes in the conditions would equally affect all the checkpoints.

D. Rubric QA

Similar to [1], we conduct a quality assurance (QA) round to estimate the frequency of potential discrepancies due to human error or bias in the rubric evaluation of real-world rollouts. We sample 895 evaluation rollouts (31.6% of the total) and ask reviewers drawn from a separate pool than the original evaluators to review the rollout videos and their rubrics. To mitigate bias, the reviewers do not know which checkpoint is evaluated in each video. The QA overall task completion discrepancy, calculated as the average over all episodes and milestones, is 2.01%.

TABLE S4
VLM backbone evaluation results across vision-language benchmarks.

Model	**RealWorldQA**	**GQA**	**SpatialEval**	**MMBench**	**SEED**	**MME-P**	**MME-R**	**LEGO**
PaliGemma2-PT (VLA backbone)	0.24	31.68	0.26	0.01	0.24	701.56	242.50	0.03
PaliGemma2-Mix	0.55	59.81	0.34	0.68	0.70	1449.43	299.29	0.27
Baseline (no co-training)	0.02	0.06	0.02	0.00	0.03	193.75	44.29	0.01
Standard Vision Language Data Co-training (+Vision-Language-Data)	0.39	57.69	0.28	0.26	0.49	88.24	54.64	0.27
Scripted Language Annotation Co-training	0.05	0.00	0.07	0.00	0.06	80.06	32.14	0.02
VLM-based Language Annotation Co-training	0.02	0.06	0.03	0.00	0.04	92.00	31.07	0.01
Cross-embodiment Robot Data Co-training	0.01	0.01	0.01	0.00	0.01	114.02	33.21	0.00
VLM-generated Language Annotation for Human Video Co-training	0.03	0.02	0.01	0.07	0.09	105.43	65.36	0.03
+Robot-Annotation-Data	0.49	57.58	0.32	0.60	0.65	977.79	177.14	0.27
+Human-Annotation-Data	0.48	55.78	0.39	0.33	0.51	1397.43	253.57	0.27
+Cross-Embodiment-Robot-Data (Final Model)	0.47	58.02	0.33	0.56	0.63	1319.75	267.86	0.27

You are a precise video-frame annotator for a bimanual robot.

The attempted task is: *[Task Instruction]*

You will receive:
- A sequence of *{K}* video frames; each frame is immediately followed by a weak heuristic hint for that same frame.
- One additional reference image at the end showing the world coordinate triad that defines forward, backward, right, left, up, and down.

Your task:
For each of the *{K}* video frames (ignore the final reference image), produce one short imperative description of the most salient manipulation action visible in that frame.

Mandatory content for every annotation:
1) Each action needs to specify whether it‘s for the left gripper or the right gripper. When both grippers move simultaneously, you need to describe the action for each separately.
2) Mention the manipulated object when visible (e.g., ’red block‘, ’mug‘, ’screw‘, ’lid‘, ’tool‘). If unsure, use a generic noun such as ’object‘, ’block‘, ’handle‘, or ’container’.
3) Include a motion direction relative to the reference triad when motion is present: forward/backward/left/right/up/down; for rotations use ‘rotate’ (yaw positive is counterclockwise, negative is clockwise), ‘tilt’ (do not say roll, pitch, yaw).
4) You don‘t have to say rotation in every frame, say it when you think rotation is obvious, and sometimes rotation can be referred to an object, like tilt towards the cup to pour water.

Use the heuristic hint as a prior: add missing object names. Resolve left/right using the reference triad (world frame), not the camera view.

Style and constraints:
- Prefer verbs: approach, align, open the gripper, close the gripper, grasp, rotate, tilt, release, lift, lower, move forward/backward/left/right/up/down, push, pull, slide, insert, retract, place, hold.
- Do not mention the reference image, frames, camera, or any meta notes.
- Do not number items, do not add commentary, and do not use code fences.

Output format:
Return only a Python list containing exactly *{K}* strings (one per non-reference frame), using single quotes and commas, e.g., [’annot1‘, ’annot2‘, ...].
If your count does not match *{K}*, adjust before returning.

[image 1] [Scripted annotation 1]
[image 2] [Scripted annotation 2]
.
.
.
[image K] [Scripted annotation K]

And here is reference frame showing the frame of reference, telling you the direction of forward, backward, right, left, up, and down. You need to strictly follow this reference frame.

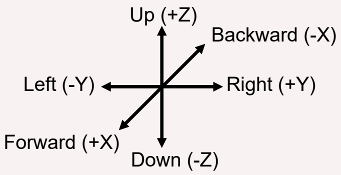
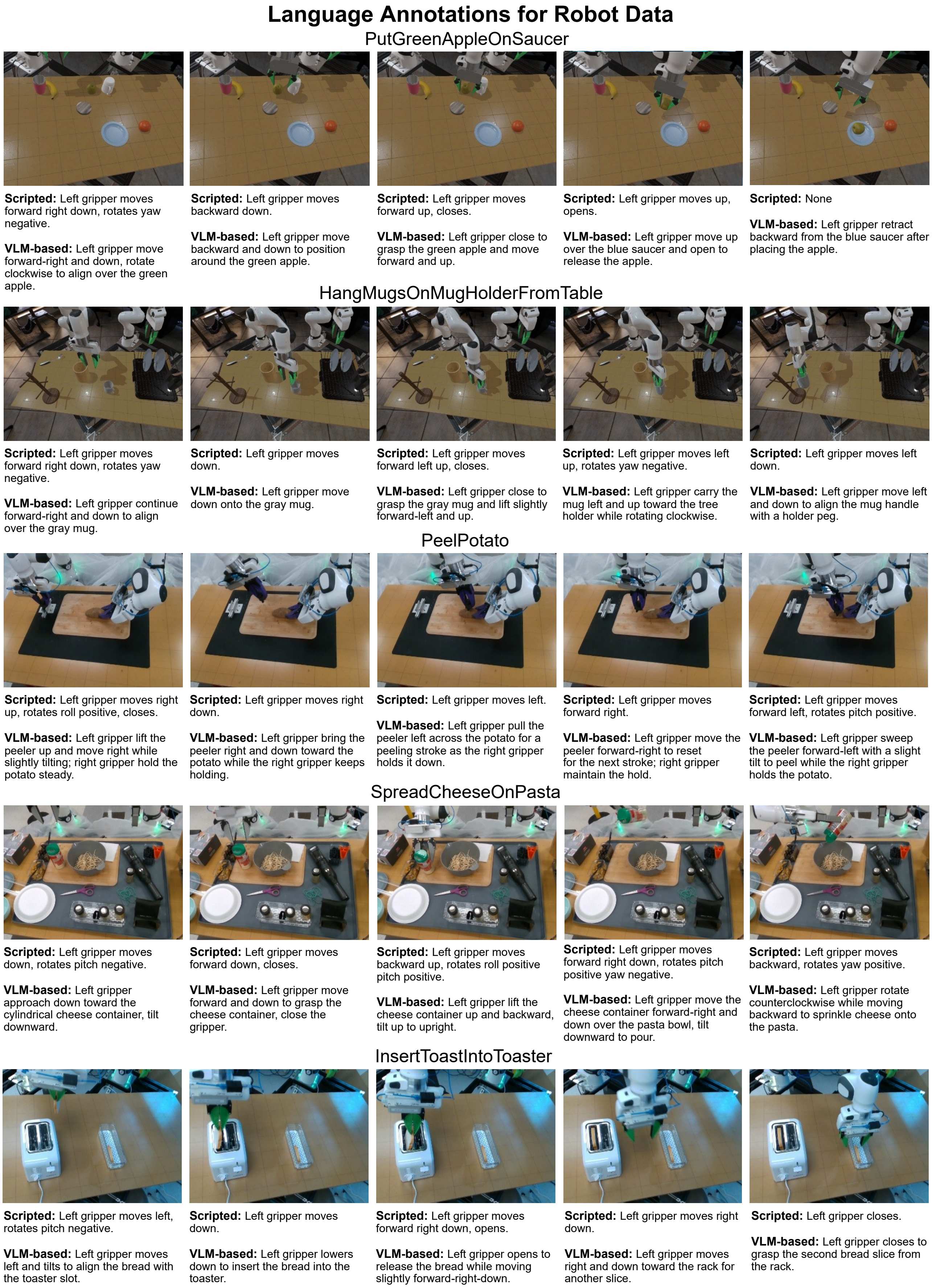


Fig. S5. The prompt used to generate VLM-based language annotations for robot data.



