GUIDANCE UNDER SHIFT: DEPLOYMENT-ALIGNED SEMANTIC INTERFACES FOR VISION–LANGUAGE–ACTION POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Explicit chain-of-thought (CoT) can improve vision–language–action (VLA) policies by exposing task progress and spatial intent, but guidance available during training is often more reliable and temporally aligned than online predictions at deployment, which may be incorrect, missing, or outdated. This mismatch can make a policy trained with reliable guidance brittle at deployment. Our flow-matching analysis bounds the deployment loss by training error, guidance-distribution coverage, and conditional drift, motivating two training principles: covering deployment-time guidance conditions and exposing the policy to temporal staleness. We therefore train a VLA policy with subtask descriptions, affordance regions, and image-space trajectories drawn from annotated, generated, corrupted, and missing guidance, while varying their physical age—the elapsed time from the source observation to policy use. Current observations and action supervision remain aligned. An asynchronously refreshed dependency graph overlaps reasoning and action generation while preserving an explicit and inspectable CoT interface. Controlled evaluations vary guidance reliability and physical age. We evaluate closed-loop success on LIBERO and on training and testing tasks of RoboTwin 2.0, as well as a real-world "desktop-cleaning" task and its variants, demonstrating robustness and generalization on physical robots.

1 INTRODUCTION

Recent vision–language–action (VLA) models add chain-of-thought (CoT) reasoning to action generation to make actions more robust and raise task success. Explicit CoT states task plans and visually grounded decisions (Zawalski et al., 2024; Yin et al., 2025), and related methods predict visual subgoals (Zhao et al., 2025) or affordances (Yu et al., 2026). We call the CoT fields that condition an action policy *guidance*, whether annotated or predicted.

Guidance gives the action policy more information, but it also introduces errors. A policy is trained on clean annotated guidance but deployed with online predictions, which can be **delayed**, **missing**, or **incorrect**. First, generating a subtask description, an affordance, or a spatial plan takes time, so the guidance used by the action policy may be generated from an earlier observation than its current input. Second, guidance is unavailable when generation fails or is not refreshed in time. Third, online predictions can be wrong or incomplete even when they are current. A policy trained to rely on clean annotations can over-trust such guidance and increase its action error, even when the guidance remains partially informative.

Prior work addresses these errors one at a time. For incorrect guidance, GHIL-Glue addresses errors in predicted subgoals (Hatch et al., 2024), ERVLA trains with reasoning dropout (Sun et al., 2026b), HiRoC aligns an executor to planner subgoals (Kong et al., 2026), and CAC-VLA gates predicted latent-action context (Xiong et al., 2026). For delayed guidance, Fast ECoT (Duan et al., 2025), VLASH (Tang et al., 2025), and DuoCore-FS (Zou et al., 2025) reduce or hide reasoning latency. TIC-VLA trains with delayed semantic inputs for navigation (Huang et al., 2026), and CloudEdgeVLA assigns current action targets to randomly delayed cloud inputs (Peng et al., 2026a).

At deployment, however, the three errors occur together: a delayed prediction can also be wrong, and each field can fail or refresh at its own time. It remains open when a guided policy keeps its advantage

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

over an observation-only policy, and how strongly it should rely on guidance whose reliability and age keep changing from one control step to the next.

We answer both by analyzing the action policy's training data. In conditional flow matching, the gain from guidance splits exactly into the information that guidance adds and the extra error that the learned policy makes in using it; guidance helps only when the first exceeds the second. We bound the extra error by the training error, the coverage of deployment guidance by training guidance, and the shift in the optimal action prediction for the same guidance between training and deployment. At the training optimum, the deployment excess risk equals this conditional drift. The bound gives a sufficient condition for guidance to help and guides the choice of training sources and ages.

We turn this condition into an error-tolerant training strategy for a two-model policy. A vision–language–model PERCEIVER predicts a subtask description, affordance regions, and an image-space trajectory. A flow-matching WORKER generates actions from the current observation and the latest guidance, with each field tagged by its physical age. The WORKER is trained on annotated, PERCEIVER-generated, perturbed, and missing guidance. Generated guidance supplies examples of the deployed predictor's outputs; perturbed and missing fields expose the WORKER to unreliable inputs. Fields drawn from randomly sampled earlier times expose it to delayed content with its physical age. At inference, asynchronous guidance refresh runs alongside action generation from the latest fields.

We evaluate on two simulation benchmarks and a real robot. On LIBERO, our policy reaches 99.0% mean success, compared with 97.1% for $\pi_{0.5}$. On RoboTwin 2.0, we train on one task subset and evaluate on held-out tasks: our policy reaches 25.04% on testing tasks, compared with 19.48% for $\pi_{0.5}$. On fixed RoboTwin 2.0 training states, wrong guidance raises the action error of a policy fine-tuned only on annotated guidance by 0.88 and that of our policy by 0.01 ($\times 10^{-2}$ rad). On a real-robot desktop-cleaning task and its variants, our policy achieves much higher mean task scores than the baseline. The robot shows emergent error correction: when a subtask is not completed, the PERCEIVER keeps predicting that subtask, and the WORKER repeats it until it is done.

Our contributions are as follows.

- **Analysis.** Guidance helps a learned policy only when the information it adds exceeds the error of using it; a bound relates this error to training fit, coverage, and drift.
- **Method.** We train the WORKER on annotated, generated, perturbed, and missing guidance tagged with its age, matching what it receives at deployment.
- **Evidence.** Our recipe makes the policy nearly insensitive to wrong guidance and reduces the harm of outdated guidance. It improves closed-loop success on LIBERO and RoboTwin 2.0, and mean task scores on a real-robot desktop-cleaning task.

2 RELATED WORK

Guidance for generative robot policies. RT-2 (Brohan et al., 2023) and OpenVLA (Kim et al., 2024) adapt vision–language representations to robot actions, while Diffusion Policy (Chi et al., 2023), CogACT (Li et al., 2024), and π_0 (Black et al., 2024) model continuous actions generatively. Such policies can be conditioned on embodied reasoning (Zawalski et al., 2024), trajectory or affordance structure (Gu et al., 2024; Zheng et al., 2024; Nasiriany et al., 2024; Li et al., 2025), and predicted visual subgoals (Zhao et al., 2025); our three fields follow these examples. Generated guidance can help or harm control (Mazzaglia et al., 2026; Sun et al., 2026b; Yin et al., 2025; Hatch et al., 2024), and several methods adapt the action policy to it (Yang et al., 2026; Kong et al., 2026; Peng et al., 2026b; Xiong et al., 2026). Our WORKER learns its reliance on each field from the training mixture, and our analysis asks when harm from guidance exceeds its information gain.

Asynchronous guidance. Fast ECoT (Duan et al., 2025) and VLASH (Tang et al., 2025) reduce reasoning or execution latency, and AsyncVLA (Hirose et al., 2026) and DuoCore-FS (Zou et al., 2025) combine delayed or asynchronous semantics with current control. TIC-VLA (Huang et al., 2026) and CloudEdgeVLA (Peng et al., 2026a) train with delayed inputs and current action targets, so this practice has precedent. We apply it to three manipulation fields that refresh separately, each with its own physical age, and mix delay with wrong and missing guidance. Appendix B compares these methods, including their guidance types and training data, in more detail.

3 WHY GUIDANCE CAN INCREASE ACTION ERROR

We analyze expected squared flow-matching error while varying guidance between training and deployment. The analysis separates the information guidance provides from the learned policy's error in using it, then relates deployment error to the training guidance distribution.

Let X contain current images, instruction, and robot state, and let $U = (Z, \Delta)$ contain guidance contents, missing markers, and field ages. Ages measure elapsed time since the source observations. For a demonstrated action chunk A, independent noise $\xi \sim \mathcal{N}(0, I)$, and independent flow time $\tau \sim \rho$ on $[0, 1]$, flow matching (Lipman et al., 2023) predicts $V = \xi - A$ from $S = (A_\tau, \tau, X)$, where $A_\tau = \tau\xi + (1-\tau)A$. Training ($D = \mathrm{tr}$) and deployment ($D = \mathrm{te}$) share the law of (X, A, ξ, τ) and differ in guidance. The target and learned fields are square-integrable in both domains.

3.1 GUIDANCE INFORMATION AND LEARNED-POLICY ERROR

Let $\hat{v}_U(S, U)$ and $\hat{v}_X(S)$ be the learned fields with and without guidance, held fixed during evaluation. Their risks $R_D^{\mathrm{FM}}(\hat{v}_U; U)$ and $R_D^{\mathrm{FM}}(\hat{v}_X)$ are the expected squared errors in predicting V under domain D. Let $B_D^{\mathrm{FM}}(X, U)$ and $B_D^{\mathrm{FM}}(X)$ be the minimum risks over square-integrable fields with the respective inputs. With $\mathbb{E}_D$ denoting expectation in domain D, the optimal fields are the conditional means $v^\star_{U,D}(S, U) = \mathbb{E}_D[V \mid S, U]$ and $v^\star_{X,D}(S) = \mathbb{E}_D[V \mid S]$. Tab. 1 defines the risks and the three quantities used in the transfer bound.

Table 1: Mathematical definitions for Section 3. $\mathbb{E}_D$ averages over domain D; p_D is the law of (S, U). Arguments X, U in B_D^{FM} name the available conditions; both policies also receive the noisy action and flow time in S.