Language Annotations for Robot Data

PutGreenAppleOnSaucer

Scripted: Left gripper moves forward right down, rotates yaw negative.

VLM-based: Left gripper move forward-right and down, rotate clockwise to align over the green apple.

Scripted: Left gripper moves backward down.

VLM-based: Left gripper move backward and down to position around the green apple.

Scripted: Left gripper moves forward up, closes.

VLM-based: Left gripper close to grasp the green apple and move forward and up.

Scripted: Left gripper moves up, opens.

VLM-based: Left gripper move up over the blue saucer and open to release the apple.

Scripted: None

VLM-based: Left gripper retract backward from the blue saucer after placing the apple.

HangMugsOnMugHolderFromTable

Scripted: Left gripper moves forward right down, rotates yaw negative.

VLM-based: Left gripper continue forward-right and down to align over the gray mug.

Scripted: Left gripper moves down.

VLM-based: Left gripper move down onto the gray mug.

Scripted: Left gripper moves forward left up, closes.

VLM-based: Left gripper close to grasp the gray mug and lift slightly forward-left and up.

Scripted: Left gripper moves left up, rotates yaw negative.

VLM-based: Left gripper carry the mug left and up toward the tree holder while rotating clockwise.

Scripted: Left gripper moves left down.

VLM-based: Left gripper move left and down to align the mug handle with a holder peg.

PeelPotato

Scripted: Left gripper moves right up, rotates roll positive, closes.

VLM-based: Left gripper lift the peeler up and move right while slightly tilting; right gripper hold the potato steady.

Scripted: Left gripper moves right down.

VLM-based: Left gripper bring the peeler right and down toward the potato while the right gripper keeps holding.

Scripted: Left gripper moves left.

VLM-based: Left gripper pull the peeler left across the potato for a peeling stroke as the right gripper holds it down.

Scripted: Left gripper moves forward right.

VLM-based: Left gripper move the peeler forward-right to reset for the next stroke; right gripper maintain the hold.

Scripted: Left gripper moves forward left, rotates pitch positive.

VLM-based: Left gripper sweep the peeler forward-left with a slight tilt to peel while the right gripper holds the potato.

SpreadCheeseOnPasta

Scripted: Left gripper moves down, rotates pitch negative.

VLM-based: Left gripper approach down toward the cylindrical cheese container, tilt downward.

Scripted: Left gripper moves forward down, closes.

VLM-based: Left gripper move forward and down to grasp the cheese container, close the gripper.

Scripted: Left gripper moves backward up, rotates roll positive pitch positive.

VLM-based: Left gripper lift the cheese container up and backward, tilt up to upright.

Scripted: Left gripper moves forward right down, rotates pitch positive yaw negative.

VLM-based: Left gripper move the cheese container forward-right and down over the pasta bowl, tilt downward to pour.

Scripted: Left gripper moves backward, rotates yaw positive.

VLM-based: Left gripper rotate counterclockwise while moving backward to sprinkle cheese onto the pasta.

InsertToastIntoToaster

Scripted: Left gripper moves left, rotates pitch negative.

VLM-based: Left gripper moves left and tilts to align the bread with the toaster slot.

Scripted: Left gripper moves down.

VLM-based: Left gripper lowers down to insert the bread into the toaster.

Scripted: Left gripper moves forward right down, opens.

VLM-based: Left gripper opens to release the bread while moving slightly forward-right-down.

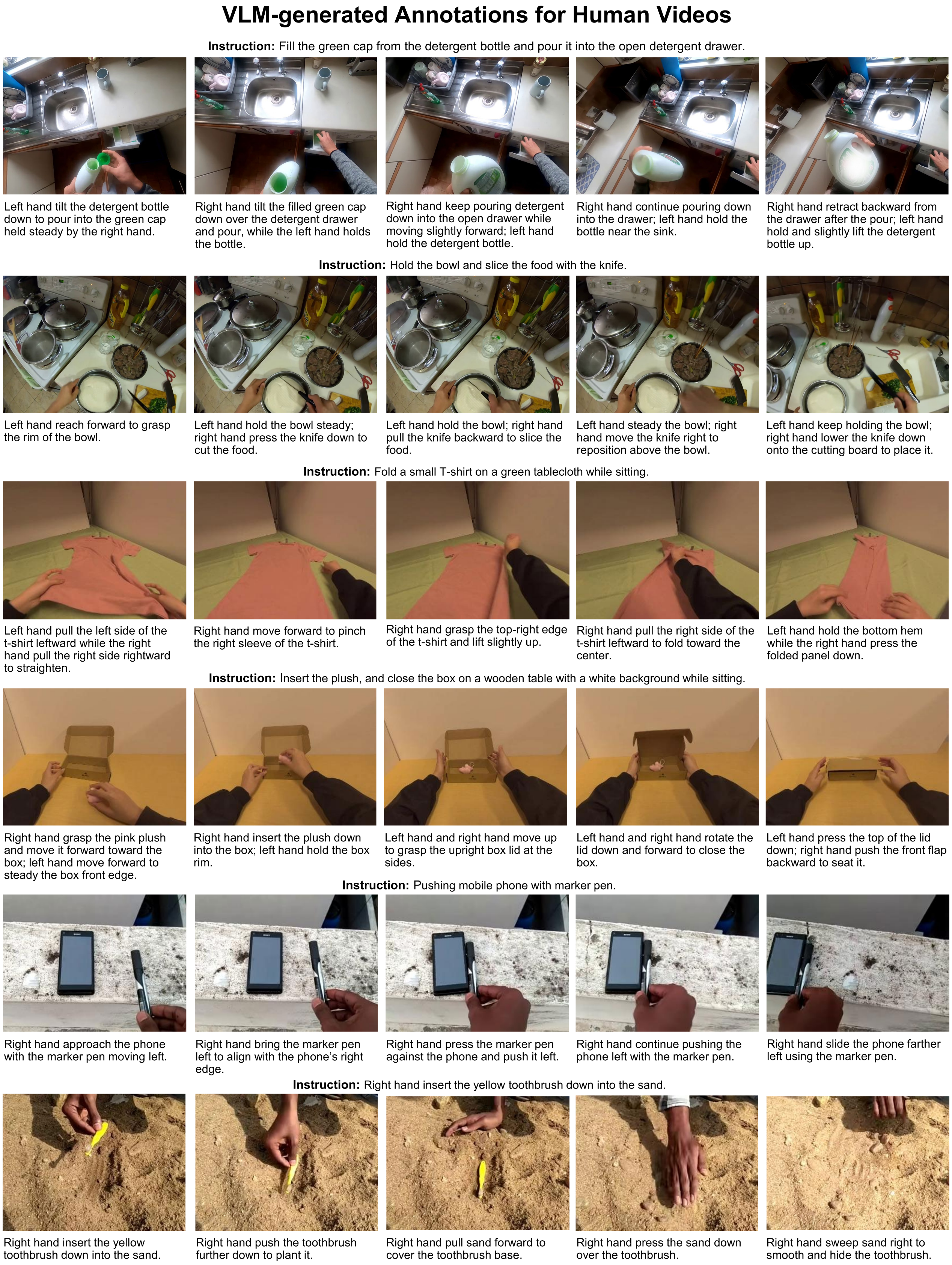
Scripted: Left gripper moves right down.

VLM-based: Left gripper moves right and down toward the rack for another slice.

Scripted: Left gripper closes.

VLM-based: Left gripper closes to grasp the second bread slice from the rack.

Fig. S6. Examples of scripted and VLM-based language annotations for robot data. Each row corresponds to a segment of an episode.



VLM-generated Annotations for Human Videos

Instruction: Fill the green cap from the detergent bottle and pour it into the open detergent drawer.

Left hand tilt the detergent bottle down to pour into the green cap held steady by the right hand.

Right hand tilt the filled green cap down over the detergent drawer and pour, while the left hand holds the bottle.

Right hand keep pouring detergent down into the open drawer while moving slightly forward; left hand hold the detergent bottle.

Right hand continue pouring down into the drawer; left hand hold the bottle near the sink.

Right hand retract backward from the drawer after the pour; left hand hold and slightly lift the detergent bottle up.