Quantity	Mathematical definition
Learned guided error	$R_D^{\mathrm{FM}}(\hat{v}_U; U) := \mathbb{E}_D[\|V - \hat{v}_U(S, U)\|_2^2]$
Learned observation-only error	$R_D^{\mathrm{FM}}(\hat{v}_X) := \mathbb{E}_D[\|V - \hat{v}_X(S)\|_2^2]$
Minimum guided error	$B_D^{\mathrm{FM}}(X, U) := \mathbb{E}_D[\|V - v^\star_{U,D}(S, U)\|_2^2]$
Minimum observation-only error	$B_D^{\mathrm{FM}}(X) := \mathbb{E}_D[\|V - v^\star_{X,D}(S)\|_2^2]$
Guided training excess risk	$\epsilon_{\mathrm{tr}}(\hat{v}_U; U) := \mathbb{E}_{\mathrm{tr}}[\|\hat{v}_U(S, U) - v^\star_{U,\mathrm{tr}}(S, U)\|_2^2]$
Coverage coefficient	$\beta(U) := \inf\{b \geq 0 : p_{\mathrm{te}}(E) \leq b\, p_{\mathrm{tr}}(E) \text{ for every measurable } E\}$
Conditional drift	$\Gamma_{\mathrm{te}}(U) := \mathbb{E}_{\mathrm{te}}[\|v^\star_{U,\mathrm{tr}}(S, U) - v^\star_{U,\mathrm{te}}(S, U)\|_2^2]$

The information gain is $G_{\mathrm{te}}(U) := B_{\mathrm{te}}^{\mathrm{FM}}(X) - B_{\mathrm{te}}^{\mathrm{FM}}(X, U) \geq 0$. Let $e_{U,\mathrm{te}}$ and $e_{X,\mathrm{te}}$ denote each learned policy's deployment risk minus its own minimum risk. Subtracting the two risks gives the exact comparison (Appendix A.2):

$$R_{\mathrm{te}}^{\mathrm{FM}}(\hat{v}_U; U) - R_{\mathrm{te}}^{\mathrm{FM}}(\hat{v}_X) = -G_{\mathrm{te}}(U) + e_{U,\mathrm{te}} - e_{X,\mathrm{te}}. \tag{1}$$

Guidance helps exactly when $e_{U,\mathrm{te}} - e_{X,\mathrm{te}} < G_{\mathrm{te}}(U)$: its information gain must exceed its additional excess risk. Even informative guidance can increase error if the learned policy uses it poorly. When U is determined by X, $G_{\mathrm{te}}(U) = 0$; guidance can still help by making available information easier to use, yielding $e_{U,\mathrm{te}} < e_{X,\mathrm{te}}$.

3.2 HOW TRAINING ERROR TRANSFERS TO DEPLOYMENT

Three quantities control the guided deployment excess risk. The training excess risk $\epsilon_{\mathrm{tr}}(\hat{v}_U; U)$ measures squared error relative to the training-optimal field. The coverage coefficient $\beta(U)$ bounds the deployment-to-training density ratio of the complete input (S, U). The conditional drift $\Gamma_{\mathrm{te}}(U)$ is the deployment-averaged squared difference between the training and deployment conditional means at the same input. Tab. 1 defines these quantities; Appendix A.3 gives the proof.

Proposition 1 (Transfer of guided field error)**.** *Under the assumptions above, if $\beta(U) < \infty$, then*

$$e_{U,\mathrm{te}} \leq \left(\sqrt{\beta(U)\epsilon_{\mathrm{tr}}(\hat{v}_U; U)} + \sqrt{\Gamma_{\mathrm{te}}(U)}\right)^2 =: C_U. \tag{2}$$

Thus $C_U < G_{\mathrm{te}}(U) + e_{X,\mathrm{te}}$ suffices for guidance to improve deployment risk. At the population training optimum, $e_{U,\mathrm{te}} = \Gamma_{\mathrm{te}}(U)$ (Appendix A.3, Corollary 1): fitting the training law leaves conditional drift as the deployment excess risk.

3.3 How the bound guides training

Mixing annotated, generated, noisy, incorrect, and missing guidance exposes the Worker to deployment inputs, targeting $\beta(U)$. Generated guidance approximates the deployed guidance–action relation, targeting $\Gamma_{\mathrm{te}}(U)$. Temporal jitter samples earlier guidance and supplies its physical age while keeping current observations and action supervision fixed. Source weights and delay ranges must reflect deployment: unrealistic mixtures can increase drift or training error. More demonstrations from a fixed population can reduce finite-data error while leaving $\beta(U)$ and $\Gamma_{\mathrm{te}}(U)$ unchanged. Appendix A.4 expands this discussion; Section 4 gives the procedure.

4 Method

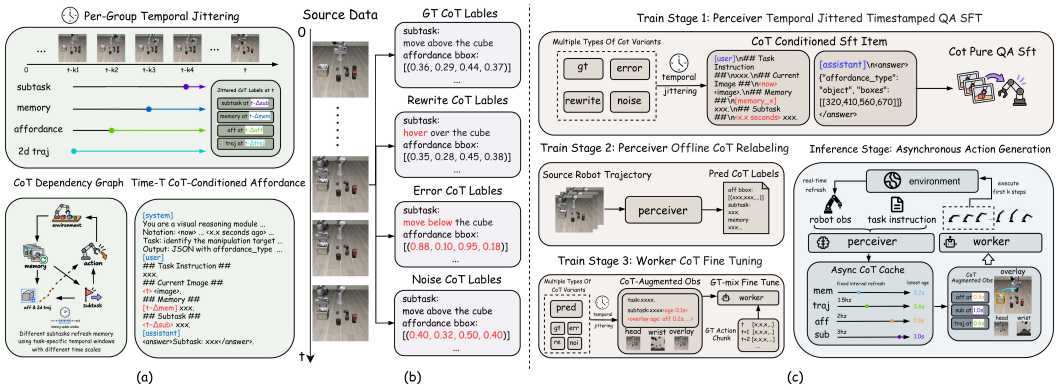


Figure 1: **Perceiver and Worker framework.** (a) CoT fields use independently sampled earlier source times. (b) Annotation perturbations supply noisy and incorrect guidance. (c) Train the Perceiver, generate guidance offline, then train the Worker on mixed sources. Asynchronous inference supplies fields with different ages.

Fig. 1 summarizes our training procedure: train the Perceiver on annotations, generate guidance offline, and train the Worker with mixed sources and ages.

4.1 Structured CoT and Model Interfaces

We define CoT as four explicit intermediate predictions: *memory*, *subtask*, *affordance*, and *2D trajectory*. Memory summarizes observed task progress; subtask describes the current operation; affordance identifies its object or target region; and trajectory specifies a short future path of the end effector in image coordinates. The Perceiver generates these fields through four question–answer tasks with shared model parameters. Each task receives the instruction and the inputs in Tab. 2, and returns a structured text answer. Boxes and trajectory points use normalized image coordinates.

Table 2: CoT inputs and outputs at inference. All queries also receive the task instruction.

Field	Visual input and CoT context	Output
Memory	Completed-window video + prior memory	Window event summary
Subtask	Recent video + completed-window memory	Current operation + target
Affordance	Current image and available subtask	Boxes around the object or region
2D trajectory	Current image and available subtask	Ordered future end-effector points

The Worker takes the current camera views, robot state, instruction, and available guidance, and outputs a continuous action chunk. Subtask and field ages enter as text; affordance boxes and trajectory points are drawn on a copy of the current front-camera image, supplied alongside the original views. The Perceiver uses memory to generate the subtask and its dependent geometry (Fig. 1(a)), which convey task history to the Worker as guidance for action prediction.

4.2 CoT Annotation, Perceiver Training, and Prediction

We annotate demonstration videos and robot data with the four CoT fields in Tab. 2, denoted GT (Appendix F.1). Memory annotation combines event-based segments and fixed time windows; inference uses fixed time windows. In Stage 1, we train the PERCEIVER on these annotations with supervised fine-tuning (SFT), first on general robot data and then on the target benchmark.

Training prompts also include temporally jittered CoT context. For geometric queries, we sample an earlier subtask and provide its age while keeping the current image and target aligned. Memory contains only completed segments, and context dropout simulates missing guidance. Appendix F.2 gives the PERCEIVER training details.

In Stage 2, we run the trained PERCEIVER once over the training demonstrations, using its own predicted history, and add the outputs as PRED CoT for WORKER training. Appendix F.3 describes the generation procedure and prediction records.

4.3 Worker Training with Mixed CoT

In Stage 3, we independently sample GT, PRED, NOISE, or ERR CoT for each field. NOISE slightly perturbs geometric coordinates; ERR uses a subtask from another time or substantially displaced geometry (Fig. 1(b)). Random dropout and an initial interval without CoT cover missing guidance.

We also independently sample an earlier source time for each field and provide its age to the WORKER (Fig. 1(a)). PRED retains its actual prediction source time. Current observations and correct demonstration actions remain aligned while the CoT content and ages vary. The WORKER uses the normalized flow-matching objective on valid action coordinates (Appendix F.4, Eq. (19)). Appendix F.5 gives source weights and age ranges for LIBERO and RoboTwin 2.0.

4.4 Asynchronous Inference

We deploy the PERCEIVER with vLLM and maintain an independent request loop for each CoT type. Memory accumulates summaries of completed fixed time windows; subtask queries use this history, and geometric queries use the available subtask. Generated results and their source times enter the shared CoT cache in Fig. 1(c), where newer source frames replace older ones.

At each action query, the WORKER reads the latest subtask, geometry, and their ages from the cache together with the current observation. It generates actions without waiting for outstanding CoT requests; missing fields stay empty, and cached results age until updated.

5 Experiments

5.1 Simulation Experiments

5.1.1 Setup

Simulation Benchmarks. LIBERO (Liu et al., 2023) has four suites, Spatial, Object, Goal, and Long, with disjoint demonstrations; our evaluation uses 500 rollouts per task, and we report each suite and their mean. On RoboTwin 2.0 (Chen et al., 2025), we split 50 tasks into 25 training and 25 testing tasks. The detailed division list is available in Tab. 8. Train and Test report success on training and testing tasks, respectively, with 100 rollouts per task.

Models and baselines. The WORKER retains the $\pi_{0.5}$ architecture (Black et al., 2025) and adds guidance through the interface in Section 4. From a pretrained checkpoint, we continue pretraining on DROID (Khazatsky et al., 2024) and AgiBot World (AgiBot-World-Contributors et al., 2025) with annotated guidance, then fine-tune on each simulation benchmark. The observation-only $\pi_{0.5}$ baseline follows the same stages without guidance.

The PERCEIVER starts from Embodied-R1.5-8B (Yuan et al., 2026), based on Qwen3-VL-8B (Bai et al., 2025). It uses the same demonstrations and the training procedure in Section 4. Tables distinguish our evaluations from reported results; Appendix C gives the protocol.

Table 3: LIBERO success (%); **bold** is best and *underlined italic* second best. Guidance groups follow ACoT-VLA (Zhong et al., 2026). We evaluate $\pi_{0.5}$, Embodied-R1.5-VLA, and our method. Other results are reported values; CoT-VLA's mean is recomputed from the reported success rates of all four suites.

Method	Guidance	Spatial	Object	Goal	Long	Mean
CoT-VLA (Zhao et al., 2025)	Visual	87.5	91.6	87.6	69.0	83.9
DreamVLA (Zhang et al., 2025)	Visual	97.5	94.0	89.5	89.5	92.6
UniVLA (Wang et al., 2025)	Visual	95.4	98.8	93.6	94.0	95.5
F1 (Lv et al., 2025)	Visual	98.2	97.8	95.4	91.3	95.7
GE-Act (Liao et al., 2025)	Visual	98.2	97.6	95.8	94.4	96.5
ECoT (Zawalski et al., 2024)	Linguistic	77.0	84.0	75.0	57.0	73.3
Fast ECoT (Duan et al., 2025)	Linguistic	83.0	85.0	83.0	69.0	80.0
ThinkAct (Huang et al., 2025)	Linguistic	88.3	91.4	87.1	70.9	84.4
DD-VLA (Liang et al., 2025)	Linguistic	97.2	98.6	97.4	92.0	96.3
MemoryVLA (Shi et al., 2025)	Linguistic	98.4	98.4	96.4	93.4	96.7
VLA-Adapter (Wang et al., 2026)	Linguistic	97.8	99.2	97.2	95.0	97.3
$\pi_{0.5}$ (Black et al., 2025)	Linguistic	98.5	99.0	97.5	93.5	97.1
Embodied-R1.5-VLA (Yuan et al., 2026)	Linguistic	98.3	99.2	97.6	93.9	97.2
ERVLA (Sun et al., 2026b)	Linguistic	97.2	97.2	93.2	87.0	93.7
LaST$_0$ (Liu et al., 2026)	Latent	**99.2**	*99.6*	98.0	95.6	98.1
ACoT-VLA (Zhong et al., 2026)	Action	*98.6*	99.0	**99.4**	*97.0*	*98.5*
CoA-VLA (Li et al., 2025)	Vis. + ling.	85.3	93.1	85.8	55.0	79.8
Ours	Vis. + ling.	**99.2**	**100.0**	*99.0*	**97.8**	**99.0**

Table 4: RoboTwin 2.0 success (%) from our evaluation of all methods on the fixed task split, with best and second-best results marked as in Tab. 3.

Method	Train	Test
π_0	50.88	12.00
$\pi_{0.5}$ (obs. only)	*77.28*	*19.48*
Wall-X (Zhai et al., 2025)	68.28	14.88
VLA-JEPA (Sun et al., 2026a)	63.36	12.68
Ours	**89.12**	**25.04**

Table 5: RoboTwin 2.0 fine-tuning ablations (%). Variants share data and budget.

Variant	Train	Test
F (full)	89.12	25.04
A1 (GT-only)	85.92	21.68
A2 (no numeric age)	83.04	17.48
A3 (no extra jitter)	85.16	19.84
A4 (PRED-only)	87.48	23.56

Table 6: Action error ($\times 10^{-2}$ rad; lower is better) on fixed RoboTwin training states when all three guidance fields are correct, wrong (ERR), stale, or MISSING. Parentheses give the change from the same recipe's correct or fresh guidance. Stale guidance is GT from 3 s earlier with its true age.

(a) Wrong and missing guidance: 25 tasks, 75 states

Recipe	Correct	ERR	MISSING
Observation-only $\pi_{0.5}$	–	–	3.10
A1: GT-only	1.97	2.85 (+0.88)	2.64 (+0.67)
A4: Pred-only	1.95	2.38 (+0.43)	2.31 (+0.36)
F: complete recipe	1.96	1.97 (+0.01)	1.99 (+0.03)

(b) Stale and missing guidance: 23 tasks, 77 states

Recipe	Fresh (0.2 s)	Stale (3 s)	MISSING
Observation-only $\pi_{0.5}$	–	–	2.27
A3: no age jitter	1.71	2.13 (+0.42)	1.92 (+0.20)
F: complete recipe	1.85	1.96 (+0.11)	1.90 (+0.05)

5.1.2 BENCHMARK RESULTS AND ABLATIONS

LIBERO. Our method leads Tab. 3 with 99.0% mean success and improves on $\pi_{0.5}$ under our protocol in all four suites. The largest gain is on Long (93.5% to 97.8%). Memory carries summaries of completed steps into each subtask query, providing a record of task progress as the scene changes. We hypothesize that this history helps the PERCEIVER maintain the correct subtask over long rollouts.

RoboTwin 2.0. Our method achieves 89.12% Train success and 25.04% Test success (Tab. 4). The Test gain of 5.56 percentage points over observation-only $\pi_{0.5}$ on 25 held-out tasks suggests that CoT's subtask and spatial predictions provide a transferable action interface. Both splits use the same guidance fields to specify the current operation and spatial target.

Ablations. Tab. 5 compares F with A1–A4, whose changes apply only to downstream fine-tuning. All five share the general WORKER checkpoint, PERCEIVER, fine-tuning data, and budget (Appendix C). F achieves the highest success on both splits. PRED-only (A4) outperforms GT-only (A1) on both splits, suggesting that generated examples help the WORKER adapt to the predictor's outputs. Both single-source variants remain below F, supporting mixed-source fine-tuning. Hiding numerical age (A2) or removing extra age jitter (A3) causes larger Test drops, of 7.56 and 5.20 percentage points. Age inputs tell the WORKER how old each field is; temporal jitter supplies delayed examples during fine-tuning. The gains support using both under asynchronous guidance.

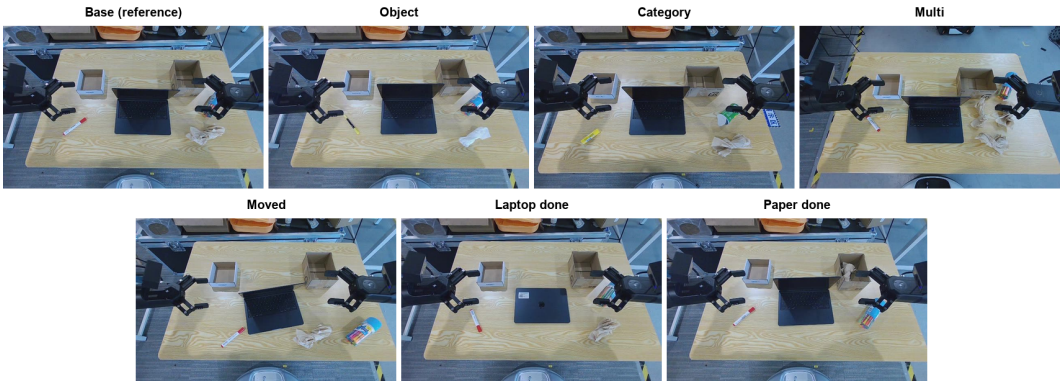


Figure 2: **Real-robot scene conditions.** Initial head-camera views. Base uses a reference demonstration's initial layout. Object changes the pen and paper appearance; Category replaces the holder with a coke can and the pen with a glue stick and binder clip; Multi uses three sheets of paper; Moved shifts loose items by about 10 cm and rotates the paper, marker box, and laptop. Laptop done and Paper done begin with the named subtask completed and unscored.

5.1.3 Action error under wrong, stale, and missing guidance

We use the 5,000-update WORKER checkpoints F, A1, A3, and A4 from the RoboTwin 2.0 ablation experiments in Tab. 5. On fixed states from RoboTwin training demonstrations, we keep observations and sampling noise unchanged while replacing all three CoT fields with correct, wrong, stale, or missing guidance. We measure root-mean-square action error against the demonstrations to assess sensitivity to guidance; Appendix E gives the evaluation details.

Wrong and missing CoT. With correct guidance, F, A1, and A4 have similar action errors. Corrupting or removing the guidance substantially harms the GT-only and Pred-only variants, while F remains stable (Tab. 6a). Measured in 10^{-2} rad, F's error is 1.96 under correct guidance, 1.97 under wrong guidance, and 1.99 when guidance is missing. Pred-only fine-tuning reduces both error increases relative to GT-only fine-tuning, while the full mixture gives the smallest increases among the three recipes. Including perturbed and missing CoT in fine-tuning reduces sensitivity to these failures while retaining accuracy under correct guidance.

Stale CoT. F degrades less than A3 when guidance becomes stale, although A3 is more accurate with fresh guidance (Tab. 6b). At a guidance age of 3 s, the error increase is 0.11×10^{-2} rad for F and 0.42×10^{-2} rad for A3. Under stale guidance, F's error is 1.96×10^{-2} rad, below the observation-only baseline's 2.27×10^{-2} rad. On RoboTwin states with outdated subtask CoT, temporal jitter trades some accuracy under fresh guidance for lower sensitivity to stale content.

Together, these results show that training with mixed CoT sources and temporal jitter yields an action policy that is robust to imperfect CoT guidance in practice. On the evaluated RoboTwin states, action error remains low even when CoT guidance is wrong, stale, or missing.

5.2 Real-robot evaluation

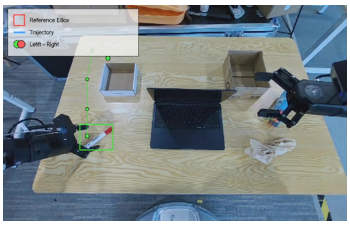
5.2.1 Setup

We deploy the policy on an AgiBot G2 dual-arm robot with a head camera and two wrist cameras. The desktop-cleaning task has four subtasks: put the pen into the pen holder, drop the paper into the bin, stand the marker box upright, and close the laptop. The WORKER predicts 14-dimensional end-effector deltas at 30 Hz. The PERCEIVER and the WORKER share one RTX 5090 GPU, and the robot executes 25 actions per chunk, for at most 200 chunks.

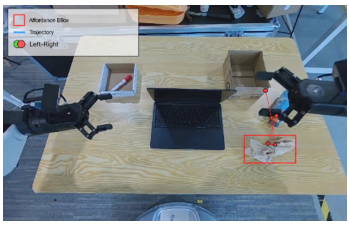
Base denotes the reference scene setting in Fig. 2. For real-robot adaptation, we train the PERCEIVER and WORKER only on the Base dataset, which contains 866 demonstrations of two desktop-cleaning tasks recorded on AgiBot robots (AgiBot-World-Contributors et al., 2025). The baseline starts from a

Table 7: Real-robot mean task scores (%) over five trials per condition. Pen: pen in holder; Paper: paper in bin, with partial credit in Multi; Box: marker box upright; Laptop: laptop closed. Overall averages the applicable subtask scores. Subtasks completed before the trial are excluded (–). The higher Overall score in each condition is bold.