Instruction: Hold the bowl and slice the food with the knife.

Left hand reach forward to grasp the rim of the bowl.

Left hand hold the bowl steady; right hand press the knife down to cut the food.

Left hand hold the bowl; right hand pull the knife backward to slice the food.

Left hand steady the bowl; right hand move the knife right to reposition above the bowl.

Left hand keep holding the bowl; right hand lower the knife down onto the cutting board to place it.

Instruction: Fold a small T-shirt on a green tablecloth while sitting.

Left hand pull the left side of the t-shirt leftward while the right hand pull the right side rightward to straighten.

Right hand move forward to pinch the right sleeve of the t-shirt.

Right hand grasp the top-right edge of the t-shirt and lift slightly up.

Right hand pull the right side of the t-shirt leftward to fold toward the center.

Left hand hold the bottom hem while the right hand press the folded panel down.

Instruction: Insert the plush, and close the box on a wooden table with a white background while sitting.

Right hand grasp the pink plush and move it forward toward the box; left hand move forward to steady the box front edge.

Right hand insert the plush down into the box; left hand hold the box rim.

Left hand and right hand move up to grasp the upright box lid at the sides.

Left hand and right hand rotate the lid down and forward to close the box.

Left hand press the top of the lid down; right hand push the front flap backward to seat it.

Instruction: Pushing mobile phone with marker pen.

Right hand approach the phone with the marker pen moving left.

Right hand bring the marker pen left to align with the phone's right edge.

Right hand press the marker pen against the phone and push it left.

Right hand continue pushing the phone left with the marker pen.

Right hand slide the phone farther left using the marker pen.

Instruction: Right hand insert the yellow toothbrush down into the sand.

Right hand insert the yellow toothbrush down into the sand.

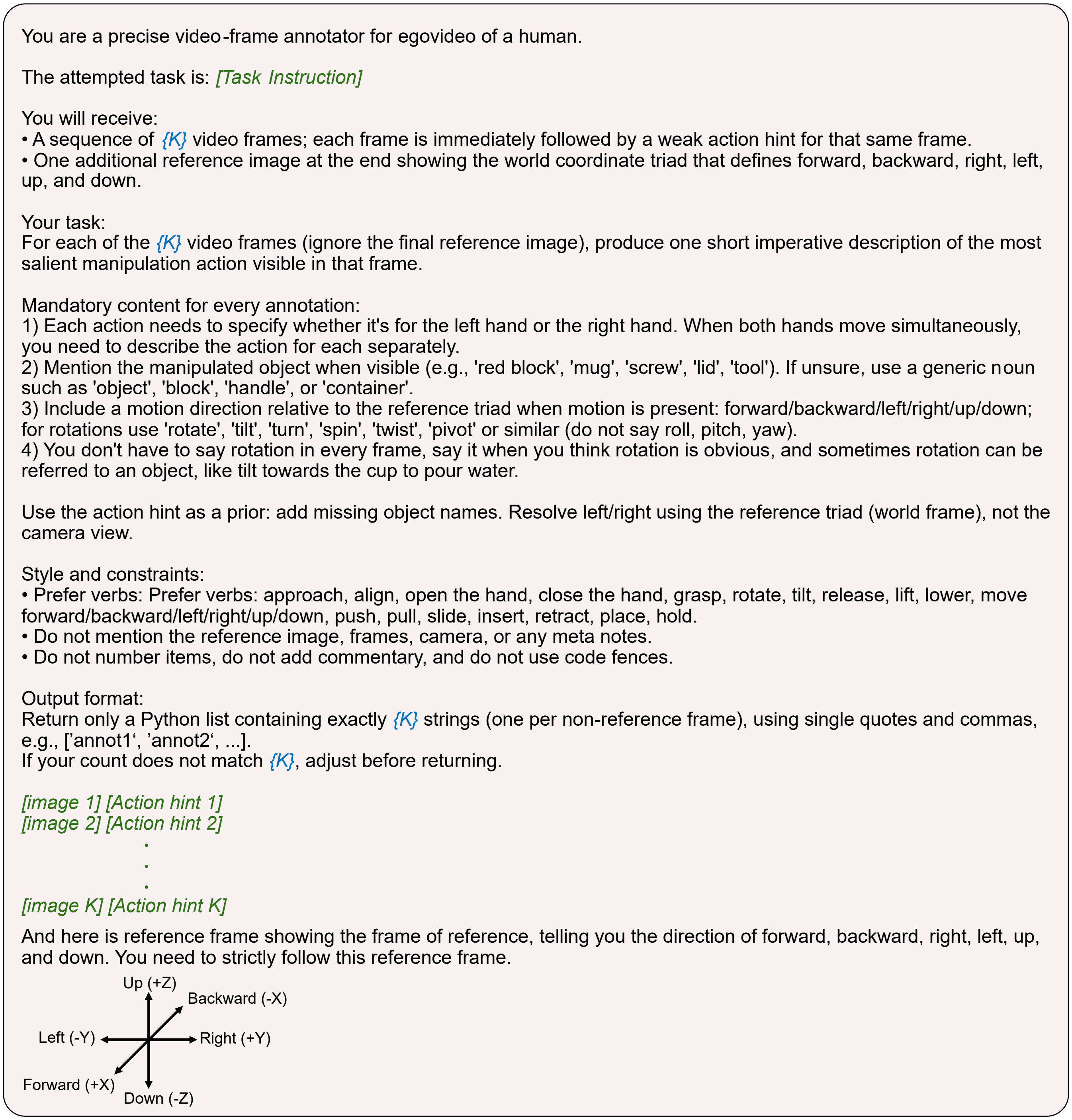
Right hand push the toothbrush further down to plant it.

Right hand pull sand forward to cover the toothbrush base.

Right hand press the sand down over the toothbrush.

Right hand sweep sand right to smooth and hide the toothbrush.

Fig. S7. Examples of VLM-generated annotations for human videos. Each row corresponds to a segment of a clip.



You are a precise video-frame annotator for egovideo of a human.

The attempted task is: *[Task Instruction]*

You will receive:
- A sequence of *{K}* video frames; each frame is immediately followed by a weak action hint for that same frame.
- One additional reference image at the end showing the world coordinate triad that defines forward, backward, right, left, up, and down.

Your task:
For each of the *{K}* video frames (ignore the final reference image), produce one short imperative description of the most salient manipulation action visible in that frame.

Mandatory content for every annotation:
1) Each action needs to specify whether it's for the left hand or the right hand. When both hands move simultaneously, you need to describe the action for each separately.
2) Mention the manipulated object when visible (e.g., 'red block', 'mug', 'screw', 'lid', 'tool'). If unsure, use a generic noun such as 'object', 'block', 'handle', or 'container'.
3) Include a motion direction relative to the reference triad when motion is present: forward/backward/left/right/up/down; for rotations use 'rotate', 'tilt', 'turn', 'spin', 'twist', 'pivot' or similar (do not say roll, pitch, yaw).
4) You don't have to say rotation in every frame, say it when you think rotation is obvious, and sometimes rotation can be referred to an object, like tilt towards the cup to pour water.

Use the action hint as a prior: add missing object names. Resolve left/right using the reference triad (world frame), not the camera view.