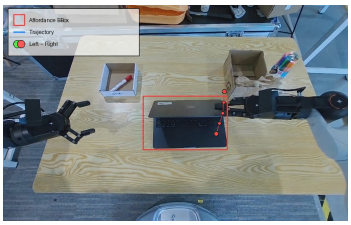
	Pen		Paper		Box		Laptop		Overall	
Condition	$\pi_{0.5}$	Ours	$\pi_{0.5}$	Ours	$\pi_{0.5}$	Ours	$\pi_{0.5}$	Ours	$\pi_{0.5}$	Ours
Base	0	100	0	100	80	80	80	100	40	**95**
Object	20	80	40	100	40	80	20	100	30	**90**
Category	0	80	20	100	0	80	40	100	15	**90**
Multi	40	80	0	84	60	40	0	80	25	**71**
Moved	60	40	0	60	0	40	0	80	15	**55**
Laptop done	40	100	40	100	60	60	–	–	46.7	**86.7**
Paper done	60	80	–	–	40	40	60	100	53.3	**73.3**



Subtask(age 0.5 s): "Pick: Left arm picks up the pen on the table"



Subtask (age 0.6 s): "Pick: Right arm picks up the tissue on the table"



Subtask (age 0.9 s): "Close: Right arm closes the notebook on the table"

Figure 3: **Guidance during a successful Base trial.** Head-camera views with affordance boxes and trajectories (green: left arm; red: right arm); subtask and age below. Left: both fields describe pen picking. Middle: subtask advances to paper; geometry still indicates pen placement. Right: the robot closes the notebook after eight rejected PERCEIVER updates, using the last accepted guidance.

pretrained $\pi_{0.5}$ checkpoint and is fine-tuned on the same Base demonstrations without guidance. We evaluate both action policies after 6,000 real-robot fine-tuning updates. We test generalization by evaluating the trained policies directly on the scene variants in Fig. 2, without further training. Both policies share the controller, cameras, instruction, and safety limits.

5.2.2 EVALUATION PROTOCOL

Fig. 2 defines the seven scene conditions. We evaluate each policy in five trials per condition. Each applicable subtask receives a score between 0 and 100, including partial credit for clearing some of the paper in Multi. All other subtask scores are either 0 or 100. Subtasks completed before a trial are excluded from scoring in Laptop done and Paper done. We report mean scores as percentages and average the applicable subtasks for the overall score (Appendix D).

5.2.3 RESULTS

Our policy achieves higher overall mean task scores than $\pi_{0.5}$ in all seven conditions (Tab. 7), including 95% versus 40% in Base. The largest gains in Base are in pen placement and paper collection, which both rise from 0% for $\pi_{0.5}$ to 100% for our policy. Both policies score 80% on standing the marker box upright, so the overall gain comes from more consistent completion of the other three subtasks. The PERCEIVER decomposes the broad cleaning instruction into a current operation and target for the WORKER. This guidance reduces the WORKER's need to infer the task sequence from an instruction that leaves individual operations unspecified. Fig. 3 shows rendered CoT and the corresponding subtask during a successful rollout.

After Base-only adaptation, the policy handles changed object appearances and categories, repeated paper collection, and partially completed tasks. Category gives the largest gain: our policy scores 90% overall, compared with 15% for $\pi_{0.5}$. The policy scores 86.7% in Laptop done and 73.3% in Paper done across the three remaining subtasks. These scene variants require selecting operations

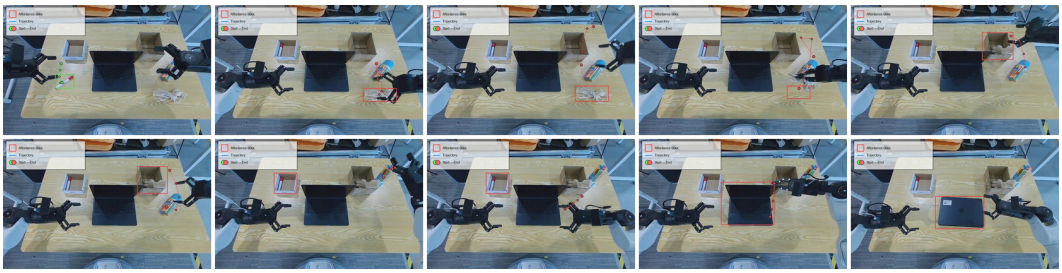


Figure 4: **Recovery from a failed tissue grasp (Base trial).** Head-camera frames run from left to right across the top row, then the bottom row. After the WORKER fails to lift the tissue, the PERCEIVER observes the failed attempt and plans another grasp. The WORKER follows the updated guidance, picks up the tissue, and places it in the bin. The remaining frames show the WORKER standing the marker box upright and closing the laptop to complete the task.

from the observed scene as the objects and remaining work change. The PERCEIVER updates its subtask and target predictions to reflect these changes. The results and rollout observations suggest that CoT conveys this scene generalization to the WORKER, which executes the specified operation and target. Moved remains the hardest condition: the tested pose changes cause greater difficulty than the appearance and category changes in Object and Category.

Fig. 4 shows the PERCEIVER and WORKER recovering from a failed tissue grasp without operator help. The WORKER fails to lift the tissue on its first attempt. The PERCEIVER observes that the tissue remains on the table and plans another grasp, which the WORKER executes successfully. **Our training data contain no sequences in which a failed grasp is followed by a retry. This recovery behavior emerges from the two models' interaction during deployment.** The new observation conveys the outcome of the previous action to the PERCEIVER. Replanning then directs the WORKER back to the unfinished subtask through the same CoT interface used for ordinary execution. The rendered CoT shows the return to tissue collection before the successful retry.

In a separate Multi trial, the subtask repeatedly requests paper collection, and the WORKER collects separated pieces despite incorrect geometry (Fig. 6 in the appendix). This example shows that execution can respond to the observed state while geometric guidance remains incorrect, consistent with the reduced sensitivity to guidance errors measured on fixed states in Section 5.1.3.

Our supplementary videos include full rollouts with rendered CoT, showing retries after subtask failures and generalization to novel object categories.

6 LIMITATIONS AND FUTURE WORK

Resource limits prevented joint pretraining of the PERCEIVER and WORKER within a single model. Future work will explore a mixture-of-transformers architecture that combines a VLM with an action expert, with joint pretraining of both components.

7 CONCLUSION

We analyzed when a learned action policy benefits from generated guidance and derived a bound relating training error, input coverage, and conditional drift to deployment error. Following this analysis, we trained a PERCEIVER–WORKER VLA on causally generated, annotated, perturbed, and missing guidance with explicit ages, and used it with asynchronous guidance refresh. Our method achieves 99.0% mean success on LIBERO, 25.04% success on held-out RoboTwin 2.0 tasks, and higher overall task scores than $\pi_{0.5}$ in all seven real-robot conditions. The observed retry shows that the PERCEIVER and WORKER can recover from a failed action through a sequence absent from our training demonstrations. On fixed RoboTwin states, the policy maintains low action error under wrong, stale, and missing CoT guidance. These results support training action policies with mixed CoT sources and temporal jitter to handle imperfect guidance during asynchronous deployment.

AI USE STATEMENT

We used generative AI tools to revise manuscript text, discuss research ideas, assist with code implementation, and prepare visualizations. We also used a vision–language model to generate CoT training annotations (Appendix F.1). The authors retain responsibility for the research decisions, verification of claims and results, and all submitted text, code, and supplementary materials.

ETHICS STATEMENT

This work studies vision–language–action policies for robot manipulation using existing demonstration datasets, simulation benchmarks, and real-robot desktop-cleaning experiments. Incorrect or outdated guidance can cause collisions and damage to nearby objects. Deployment around people should include safety testing for the intended tasks, human supervision, and mechanisms to stop execution. Camera observations may contain personal information, so data collection and release should protect privacy and follow the terms of the source datasets and pretrained models.

REPRODUCIBILITY STATEMENT

Appendix A states the assumptions and proves the theoretical results. Appendices C–E describe the task splits, evaluation protocols, baselines, ablations, and real-robot scoring criteria. Appendix F specifies data annotation, model training, guidance source probabilities and age sampling, asynchronous inference, and prompt templates. We will publicly release our training code and CoT-annotated data.

REFERENCES

AgiBot-World-Contributors, Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, et al. AgiBot World Colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669v4*, 2025. doi: 10.48550/arXiv.2503.06669. URL https://arxiv.org/abs/2503.06669v4.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631v2*, 2025. doi: 10.48550/arXiv.2511.21631. URL https://arxiv.org/abs/2511.21631v2.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164v4*, 2024. doi: 10.48550/arXiv.2410.24164. URL https://arxiv.org/abs/2410.24164v4.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054v1*, 2025. doi: 10.48550/arXiv.2504.16054. URL https://arxiv.org/abs/2504.16054v1.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818v1*, 2023. doi: 10.48550/arXiv.2307.15818. URL https://arxiv.org/abs/2307.15818v1.

ByteDance Seed. Seed2.0 model card: Towards intelligence frontier for real-world complexity. https://github.com/ByteDance-Seed/Seed2.0, 2026. URL https://github.com/ByteDance-Seed/Seed2.0.

Remi Cadene, Simon Alibert, Francesco Capuano, Michel Aractingi, Adil Zouitine, et al. LeRobot: An open-source library for end-to-end robot learning. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/pdf/64e1bfd102f15f5d0e838e4865a9e04488761851.pdf.

Tianxing Chen et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088v2*, 2025. doi: 10.48550/arXiv.2506.18088. URL https://arxiv.org/abs/2506.18088v2.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://roboticsproceedings.org/rss19/p026.html.

Zhekai Duan, Yuan Zhang, Shikai Geng, Gaowen Liu, Joschka Boedecker, and Chris Xiaoxuan Lu. Fast ECoT: Efficient embodied chain-of-thought via thoughts reuse. *arXiv preprint arXiv:2506.07639v2*, 2025. doi: 10.48550/arXiv.2506.07639. URL https://arxiv.org/abs/2506.07639v2.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, et al. LIBERO-Plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 38574–38583, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Fei_LIBERO-Plus_A_Progressive_Robustness_Benchmark_for_Visual-Language-Action_Models_CVPR_2026_paper.html.

Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, et al. RT-Trajectory: Robotic task generalization via hindsight trajectory sketches. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/c383e44d9a878d1982d9abb838bd5d8a-Abstract-Conference.html.

Kyle B. Hatch, Ashwin Balakrishna, Oier Mees, Suraj Nair, Seohong Park, Blake Wulfe, Masha Itkina, Benjamin Eysenbach, Sergey Levine, Thomas Kollar, and Benjamin Burchfiel. GHIL-Glue: Hierarchical control with filtered subgoal images. *arXiv preprint arXiv:2410.20018v1*, 2024. doi: 10.48550/arXiv.2410.20018. URL https://arxiv.org/abs/2410.20018v1.