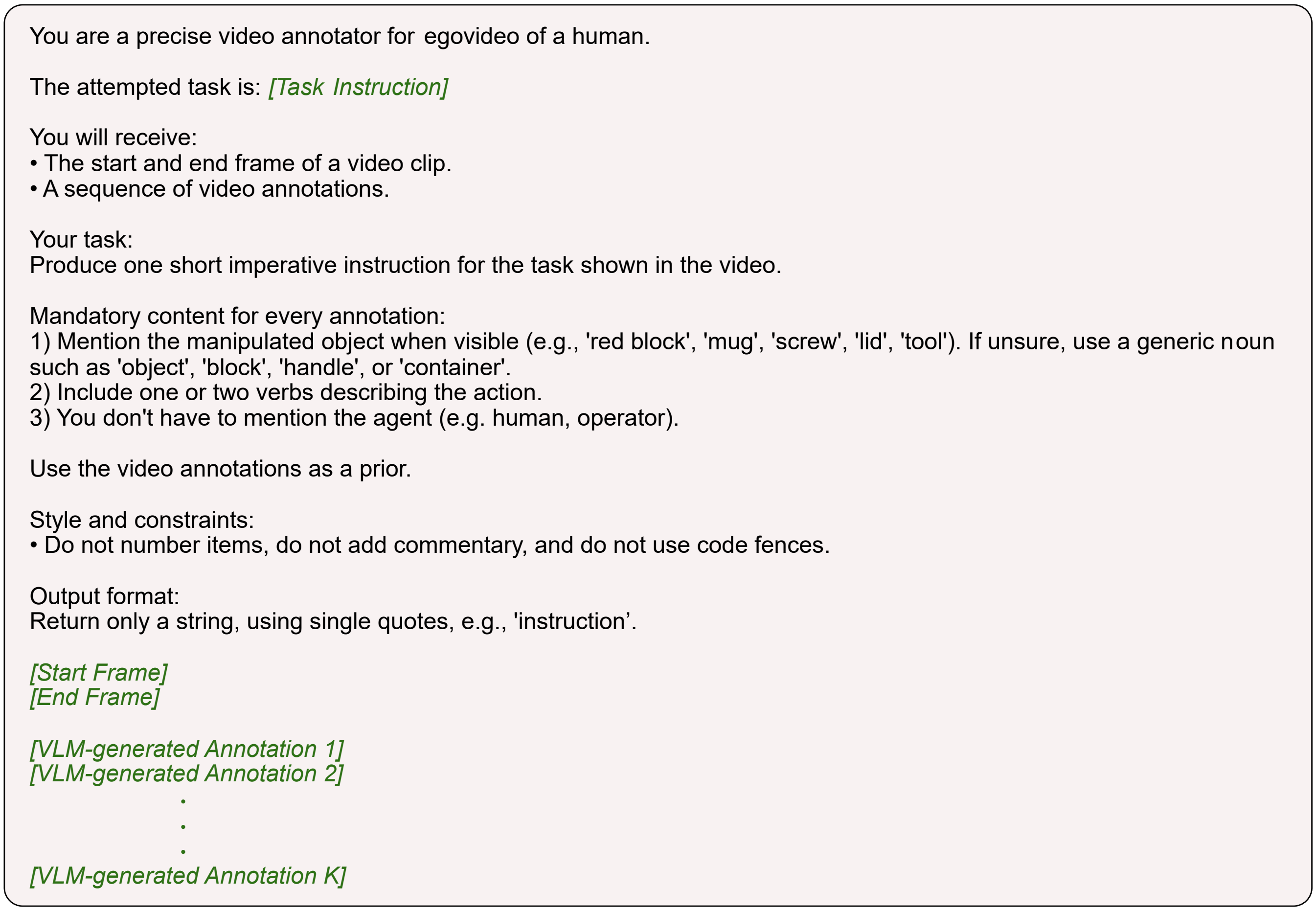
Style and constraints:
- Prefer verbs: Prefer verbs: approach, align, open the hand, close the hand, grasp, rotate, tilt, release, lift, lower, move forward/backward/left/right/up/down, push, pull, slide, insert, retract, place, hold.
- Do not mention the reference image, frames, camera, or any meta notes.
- Do not number items, do not add commentary, and do not use code fences.

Output format:
Return only a Python list containing exactly *{K}* strings (one per non-reference frame), using single quotes and commas, e.g., [’annot1‘, ’annot2‘, ...].
If your count does not match *{K}*, adjust before returning.

[image 1] [Action hint 1]
[image 2] [Action hint 2]
⋮
[image K] [Action hint K]

And here is reference frame showing the frame of reference, telling you the direction of forward, backward, right, left, up, and down. You need to strictly follow this reference frame.

Fig. S8. The prompt used for VLM-generated annotations for the Ego4D dataset.



You are a precise video annotator for egovideo of a human.

The attempted task is: *[Task Instruction]*

You will receive:
• The start and end frame of a video clip.
• A sequence of video annotations.

Your task:
Produce one short imperative instruction for the task shown in the video.

Mandatory content for every annotation:
1) Mention the manipulated object when visible (e.g., 'red block', 'mug', 'screw', 'lid', 'tool'). If unsure, use a generic noun such as 'object', 'block', 'handle', or 'container'.
2) Include one or two verbs describing the action.
3) You don't have to mention the agent (e.g. human, operator).

Use the video annotations as a prior.

Style and constraints:
• Do not number items, do not add commentary, and do not use code fences.

Output format:
Return only a string, using single quotes, e.g., 'instruction'.

[Start Frame]
[End Frame]

[VLM-generated Annotation 1]
[VLM-generated Annotation 2]
.
.
.
[VLM-generated Annotation K]

Fig. S9. The prompt used to generate an episode instruction for a video clip from the Ego4D dataset.

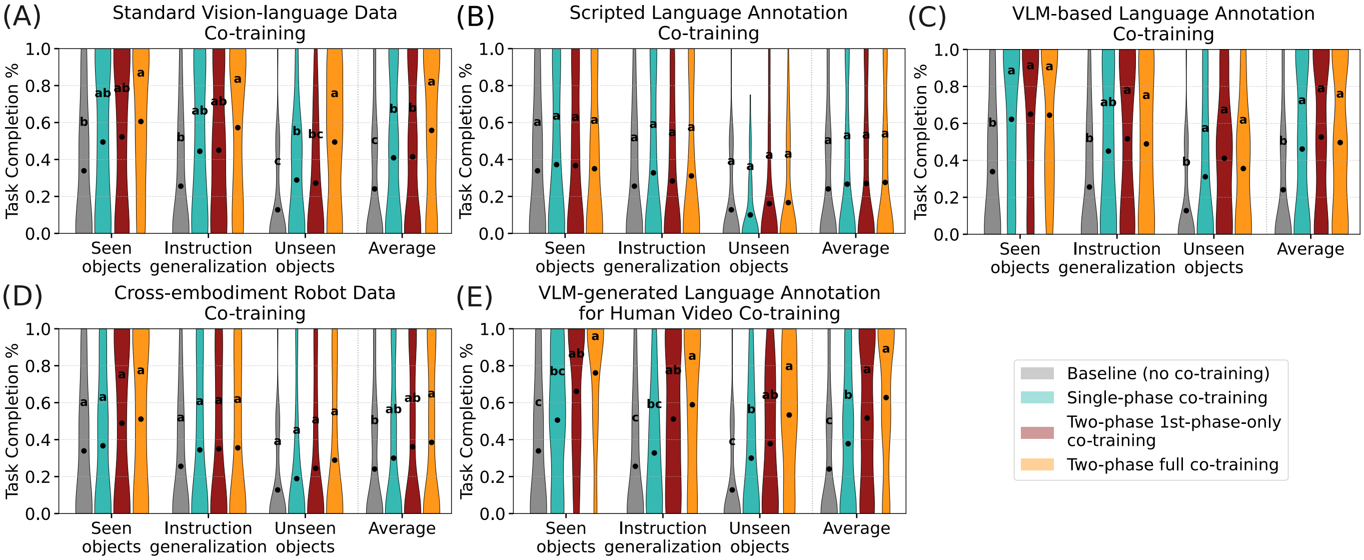


Fig. S10. Full distribution figures for the ablation of co-training data sources and strategies in real-world (Fig. 5).

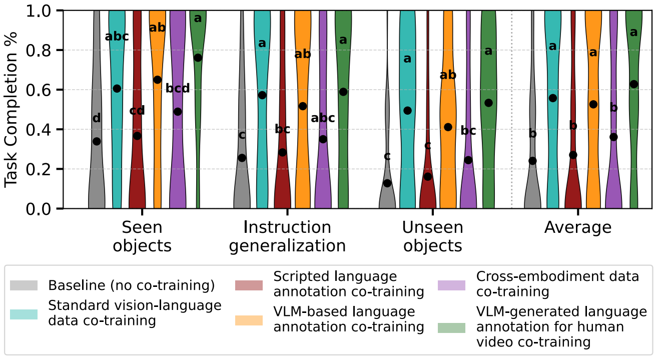


Fig. S11. Full distribution figure for the real-world performance of the policies trained with the best co-training strategy for all the effective co-training data modalities (Fig. 9B).