Noriaki Hirose, Catherine Glossop, Dhruv Shah, and Sergey Levine. AsyncVLA: An asynchronous VLA for FAST and robust navigation on the edge. *arXiv preprint arXiv:2602.13476v1*, 2026. doi: 10.48550/arXiv.2602.13476. URL https://arxiv.org/abs/2602.13476v1.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/pdf?id=nZeVKeeFYf9.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Yu-Chiang Frank Wang, and Fu-En Yang. Thinkact: Vision-language-action reasoning via reinforced visual latent planning. *arXiv preprint arXiv:2507.16815v2*, 2025. doi: 10.48550/arXiv.2507.16815. URL https://arxiv.org/abs/2507.16815v2.

Zhiyu Huang, Yun Zhang, Johnson Liu, Rui Song, Chen Tang, and Jiaqi Ma. TIC-VLA: A think-in-control vision-language-action model for robot navigation in dynamic environments. *arXiv preprint arXiv:2602.02459v2*, 2026. doi: 10.48550/arXiv.2602.02459. URL https://arxiv.org/abs/2602.02459v2.

Tao Jiang, Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Jianning Cui, Xiao Liu, et al. Galaxea open-world dataset and G0 dual-system VLA model. *arXiv preprint arXiv:2509.00576v1*, 2025. doi: 10.48550/arXiv.2509.00576. URL https://arxiv.org/abs/2509.00576v1.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, et al. DROID: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945v2*, 2024. doi: 10.48550/arXiv.2403.12945. URL https://arxiv.org/abs/2403.12945v2.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, et al. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246v3*, 2024. doi: 10.48550/arXiv.2406.09246. URL https://arxiv.org/abs/2406.09246v3.

He Kong, Zengjue Chen, Qi Wang, Qianli Xing, Runliang Niu, Peidong Liu, Jiawei Li, Shiqi Wang, and Yi Chang. Beyond flat policies: Hierarchical post-training for embodied agents in robotic manipulation. *arXiv preprint arXiv:2608.05999v1*, 2026. doi: 10.48550/arXiv.2608.05999. URL https://arxiv.org/abs/2608.05999v1.

Jinming Li, Yichen Zhu, Zhibin Tang, Junjie Wen, Minjie Zhu, Xiaoyu Liu, Chengmeng Li, Ran Cheng, Yaxin Peng, Yan Peng, and Feifei Feng. CoA-VLA: Improving vision-language-action models via visual-text chain-of-affordance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9759–9769, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Li_CoA-VLA_Improving_Vision-Language-Action_Models_via_Visual-Text_Chain-of-Affordance_ICCV_2025_paper.html.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, et al. CogACT: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650v1*, 2024. doi: 10.48550/arXiv.2411.19650. URL https://arxiv.org/abs/2411.19650v1.

Zhixuan Liang, Yizhuo Li, Tianshuo Yang, Chengyue Wu, Sitong Mao, Liuao Pei, Xiaokang Yang, Jiangmiao Pang, Yao Mu, and Ping Luo. Discrete Diffusion VLA: Bringing discrete diffusion to action decoding in vision-language-action policies. *arXiv preprint arXiv:2508.20072v1*, 2025. doi: 10.48550/arXiv.2508.20072. URL https://arxiv.org/abs/2508.20072v1.

Yue Liao, Pengfei Zhou, Siyuan Huang, Donglin Yang, Shengcong Chen, Yuxin Jiang, Yue Hu, Jingbin Cai, Si Liu, Jianlan Luo, et al. Genie Envisioner: A unified world foundation platform for robotic manipulation. *arXiv preprint arXiv:2508.05635v3*, 2025. doi: 10.48550/arXiv.2508.05635. URL https://arxiv.org/abs/2508.05635v3.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/pdf?id=PqvMRDCJT9t.

Bo Liu, Yifeng Zhu, et al. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Zhuoyang Liu, Jiaming Liu, Hao Chen, Jiale Yu, Ziyu Guo, Chengkai Hou, et al. LaST_0: Latent spatio-temporal chain-of-thought for robotic vision-language-action model. *arXiv preprint arXiv:2601.05248v4*, 2026. doi: 10.48550/arXiv.2601.05248. URL https://arxiv.org/abs/2601.05248v4.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/pdf?id=Bkg6RiCqY7.

Qi Lv, Weijie Kong, Hao Li, Jia Zeng, Zherui Qiu, Delin Qu, Haoming Song, Qizhi Chen, Xiang Deng, and Jiangmiao Pang. F1: A vision-language-action model bridging understanding and generation to actions. *arXiv preprint arXiv:2509.06951v2*, 2025. doi: 10.48550/arXiv.2509.06951. URL https://arxiv.org/abs/2509.06951v2.

Pietro Mazzaglia, Cansu Sancaktar, Markus Peschl, and Daniel Dijkman. Hybrid training for vision-language-action models. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=IBJtOltTbx.

Soroush Nasiriany, Sean Kirmani, Tianli Ding, Laura Smith, Yuke Zhu, Danny Driess, Dorsa Sadigh, and Ted Xiao. RT-Affordance: Affordances are versatile intermediate representations for robot manipulation. *arXiv preprint arXiv:2411.02704v1*, 2024. doi: 10.48550/arXiv.2411.02704. URL https://arxiv.org/abs/2411.02704v1.

Daojie Peng, Fulong Ma, Bingtao Wang, Sheng Wang, and Jun Ma. Latency-tolerant cloud-edge collaborative vision-language-action models via emergent representational specialization. *arXiv preprint arXiv:2608.00569v3*, 2026a. doi: 10.48550/arXiv.2608.00569. URL https://arxiv.org/abs/2608.00569v3.

Jiaqi Peng, Xiqian Yu, Delin Feng, Yuqiang Yang, Wenzhe Cai, Jing Xiong, Ganlin Yang, Jinliang Zheng, Jiafei Cao, Xueyuan Wei, Jiangmiao Pang, Yuan Shen, and Tai Wang. Cortex:

A bidirectionally aligned embodied agent framework for long-horizon manipulation. *arXiv preprint arXiv:2607.05377v1*, 2026b. doi: 10.48550/arXiv.2607.05377. URL https://arxiv.org/abs/2607.05377v1.

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:2501.09747v1*, 2025. doi: 10.48550/arXiv.2501.09747. URL https://arxiv.org/abs/2501.09747v1.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236v2*, 2025. doi: 10.48550/arXiv.2508.19236. URL https://arxiv.org/abs/2508.19236v2.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. VLA-JEPA: Enhancing vision-language-action model with latent world model. *arXiv preprint arXiv:2602.10098v2*, 2026a. doi: 10.48550/arXiv.2602.10098. URL https://arxiv.org/abs/2602.10098v2.

Nan Sun, Yuan Zhang, Yongkun Yang, Wentao Zhao, Peiyan Li, Jun Guo, et al. Revisiting embodied chain-of-thought for generalizable robot manipulation. *arXiv preprint arXiv:2606.03784v2*, 2026b. doi: 10.48550/arXiv.2606.03784. URL https://arxiv.org/abs/2606.03784v2.

Jiaming Tang, Yufei Sun, Yilong Zhao, Shang Yang, Yujun Lin, et al. VLASH: Real-time VLAs via future-state-aware asynchronous inference. *arXiv preprint arXiv:2512.01031v2*, 2025. doi: 10.48550/arXiv.2512.01031. URL https://arxiv.org/abs/2512.01031v2.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, et al. VLA-Adapter: An effective paradigm for tiny-scale vision-language-action model. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40 (22):18638–18646, 2026. doi: 10.1609/aaai.v40i22.38931. URL https://ojs.aaai.org/index.php/AAAI/article/view/38931.

Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model. *arXiv preprint arXiv:2506.19850v1*, 2025. doi: 10.48550/arXiv.2506.19850. URL https://arxiv.org/abs/2506.19850v1.

Yifu Xiong, Wenhao Yu, Jiaxuan Lin, Bojun Zou, Jiahao Li, Lu Zhang, Yanyong Zhang, and Jianmin Ji. CAC-VLA: Context-gated action conditioning for vision-language-action models. *arXiv preprint arXiv:2607.04816v2*, 2026. doi: 10.48550/arXiv.2607.04816. URL https://arxiv.org/abs/2607.04816v2.

Jiarui Yang, Wen Huang, Jiale Zhang, Maowei Hu, and Hang Guo. In-Context VLA: Endowing vision-language-action models with language via in-context post-training and agentic tool use. *arXiv preprint arXiv:2608.05738v1*, 2026. doi: 10.48550/arXiv.2608.05738. URL https://arxiv.org/abs/2608.05738v1.

Cheng Yin, Yankai Lin, Wang Xu, Sikyuen Tam, Xiangrui Zeng, Zhiyuan Liu, and Zhouping Yin. DeepThinkVLA: Enhancing reasoning capability of vision-language-action models. *arXiv preprint arXiv:2511.15669v3*, 2025. doi: 10.48550/arXiv.2511.15669. URL https://arxiv.org/abs/2511.15669v3.

Qize Yu, Jiadi You, Yuran Wang, Jiaqi Liang, Bowen Ping, Yang Tian, Yue Chen, Minghong Cai, Zeying Gong, Ruihai Wu, Yinchuan Li, Junwei Liang, and Yingcong Chen. AffordanceVLA: A vision-language-action model empowering action generation through affordance-aware understanding. *arXiv preprint arXiv:2606.06155v1*, 2026. doi: 10.48550/arXiv.2606.06155. URL https://arxiv.org/abs/2606.06155v1.

Yifu Yuan, Yaoting Huang, Xianze Yao, Yutong Li, Shuoheng Zhang, Linqi Han, et al. Embodied-R1.5: Evolving physical intelligence via embodied foundation models. *arXiv preprint arXiv:2606.11324v2*, 2026. doi: 10.48550/arXiv.2606.11324. URL https://arxiv.org/abs/2606.11324v2.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693v3*, 2024. doi: 10.48550/arXiv.2407.08693. URL https://arxiv.org/abs/2407.08693v3.

Andy Zhai, Brae Liu, Bruno Fang, Chalse Cai, Ellie Ma, Ethan Yin, Hao Wang, Hugo Zhou, James Wang, Lights Shi, Lucy Liang, Make Wang, Qian Wang, Roy Gan, Ryan Yu, Shalfun Li, Starrick Liu, Sylas Chen, Vincent Chen, and Zach Xu. Igniting VLMs toward the embodied space. *arXiv preprint arXiv:2509.11766v1*, 2025. doi: 10.48550/arXiv.2509.11766. URL https://arxiv.org/abs/2509.11766v1.

Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, Fan Lu, He Wang, et al. DreamVLA: a vision-language-action model dreamed with comprehensive world knowledge. *arXiv preprint arXiv:2507.04447v3*, 2025. doi: 10.48550/arXiv.2507.04447. URL https://arxiv.org/abs/2507.04447v3.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Ming-Yu Liu, Donglai Xiang, Gordon Wetzstein, and Tsung-Yi Lin. CoT-VLA: Visual chain-of-thought reasoning for vision-language-action models. *arXiv preprint arXiv:2503.22020v1*, 2025. doi: 10.48550/arXiv.2503.22020. URL https://arxiv.org/abs/2503.22020v1.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, et al. TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345v3*, 2024. doi: 10.48550/arXiv.2412.10345. URL https://arxiv.org/abs/2412.10345v3.

Linqing Zhong, Yi Liu, Yifei Wei, Ziyu Xiong, Maoqing Yao, Si Liu, et al. ACoT-VLA: Action chain-of-thought for vision-language-action models. *arXiv preprint arXiv:2601.11404v2*, 2026. doi: 10.48550/arXiv.2601.11404. URL https://arxiv.org/abs/2601.11404v2.

Teqiang Zou, Hongliang Zeng, Yuxuan Nong, Yifan Li, Kehui Liu, Haotian Yang, et al. Asynchronous FAST-slow vision-language-action policies for whole-body robotic manipulation. *arXiv preprint arXiv:2512.20188v1*, 2025. doi: 10.48550/arXiv.2512.20188. URL https://arxiv.org/abs/2512.20188v1.

APPENDIX CONTENTS

A Definitions, Proofs, and Training Implications for Section 3 **16**
A.1 Setup and notation 16
A.2 Exact comparison (Section 3.1) 16
A.3 Bounding the guided excess risk (Section 3.2) 17
A.4 How the bound guides training 18

B Extended Related Work **19**
B.1 Semantic Guidance for Generative Robot Policies 19
B.2 Asynchronous Guidance under Deployment Shift 19

C Additional Experimental Setup **19**

D Real-Robot Setup and Protocol **20**

E Details of the Guidance-Shift Test **21**

F Detailed Method **23**
F.1 Data Annotation 23
F.2 Perceiver Training 27
F.3 Predicted CoT Generation 27
F.4 Worker Training 28
F.5 Guidance Hyperparameters 28
F.6 Asynchronous Inference 29
F.7 Prompt Templates 29

Real-robot videos. We provide videos of our real-robot experiments in the supplementary materials. See the `README.md` file in the supplementary materials for details.

A Definitions, Proofs, and Training Implications for Section 3

This appendix provides the full definitions, proofs, and training implications for Section 3. Appendix A.1 fixes the notation, Appendix A.2 derives the exact learned-policy comparison, and Appendix A.3 proves the transfer bound and its consequence at the training optimum. Appendix A.4 explains how the bound informs source mixtures, temporal jitter, and demonstration count.

A.1 Setup and notation

We first make the guidance input precise. Fix a physical control time t, the time at which the Worker queries the guidance to predict the current action chunk. The current input X_t contains the current images, instruction, and robot state, as in Section 3. The field index $j \in \mathcal{J} = \{\mathrm{sub}, \mathrm{aff}, \mathrm{traj}\}$ selects subtask, affordance, or trajectory guidance. Field $Z^j_{s_j}$ was generated from an observation at physical time $s_j \le t$, so its age at control time is $\Delta^j_t = t - s_j$. The guidance input

$$U_t = \left(Z^j_{s_j}, \Delta^j_t\right)_{j \in \mathcal{J}}, \tag{3}$$

therefore contains one content–age pair per field; an unavailable pair is denoted by $\varnothing$. Below, $X = X_t$ and $U = U_t$ suppress the fixed control-time subscript. The tuple Z collects field contents and missing markers, and Δ collects the ages of available fields, giving $U = (Z, \Delta)$.

The regression target comes from the flow-matching path. Let A be a demonstrated action chunk, $\xi \sim \mathcal{N}(0, I)$ independent Gaussian noise with identity covariance I, and $\tau \sim \rho$ an independently sampled flow time with sampling law ρ on $[0, 1]$. The noisy action, target velocity, and common regression input are

$$A_\tau = \tau\xi + (1-\tau)A, \qquad V = \xi - A, \qquad S = (A_\tau, \tau, X). \tag{4}$$

Generation runs from noise ($\tau = 1$) to actions ($\tau = 0$); Δ denotes physical age.

Risks are taken in one of two domains. The index $D \in \{\mathrm{tr}, \mathrm{te}\}$ denotes training or deployment. The symbol $\mathbb{E}_D$ averages over actions, inputs, sampled noise, and flow time in that domain, and p_D is its marginal law of the complete regression input (S, U). As in Section 3, both domains share the law of (X, A, ξ, τ) and differ in guidance. The target and learned fields have finite second moments in both domains. For a field v, the superscript FM identifies squared flow-matching prediction risk:

$$R^{\mathrm{FM}}_D(v; U) = \mathbb{E}_D\left[\|V - v(S, U)\|_2^2\right]. \tag{5}$$

The argument after the semicolon identifies the optional guidance input; for a field $v(S)$ that ignores guidance, we write $R^{\mathrm{FM}}_D(v)$. Appendices A.2 and A.3 hold for any two such domains. Section 3 uses a fixed demonstration population while varying the guidance law.

The optimal fields and their risks complete the notation. The superscript $\star$ marks a minimum-risk conditional-mean field, with its arguments naming the inputs it uses. Under squared loss, the guided and observation-only optima are $v^\star_{U,D}(S, U) = \mathbb{E}_D[V \mid S, U]$ and $v^\star_{X,D}(S) = \mathbb{E}_D[V \mid S]$, with minimum risks $B^{\mathrm{FM}}_D(X, U)$ and $B^{\mathrm{FM}}_D(X)$. The arguments X, U in B^{FM}_D label the available observation conditions; the noisy action and flow time in S are common to both risks. In norms below, $v^\star_{U,D}$ without displayed arguments abbreviates the guided field $v^\star_{U,D}(S, U)$, and $\|f\|^2_{L_2(p_D)} = \mathbb{E}_D[\|f(S, U)\|_2^2]$ for a field f. Hats mark learned fields, and stars mark population optima. On fields and excess risks, the subscripts U and X identify the guided and observation-only policies, respectively. The index D on an optimal field identifies the distribution defining its conditional mean. We evaluate each learned policy under both distributions without updating its parameters.

A.2 Exact comparison (Section 3.1)

The exact comparison rests on one identity. For a square-integrable field v, write $V - v = (V - v^\star_{U,D}) + (v^\star_{U,D} - v)$. The first term has conditional mean zero given (S, U), while the second is measurable with respect to (S, U). Their expected inner product therefore vanishes:

$$R^{\mathrm{FM}}_D(v; U) = B^{\mathrm{FM}}_D(X, U) + \|v - v^\star_{U,D}\|^2_{L_2(p_D)}. \tag{6}$$

Applied to the observation-only optimum $v^\star_{X,D}(S)$, this identity gives the information gain in closed form:

$$G_D(U) = B^{\mathrm{FM}}_D(X) - B^{\mathrm{FM}}_D(X,U) = \mathbb{E}_D\big[\|v^\star_{U,D}(S,U) - v^\star_{X,D}(S)\|_2^2\big] \geq 0. \tag{7}$$

In the deployment domain, this becomes

$$G_{\mathrm{te}}(U) := B^{\mathrm{FM}}_{\mathrm{te}}(X) - B^{\mathrm{FM}}_{\mathrm{te}}(X,U) \geq 0. \tag{8}$$

The gain is therefore zero exactly when $v^\star_{U,D}(S,U) = v^\star_{X,D}(S)$ almost surely. Two such cases are U determined by X, and U and V conditionally independent given S.

Applied to the fixed learned fields $\hat{v}_U(S,U)$ and $\hat{v}_X(S)$, the same identity expresses the excess risks of Section 3.1 as squared distances:

$$e_{U,\mathrm{te}} := R^{\mathrm{FM}}_{\mathrm{te}}(\hat{v}_U;U) - B^{\mathrm{FM}}_{\mathrm{te}}(X,U) = \|\hat{v}_U - v^\star_{U,\mathrm{te}}\|^2_{L_2(p_{\mathrm{te}})},$$
$$e_{X,\mathrm{te}} := R^{\mathrm{FM}}_{\mathrm{te}}(\hat{v}_X) - B^{\mathrm{FM}}_{\mathrm{te}}(X) = \|\hat{v}_X - v^\star_{X,\mathrm{te}}\|^2_{L_2(p_{\mathrm{te}})}.$$

Both differences use the deployment domain. Subtracting the observation-only optimal risk from the guided learned risk gives

$$R^{\mathrm{FM}}_{\mathrm{te}}(\hat{v}_U;U) - B^{\mathrm{FM}}_{\mathrm{te}}(X) = -G_{\mathrm{te}}(U) + e_{U,\mathrm{te}}. \tag{9}$$

Subtracting $e_{X,\mathrm{te}}$ from Eq. (9) proves the exact learned-policy comparison in Eq. (1).

Guidance can help a learned policy even when $G_{\mathrm{te}}(U) = 0$. If U is a deterministic function of X, it leaves the optimal prediction unchanged. An explicit target region may still make task-relevant image information easier for a learned policy to use. Any resulting reduction in risk appears as $e_{U,\mathrm{te}} < e_{X,\mathrm{te}}$. Thus, guidance quality depends both on the information it provides and on the policy's ability to use it under the deployment distribution.

A.3 Bounding the guided excess risk (Section 3.2)

The bound compares the guided field with the training optimum and transfers that error to deployment. Its three terms are defined as follows. The training excess risk $\epsilon_{\mathrm{tr}}(\hat{v}_U;U)$ uses the training optimum as reference, and its arguments identify the learned guided field and its guidance input. The coefficient $\beta(U)$ measures coverage of the full guided input (S,U):

$$\epsilon_{\mathrm{tr}}(\hat{v}_U;U) = \|\hat{v}_U - v^\star_{U,\mathrm{tr}}\|^2_{L_2(p_{\mathrm{tr}})}, \tag{10}$$

$$\beta(U) = \begin{cases} \left\| \dfrac{dp_{\mathrm{te}}}{dp_{\mathrm{tr}}} \right\|_{L_\infty(p_{\mathrm{tr}})}, & p_{\mathrm{te}} \ll p_{\mathrm{tr}}, \\ \infty, & \text{otherwise.} \end{cases} \tag{11}$$

The expression for $\beta(U)$ equals the measure-domination definition in Tab. 1, with $\inf \varnothing = \infty$. The training excess risk also equals $R^{\mathrm{FM}}_{\mathrm{tr}}(\hat{v}_U;U) - B^{\mathrm{FM}}_{\mathrm{tr}}(X,U)$ by Eq. (6). Thus ϵ_{tr} averages excess error over the training population and can include finite-data, capacity, and optimization error. The notation $p_{\mathrm{te}} \ll p_{\mathrm{tr}}$ means that every set with zero training probability also has zero deployment probability. The derivative $dp_{\mathrm{te}}/dp_{\mathrm{tr}}$ is the deployment-to-training density ratio, and its $L_\infty(p_{\mathrm{tr}})$ norm is the smallest bound that holds outside a set of zero training probability. Finite $\beta(U)$ bounds the ratio for the complete input (S,U), which contains the noisy action, flow time, current observations, guidance contents, missing markers, and ages.