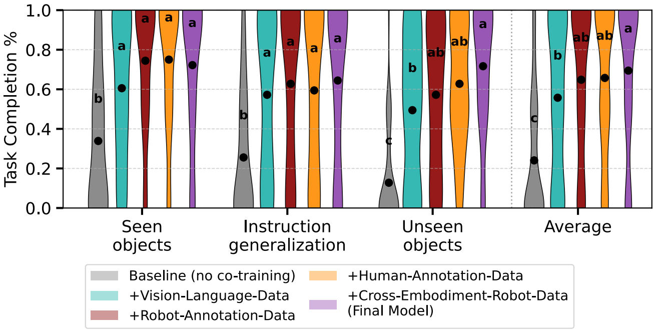


Fig. S12. Full distribution figure for the real-world the performance of the policy co-trained with all the effective data modalities combined (Fig. 10B).

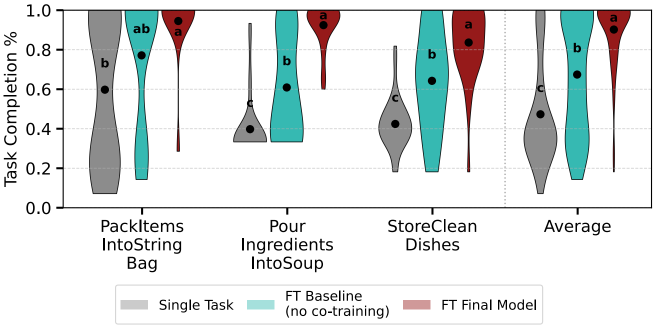


Fig. S13. Full distribution figure for the adaptation to unseen long-horizon, dexterous manipulation tasks via fine-tuning (Fig. 11).

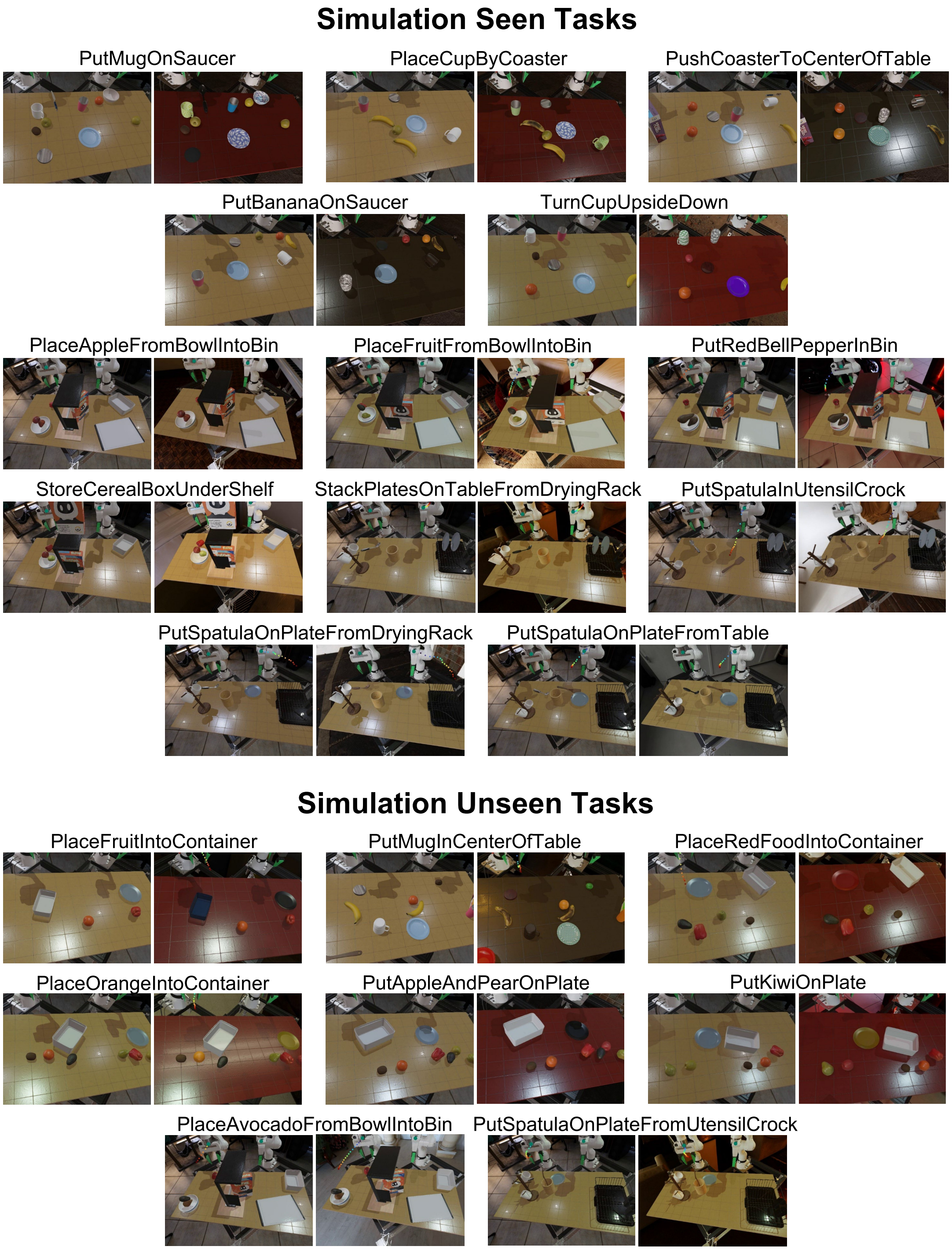
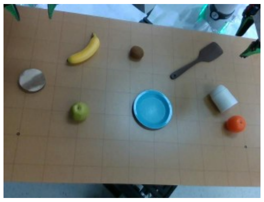


Fig. S14. Illustration of the 13 seen tasks and 8 unseen tasks used for evaluation in our simulation benchmark under both nominal and distribution shift conditions.

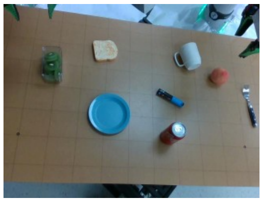
Language Following

Seen Objects & Instruction Generalization



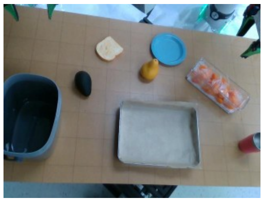
Seen Objects
1. pick up the banana and place it on the plate
2. pick up the apple and place it on the coaster
3. pick up the kiwi and place it next to the spatula

Instruction Generalization:
1. lift the yellow fruit and put it onto the plate
2. grab the green apple, which should then be placed on the coaster
3. take hold of the kiwi and position it beside the spatula



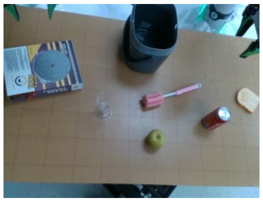
Seen Objects
1. pick up the kiwi slices and place them on the plate
2. pick up the pen and place it next to the fork
3. pick up the mug and place it next to the bread

Instruction Generalization:
1. move the slices of green fruit and place them on the plate
2. grab the writing tool, which should be then set next to the fork
3. move the mug so that it is next to the bread



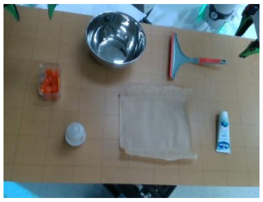
Seen Objects
1. pick up the pink cup and place it on the carrot slice box
2. pick up the avocado and place it in the trash bin
3. pick up the pear and place it on the baking pan

Instruction Generalization:
1. retrieve the cup that is pink and situate it onto the carrot box
2. take the black fruit that looks overripe and put it where the trash belongs
3. before baking, pick up the pear and position it on the pan



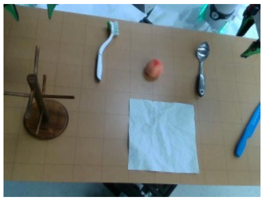
Seen Objects
1. pick up the wine glass and place it on the cereal box
2. pick up the can and place it in the trash bin
3. pick up the cleaning tool and place it next to the bread

Instruction Generalization:
1. take the fragile object used for wine and rest it over the cereal container
2. take the small metallic cylindrical shaped object and drop it in the rubbish bin
3. pick up the object used for brushing the toilet and place it close to the baked item



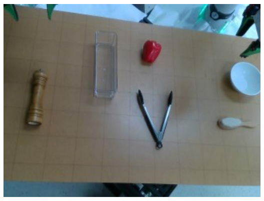
Seen Objects
1. pick up the carrot slices and place them in the bowl
2. pick up the baby cup and place it on the parchment paper
3. pick up the cream and place it next to the cleaning tool