Coverage also makes the training field usable at deployment inputs. For finite $\beta(U)$, the training conditional field is defined deployment-almost everywhere, and the conditional drift is

$$\Gamma_{\mathrm{te}}(U) = \|v^\star_{U,\mathrm{tr}} - v^\star_{U,\mathrm{te}}\|^2_{L_2(p_{\mathrm{te}})}. \tag{12}$$

The subscript te indicates that this squared difference is averaged over deployment inputs, and the argument U identifies the guidance input. The drift compares conditional means at the same input. Like $G_D(U)$ and $\beta(U)$, it is a property of the domain distributions. Guidance values can therefore occur in both domains while implying different target predictions, even if the base distribution of (X,A) is fixed.

Proposition 1 in Section 3.2 bounds the guided deployment excess risk by C_U. Combining that bound with Eq. (1) yields

$$R^{\mathrm{FM}}_{\mathrm{te}}(\hat{v}_U; U) - R^{\mathrm{FM}}_{\mathrm{te}}(\hat{v}_X) \le -G_{\mathrm{te}}(U) + C_U - e_{X,\mathrm{te}}. \tag{13}$$

Hence $C_U < G_{\mathrm{te}}(U) + e_{X,\mathrm{te}}$ suffices for lower deployment risk than the learned observation-only policy, and the stronger condition $C_U < G_{\mathrm{te}}(U)$ suffices to beat the observation-only optimum. The exact criterion remains $e_{U,\mathrm{te}} < G_{\mathrm{te}}(U) + e_{X,\mathrm{te}}$.

Proof of Proposition 1. Add and subtract the training-optimal field. The L_2 triangle inequality and change of measure give

$$\sqrt{e_{U,\mathrm{te}}} \le \|\hat{v}_U - v^\star_{U,\mathrm{tr}}\|_{L_2(p_{\mathrm{te}})} + \sqrt{\Gamma_{\mathrm{te}}(U)}, \tag{14}$$

$$\begin{aligned}\|\hat{v}_U - v^\star_{U,\mathrm{tr}}\|^2_{L_2(p_{\mathrm{te}})} &= \mathbb{E}_{\mathrm{tr}}\left[\frac{dp_{\mathrm{te}}}{dp_{\mathrm{tr}}}(S,U)\|\hat{v}_U(S,U) - v^\star_{U,\mathrm{tr}}(S,U)\|_2^2\right] \\ &\le \beta(U)\epsilon_{\mathrm{tr}}(\hat{v}_U; U).\end{aligned} \tag{15}$$

Combining these inequalities and squaring proves Eq. (2). Substitution into Eq. (1) proves Eq. (13). The assumed finite density-ratio bound controls the change of measure in Eq. (15). □

At the population training optimum, the bound becomes an equality.

Corollary 1 (Conditional drift at the training optimum)**.** *Under the assumptions of Proposition 1, if* $\epsilon_{\mathrm{tr}}(\hat{v}_U; U) = 0$*, then*

$$\begin{aligned}e_{U,\mathrm{te}} &= \Gamma_{\mathrm{te}}(U), \\ R^{\mathrm{FM}}_{\mathrm{te}}(\hat{v}_U; U) - R^{\mathrm{FM}}_{\mathrm{te}}(\hat{v}_X) &= \Gamma_{\mathrm{te}}(U) - G_{\mathrm{te}}(U) - e_{X,\mathrm{te}}.\end{aligned} \tag{16}$$

Proof of Corollary 1. The condition $\epsilon_{\mathrm{tr}} = 0$ implies $\hat{v}_U = v^\star_{U,\mathrm{tr}}$ training-almost everywhere. Absolute continuity transfers this equality to deployment, which proves the first identity; Eq. (1) then gives the second. The training risk at this population optimum equals $B^{\mathrm{FM}}_{\mathrm{tr}}(X, U)$. □

A.4 How the bound guides training

By Eqs. (1) and (2), $C_U < G_{\mathrm{te}}(U) + e_{X,\mathrm{te}}$ suffices for guidance to improve deployment risk. The three terms in C_U point to distinct training choices under a fixed deployment distribution. Mixing annotated (GT), generated (PRED), noisy (NOISE), incorrect (ERR), and missing guidance gives the WORKER inputs it may encounter online. Giving underrepresented deployment inputs more training probability reduces their density ratios, with the aim of lowering the global coverage coefficient $\beta(U)$. We use PRED guidance to approximate the guidance–action relation induced by the deployed generator, targeting a smaller conditional drift $\Gamma_{\mathrm{te}}(U)$. Source weights allocate training probability across input conditions; increasing one reduces the share available to others. Unrealistic corruptions can increase $\Gamma_{\mathrm{te}}(U)$; broader mixtures may need more capacity or training to reduce $\epsilon_{\mathrm{tr}}(\hat{v}_U; U)$.

Temporal jitter changes the content as well as the age of guidance: for each field, it selects CoT produced from an earlier source observation and supplies the elapsed physical time to the current action prediction. The current observation and supervised action stay fixed. This makes delayed content and its age available during training, with the aim of reducing the coverage gap for stale inputs in $\beta(U)$. When the sampled delays and source contents resemble deployment, they can also bring the training conditional action relation closer to deployment and reduce $\Gamma_{\mathrm{te}}(U)$. The delay range must reflect the use case: excessive weight on old content can displace fresh inputs, while unrealistic content–delay pairs may increase $\Gamma_{\mathrm{te}}(U)$ by changing the conditional relation and make $\epsilon_{\mathrm{tr}}(\hat{v}_U; U)$ harder to reduce.

More independent demonstrations from the same population and the same CoT sampling recipe can reduce the finite-data contribution to $\epsilon_{\mathrm{tr}}(\hat{v}_U; U)$, given sufficient optimization to fit the training-optimal field. With the population training and deployment laws fixed, adding samples leaves $\beta(U)$ and $\Gamma_{\mathrm{te}}(U)$ unchanged. Changing the CoT source mix changes the population training law and can affect both coverage and conditional drift.

B Extended Related Work

B.1 Semantic Guidance for Generative Robot Policies

RT-2 (Brohan et al., 2023) and OpenVLA (Kim et al., 2024) adapt vision–language representations to robot actions, while Diffusion Policy (Chi et al., 2023), CogACT (Li et al., 2024), and π_0 (Black et al., 2024) model continuous actions generatively. Such policies can be conditioned on intermediate guidance, including embodied reasoning (Zawalski et al., 2024), trajectory or affordance structure (Gu et al., 2024; Zheng et al., 2024; Nasiriany et al., 2024; Li et al., 2025), and predicted visual subgoals (Zhao et al., 2025). Our subtask, affordance, and trajectory fields follow these examples.

Several studies measure how the quality of generated guidance affects control. Action-related fields can help, whereas noisy grounding or mandatory reasoning can be harmful (Mazzaglia et al., 2026; Sun et al., 2026b). DeepThinkVLA studies the decoding and causal alignment required for CoT to affect task success (Yin et al., 2025). GHIL-Glue addresses errors in predicted subgoals (Hatch et al., 2024). These results show that generated guidance can help or harm a policy. Our analysis asks when the harm from using guidance exceeds the information it provides.

Other methods adapt the action policy to generated guidance. In-Context VLA consumes grounded external language with action-only supervision (Yang et al., 2026). HiRoC aligns an executor to planner-generated subgoals (Kong et al., 2026). Cortex standardizes executable subtask plans for long-horizon manipulation (Peng et al., 2026b). CAC-VLA learns a gate over predicted latent-action context (Xiong et al., 2026). We share the use of generated guidance in action training. Our focus is the changing reliability and age of several explicit guidance fields, and how the training mixture shapes the learned policy's risk under these changes. Our WORKER learns its reliance on each field from that mixture.

B.2 Asynchronous Guidance under Deployment Shift

Fast ECoT (Duan et al., 2025) and VLASH (Tang et al., 2025) reduce reasoning or execution latency. AsyncVLA (Hirose et al., 2026) combines delayed guidance with current observations. DuoCore-FS (Zou et al., 2025) couples an asynchronous semantic buffer to a flow-based manipulation controller. TIC-VLA (Huang et al., 2026) supplies latency and ego-motion metadata and trains with delayed semantic inputs for navigation. CloudEdgeVLA trains its slow representation by assigning current action targets to both fresh and randomly delayed cloud inputs (Peng et al., 2026a). Training with delayed inputs and current action targets therefore has precedent. We apply it to three manipulation guidance fields that refresh separately, each with its own physical age, and combine delay with wrong and missing guidance in the same training mixture.

C Additional Experimental Setup

LIBERO. We report task success on LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long (Liu et al., 2023). Our evaluation uses 500 rollouts per task and one joint policy over the four suites. Each suite has ten tasks, giving 5,000 rollouts per suite and 20,000 in total. The Long suite contains the ten long-horizon tasks from LIBERO-100. The suites use disjoint demonstration sets, and the mean in Tab. 3 averages their success rates.

RoboTwin 2.0. We split 50 tasks into 25 training and 25 held-out tasks, with 50 demonstrations per training task (1,250 in total) and 100 rollouts per evaluation task. Each split therefore contains 2,500 evaluation rollouts. The frozen evaluation uses clean scenes and unseen instruction variants, with the same expert-accepted scene seeds across methods. Tab. 4 reports two outcomes: Train measures performance on training tasks, and Test measures generalization to held-out tasks. The π_0, Wall-X (the LeRobot integration of WALL-OSS (Zhai et al., 2025)), and VLA-JEPA baselines use their LeRobot implementations (Cadene et al., 2026). Tab. 8 lists the tasks in each split.

Table 8: RoboTwin 2.0 task split. Columns list benchmark task identifiers in alphabetical order.