Instruction Generalization:
1. grab the vegetables and put them inside the bowl
2. raise the little container meant for kids and set it on the cooking paper
3. hold the tube and lay it close to the tool for cleaning



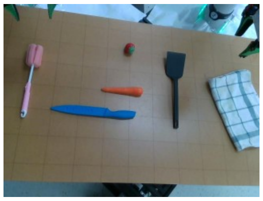
Seen Objects
1. pick up the peach and place it next to the drying rack
2. pick up the brush and place it on the towel
3. pick up the spoon and place it next to the knife

Instruction Generalization:
1. lift the fruit and place it next to the wooden structure
2. grasp the tool meant for scrubbing and rest it atop the towel
3. hold the utensil used for scooping and set it beside the cutting tool



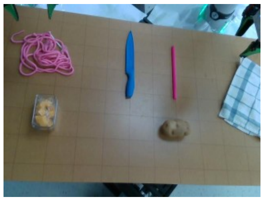
Seen Objects
1. pick up the tong and place it next to the brush
2. pick up the red bell pepper and place it in the bowl
3. pick up the pepper shaker and place it in the bin

Instruction Generalization:
1. lift the metal gripping tool and set it beside the item with bristles
2. move the food into the bowl
3. retrieve the spice jar and dispose of it properly in the bin



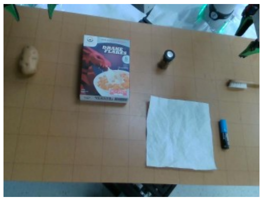
Seen Objects
1. pick up the knife and place it on the kitchen utensil
2. pick up the strawberry and place it on the towel
3. pick up the carrot and place it next to the cleaning tool

Instruction Generalization:
1. lift the cutting tool and rest it on top of the eating utensil
2. hold the soft red berry and put it on the towel for drying
3. pick up the vegetable with a tapered end and place it near the cleaning device



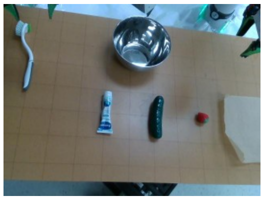
Seen Objects
1. pick up the potato and place it on the towel
2. pick up the straw and place it next to the rope
3. pick up the cheese container and place it next to the knife

Instruction Generalization:
1. lift the vegetable and set it on the cloth
2. pick up the thin plastic tube and set it near the rope
3. move the plastic box to the next to the blade



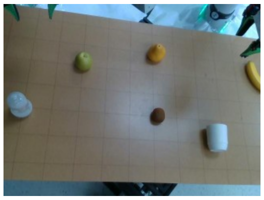
Seen Objects
1. pick up the potato and place it on the cereal box
2. pick up the spice jar and place it on the paper
3. pick up the brush and place it next to the pen

Instruction Generalization:
1. hold the potato-shaped item and lay it over the box of corn flakes
2. after picking up the bottle, set it carefully on the paper
3. lift the object used for brushing and lay it close to the writing tool



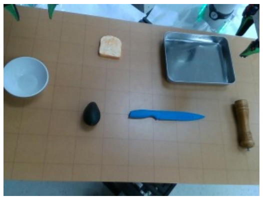
Seen Objects
1. pick up the brush and place it next to the cream
2. pick up the cucumber and place it in the bowl
3. pick up the strawberry and place it on the parchment paper

Instruction Generalization:
1. grasp the handle with soft bristles and put it by the tube of cream
2. retrieve the fresh vegetable and move it into the bowl
3. hold the juicy red berry and rest it on the parchment layer



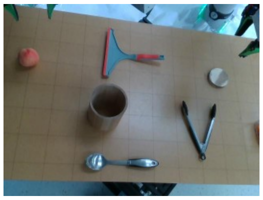
Seen Objects
1. pick up the baby cup and place it next to the apple
2. pick up the kiwi and place it next to the pear
3. pick up the mug and place it next to the banana

Instruction Generalization:
1. take the cup designed for infants and put it next to the apple
2. take hold of the kiwi-shaped item and put it next to the pear
3. grab the drinking mug and move it beside the banana



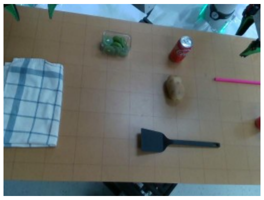
Seen Objects
1. pick up the avocado and place it in the bowl
2. pick up the knife and place it on the bread slice
3. pick up the pepper shaker and place it in the pan

Instruction Generalization:
1. pick up the overripe fruit and dispose of it in the white container
2. hold the item used for slicing and put it over the piece of bread
3. grasp the spice container and set it down in the metal rectangular object



Seen Objects
1. pick up the peach and place it in the utensil crock
2. pick up the tongs and place it next to the cleaning tool
3. pick up the carrot slices and place them on the coaster

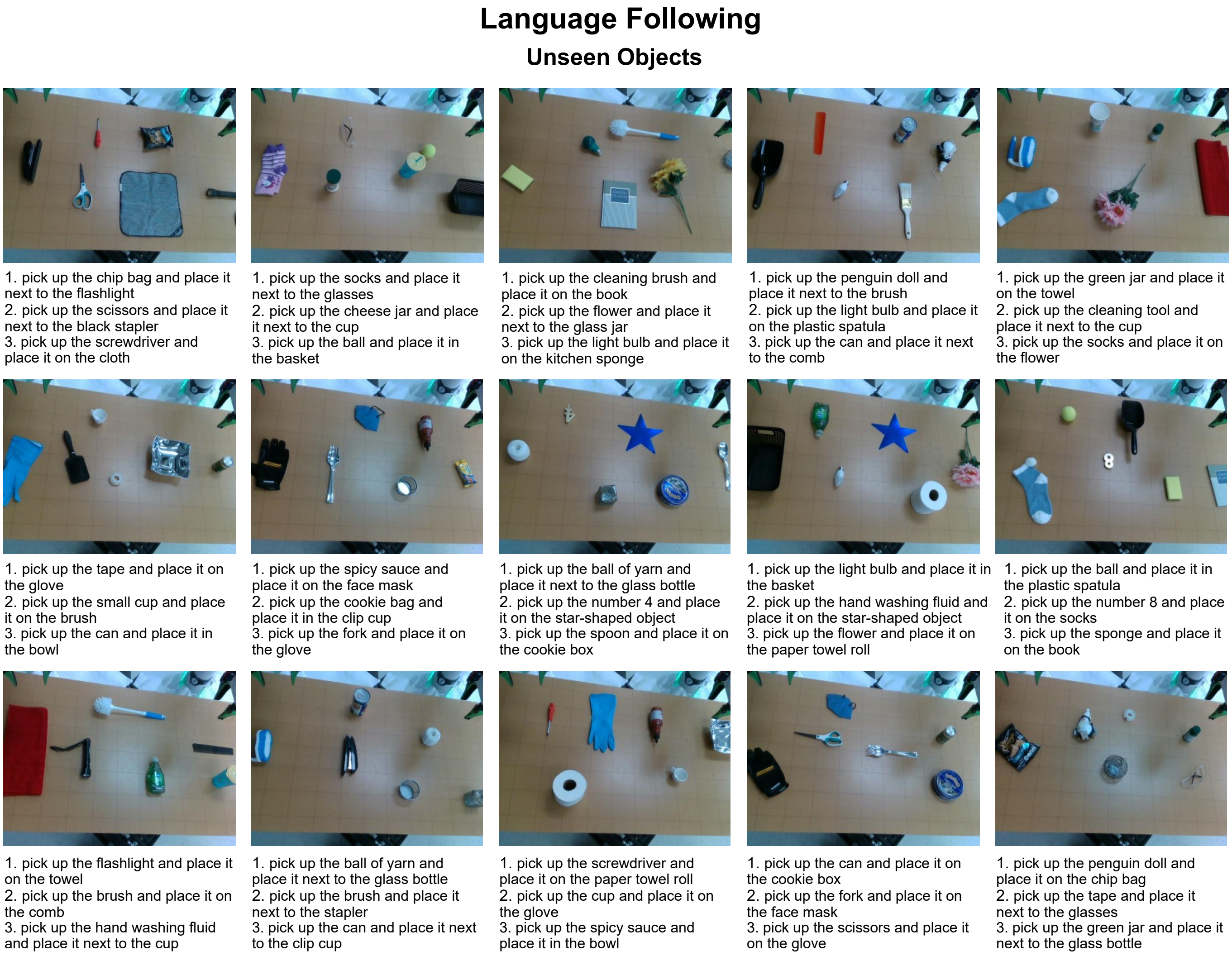
Instruction Generalization:
1. pick up the sweet fruit with pinkish skin and set it into the utensil jar
2. retrieve the tongs-like object and set it beside the cleaning device
3. move the container with thin orange pieces of food onto the drink mat



Seen Objects
1. pick up the kiwi slices and place them on the towel
2. pick up the utensil and place it next to the straw
3. pick up the can and place it next to the pink cup

Instruction Generalization:
1. lift the thin green pieces and set them down on the fabric
2. take the spatula and place it next to the thin hollow tube
3. grab the soda can and move it near the pink drinking vessel

Fig. S15. All the initial conditions and instructions used for real-world language following experiments for seen objects and instruction generalization settings. For each initial condition, the numbered labels (e.g., 1.) indicate the corresponding instruction used in the seen objects evaluation and its alternative phrasing used in the instruction generalization evaluation.



Language Following

Unseen Objects

1. pick up the chip bag and place it next to the flashlight
2. pick up the scissors and place it next to the black stapler
3. pick up the screwdriver and place it on the cloth

1. pick up the socks and place it next to the glasses
2. pick up the cheese jar and place it next to the cup
3. pick up the ball and place it in the basket

1. pick up the cleaning brush and place it on the book
2. pick up the flower and place it next to the glass jar
3. pick up the light bulb and place it on the kitchen sponge

1. pick up the penguin doll and place it next to the brush
2. pick up the light bulb and place it on the plastic spatula
3. pick up the can and place it next to the comb

1. pick up the green jar and place it on the towel
2. pick up the cleaning tool and place it next to the cup
3. pick up the socks and place it on the flower

1. pick up the tape and place it on the glove
2. pick up the small cup and place it on the brush
3. pick up the can and place it in the bowl

1. pick up the spicy sauce and place it on the face mask
2. pick up the cookie bag and place it in the clip cup
3. pick up the fork and place it on the glove

1. pick up the ball of yarn and place it on the paper towel roll
2. pick up the number 4 and place it on the star-shaped object
3. pick up the spoon and place it on the cookie box

1. pick up the light bulb and place it in the basket
2. pick up the hand washing fluid and place it on the star-shaped object
3. pick up the flower and place it on the paper towel roll

1. pick up the ball and place it in the plastic spatula
2. pick up the number 8 and place it on the socks
3. pick up the sponge and place it on the book

1. pick up the flashlight and place it on the towel
2. pick up the brush and place it on the comb
3. pick up the hand washing fluid and place it next to the cup

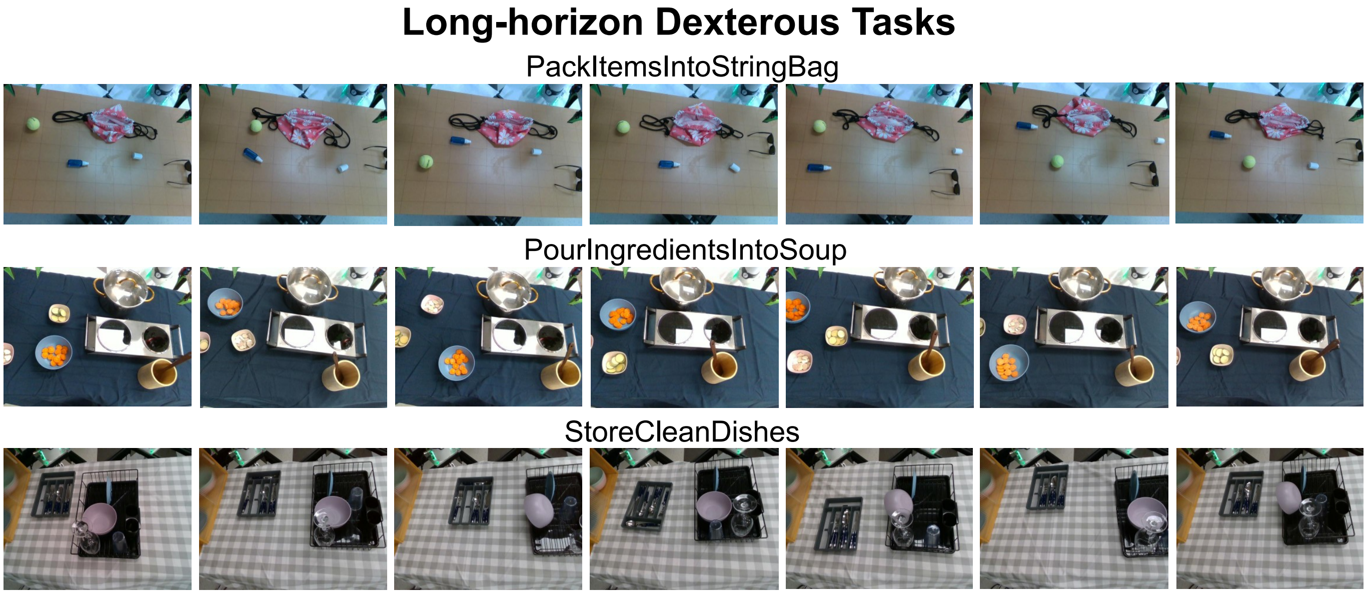
1. pick up the ball of yarn and place it next to the glass bottle
2. pick up the brush and place it next to the stapler
3. pick up the can and place it next to the clip cup

1. pick up the screwdriver and place it on the paper towel roll
2. pick up the cup and place it on the glove
3. pick up the spicy sauce and place it in the bowl

1. pick up the can and place it on the cookie box
2. pick up the fork and place it on the face mask
3. pick up the scissors and place it on the glove

1. pick up the penguin doll and place it on the chip bag
2. pick up the tape and place it next to the glasses
3. pick up the green jar and place it next to the glass bottle

Fig. S16. All the initial conditions and instructions used for real-world language following experiments for the unseen objects setting.



Long-horizon Dexterous Tasks

PackItemsIntoStringBag

PourIngredientsIntoSoup

StoreCleanDishes

Fig. S17. Sample initial conditions for the three real-world unseen, long-horizon, and dexterous tasks.

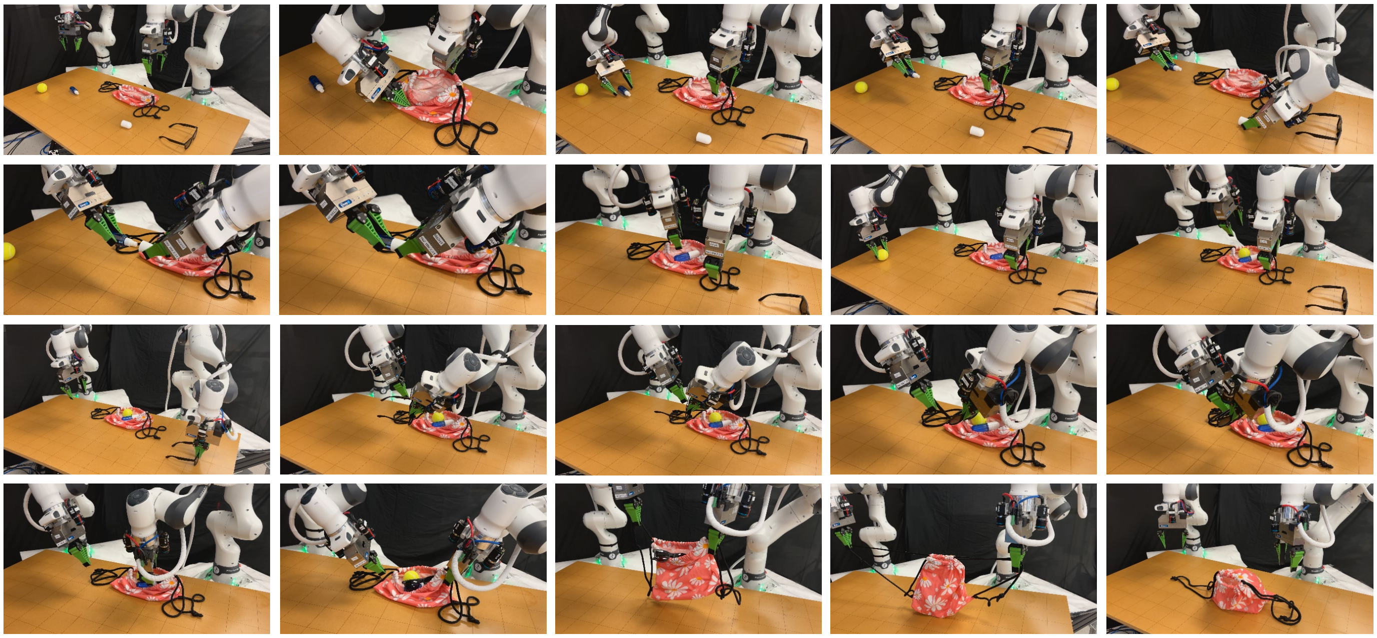


Fig. S18. Video snapshots showing the robot executing the *PackItemsIntoStringBag* task. The goal is to open the string bag, pack all the items (bottle, tennis ball, and sunglasses) into it and pull its strings to close it. The robot needs to cap the bottle and fold the sunglasses before packing them.

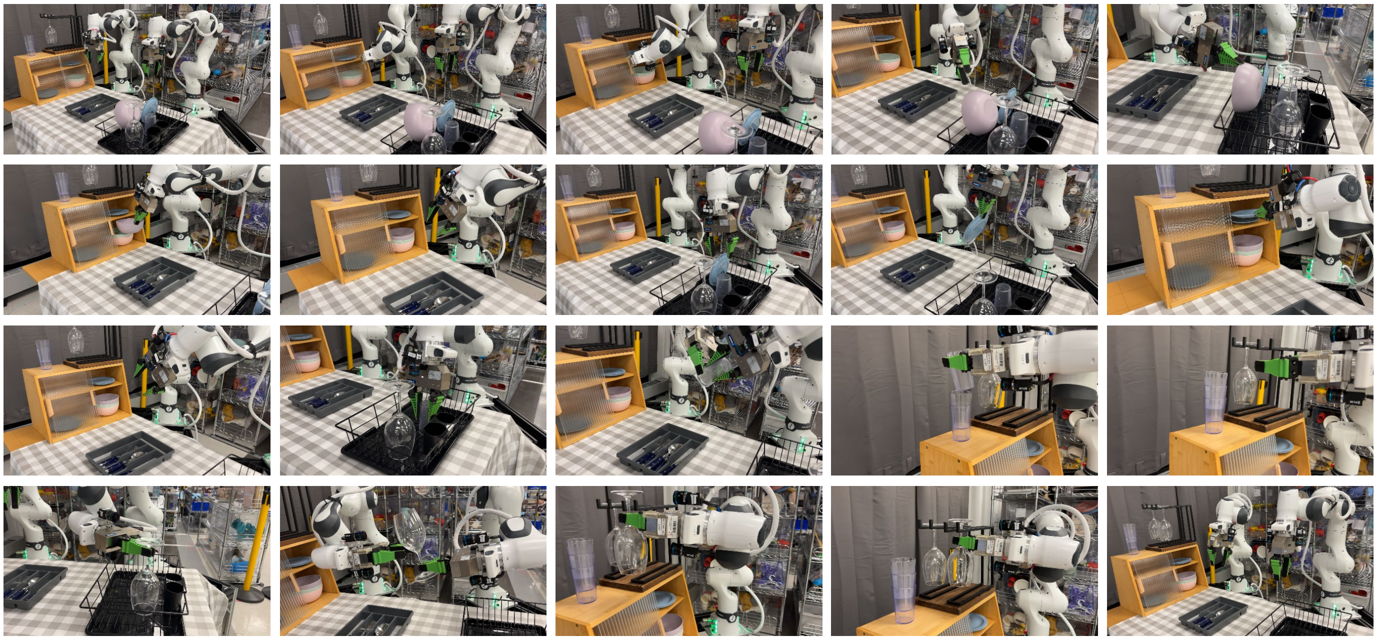


Fig. S19. Video snapshots showing the robot executing the *StoreCleanDishes* task. The goal is to store all the items (bowl, plate, cup, and wine glass) from the drying rack into the dish cabinet. The robot must place the bowl onto the bowl stack, the plate onto the plate stack, stack the cup on top of the other cups, and insert the wine glass into the glass rack.

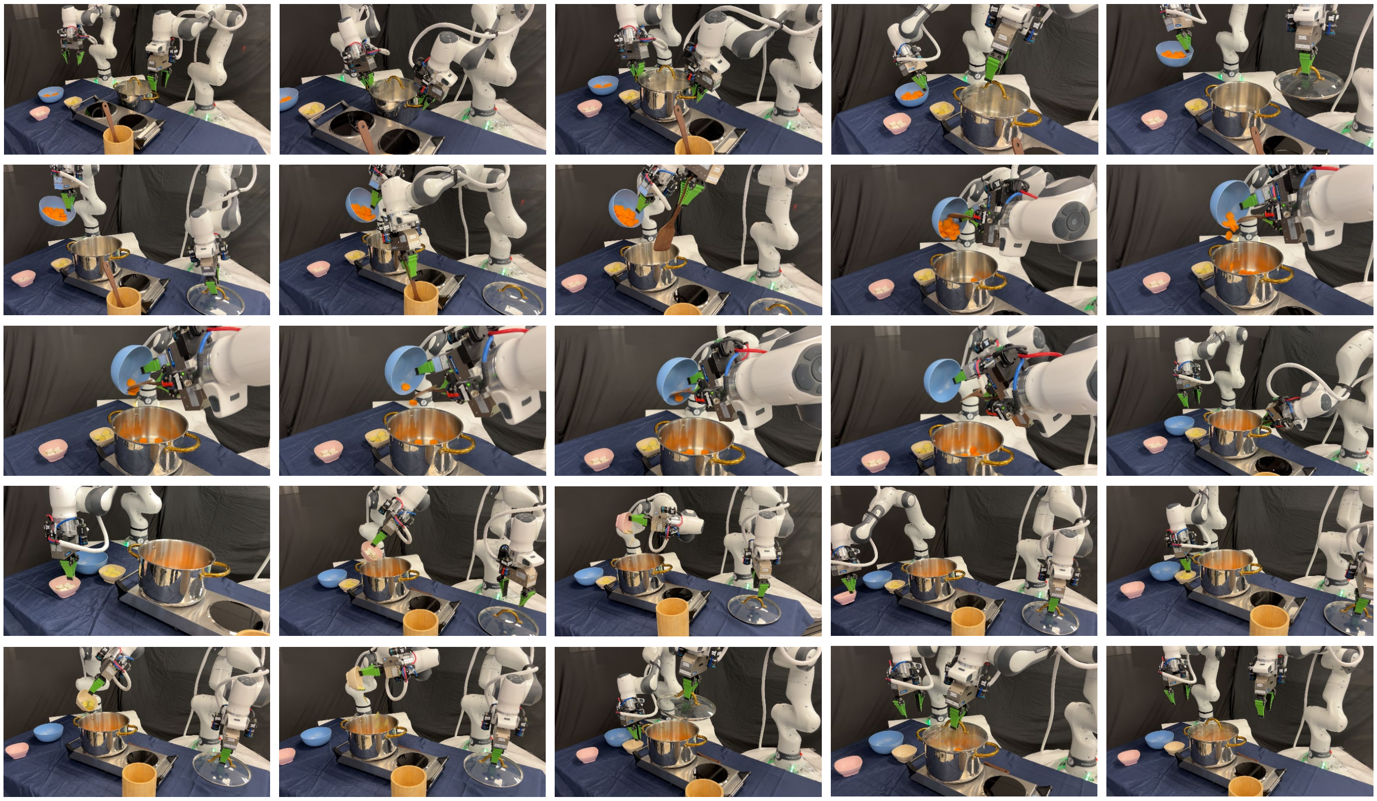


Fig. S20. Video snapshots showing the robot executing the *PourIngredientsIntoSoup* task. The goal is to pour all the ingredients (chopped carrots, mushrooms, and cucumbers) into the soup pot. Prior to pouring, the robot must put the pot onto the stove and remove its lid. The robot must scoop out all of the carrot slices using a spatula.