Training tasks (25)	Held-out tasks (25)
blocks_ranking_rgb	adjust_bottle
click_alarmclock	beat_block_hammer
dump_bin_bigbin	blocks_ranking_size
handover_block	click_bell
lift_pot	grab_roller
move_can_pot	handover_mic
move_pillbottle_pad	hanging_mug
open_laptop	move_playingcard_away
pick_diverse_bottles	move_stapler_pad
place_a2b_left	open_microwave
place_a2b_right	pick_dual_bottles
place_bread_basket	place_dual_shoes
place_bread_skillet	place_empty_cup
place_burger_fries	place_fan
place_can_basket	place_mouse_pad
place_cans_plasticbox	place_object_basket
place_container_plate	place_object_scale
place_shoe	place_object_stand
press_stapler	place_phone_stand
put_bottles_dustbin	put_object_cabinet
rotate_qrcode	shake_bottle_horizontally
scan_object	stack_bowls_three
shake_bottle	stack_bowls_two
stack_blocks_three	stamp_seal
stack_blocks_two	turn_switch

Baselines and metrics. On LIBERO, we evaluate $\pi_{0.5}$, Embodied-R1.5-VLA, and our method using one joint policy per method over the four suites and 500 rollouts per task. The remaining entries in Tab. 3 are taken from the cited papers and retain their original training and evaluation protocols. On RoboTwin 2.0, we evaluate π_0, observation-only $\pi_{0.5}$, Wall-X, VLA-JEPA, and our method on the same 25 training and 25 held-out tasks, using 100 rollouts per task and the shared scene seeds described above. The observation-only $\pi_{0.5}$ control follows the same continued-pretraining and downstream fine-tuning stages as our WORKER, without guidance input (Section 5.1). Closed-loop task success is the primary metric; we report the four LIBERO suites and their mean, and the RoboTwin Train and Test splits separately.

RoboTwin 2.0 ablations. F and A1–A4 start from the same general WORKER checkpoint after 30,000 updates and share the same PERCEIVER, fine-tuning data, budget, and evaluation denominators: 25 training tasks and 25 held-out tasks. All ablation changes apply only to downstream fine-tuning; continued pretraining is shared across variants. Subtask rewriting is disabled. The changes from F are: A1, causal GT-only guidance for both field types, with the other sources and random emptying disabled; A2, the numerical age hidden while source content and field presence are kept; A3, the extra history jitter removed while the actual age of sparse predictions is kept; A4, strict causal PRED-only subtask and geometry, with random emptying and annotation fallback disabled, retaining valid missing and abstention states.

A1 and A4 compare downstream guidance recipes: the mixture, causal GT-only, and strict Pred-only. A2 tests the numerical age input with field content and presence fixed. A3 tests extra WORKER jitter while retaining the actual age of sparse predictions. A4 keeps every training row and uses the predicted content or its valid missing state. Section 5.1.3 compares the same recipes under wrong, stale, and missing guidance with fixed observations.

D REAL-ROBOT SETUP AND PROTOCOL

Data and models. The demonstrations come from two desktop-cleaning tasks with 465 and 401 episodes; the evaluation uses the layout of the first task. Guidance labels are generated automatically

(Appendix F.1). The PERCEIVER is fine-tuned from the general PERCEIVER of Appendix F for two epochs. The WORKER is trained from the general WORKER on a 30,000-update schedule, and we evaluate its checkpoint after 6,000 updates; its subtask sources GT, PRED, ERR, and MISSING have probabilities 0.10, 0.75, 0.10, and 0.05, geometric fields use PRED with probability 0.60 and are empty with probability 0.05, and source lags are uniform on 0.1–1.0 s. The instruction is "Clean the desktop by putting every item in its place and closing the notebook." Actions are 14-dimensional end-effector deltas of both arms at 30 Hz, with chunks of 50 steps. We fine-tune the baseline from pretrained $\pi_{0.5}$ for 6,000 updates on the same demonstrations and actions without guidance.

Deployment. Both models run in bfloat16 on one RTX 5090 GPU. The WORKER reads the latest entry of each field as in Section 4; an entry older than a per-field limit of 1.5–3.0 s is treated as missing. Each chunk is converted to joint targets by inverse kinematics, and the robot executes its first 25 steps before the next query. A trial ends when all subtasks succeed or after 200 chunks.

Scene conditions. All conditions use the same table, background, cameras, lighting, start pose, and instruction. Base places the six objects as in the first frame of a reference demonstration. Object replaces the pen and the paper with items of different appearance that the gripper can hold. Category tests object categories absent from the demonstrations: a potato chip can replaces the pen holder, and a glue stick replaces the pen, while the paper, bin, marker box, and laptop are unchanged. Multi places three sheets of paper at registered positions. Moved places the pen, paper, box, and laptop at registered positions about 10 cm from Base within reach and rotates the paper, box, and laptop by randomly chosen angles of approximately $20°$–$40°$ relative to Base, with containers unchanged. Laptop done and Paper done start with the laptop closed and the paper in the bin.

Scoring and aggregation. The recorded batch contains five trials of each policy for each of Base, Object, Category, Multi, Moved, Laptop done, and Paper done. Each applicable subtask has a score in $[0, 1]$. Full completion means the pen is inside the holder, paper is inside the bin, the whole marker box is upright, or the laptop is closed, with the final state stable for 2 s. Multi permits partial paper credit: our policy's five recorded Paper scores are $1, 0.6, 0.6, 1, 1$, giving a mean of 0.84. The other recorded subtask scores are binary.

Tab. 7 reports each subtask's mean over its five trials, multiplied by 100. For each trial, the overall score is the mean over applicable subtasks; the table averages these trial scores and multiplies by 100. Base, Object, Category, Multi, and Moved have four applicable subtasks. Laptop done excludes Laptop, and Paper done excludes Paper, leaving three applicable subtasks. Their initially completed goals receive no score. Thus the table reports average task credit, including partial completion, rather than the fraction of trials completing every goal.

Recorded guidance stamps. The evaluation overlay logs, for every rendered frame, the sample index and the source stamp of each field that the WORKER reads. Across frames sampled every 10 s in a successful Base trial, the subtask field is 0.5–1.1 s old and the geometry field 1.0–2.1 s old; the oldest geometry entries coincide with stretches of rejected PERCEIVER updates (up to eight in a row), during which the last accepted guidance is held and its age keeps growing. These observed ages fall in or near the training range of source lags (uniform on 0.1–1.0 s per field), so the WORKER operates on the age distribution it was trained for.

E DETAILS OF THE GUIDANCE-SHIFT TEST

The analysis uses the `005000` WORKER checkpoints from the RoboTwin 2.0 ablations: F (complete recipe), A1 (GT-only), A3 (no age jitter), and A4 (Pred-only). All four start from the same general-training `030000` checkpoint and use the same RoboTwin train25 data and PERCEIVER checkpoint `000976`. We evaluate fixed states sampled from those training demonstrations. Each state keeps its current images, robot state, and instruction. All three fields take one status: correct GT, ERR, GT from 3 s earlier with its true age, or MISSING. ERR applies the training corruptions: a subtask from another time in the same episode, an affordance box shifted by 30% of the image size, and a trajectory shifted by 20–40% of the short image side and rotated by 15–$45°$. The metric is the root-mean-square distance between the first 25 predicted actions and the demonstrated actions over 14 joints. Each state uses four initial noises shared by all conditions and 10 Euler steps. We average



Figure 5: **Final state of every real-robot trial of our policy** (one row per condition, five trials per condition). The grey cell marks a trial whose recording is missing; its outcome is recorded in the trial log (score 1, pen only).

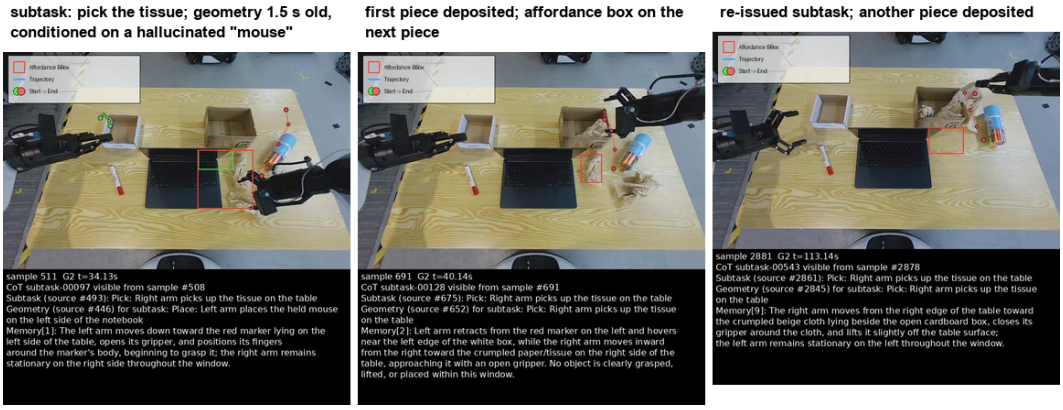


Figure 6: **Guidance during the piecewise recovery in a Multi trial.** Overlay frames of the trial; the text panel that the WORKER reads is attached below each frame. At the first grasp (left) the subtask field requests the tissue while the geometry field is 1.5 s old and conditioned on an earlier subtask that describes placing an object ("the held mouse") that is not on the table. The subtask field is re-issued after each deposit (middle, right) until the paper subtask succeeds.

states within each task and then average across tasks. Sampling states from training demonstrations holds the task distribution fixed and measures the response to changes in guidance.

Tab. 6 uses two state sets. Set (a) contains one state per field in each of 25 tasks, with guidance 0.3–0.48 s old. Set (b) contains 77 states in 23 tasks where old guidance differs from current guidance: 0.3 s after a subtask change, or during arm motion where the trajectory from 0.8 s earlier differs most from the current one. Set (a) compares F with A1 and A4, which differ in ERR and MISSING exposure

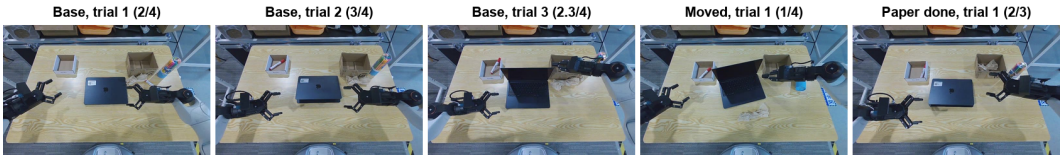


Figure 7: **Final state of the five recorded $\pi_{0.5}$ baseline trials.** In the three Base trials (left to right) the paper ends beside the bin, on the table, and hanging over the bin edge; in the Moved trial only the pen reaches the holder. The marker box is left leaning against the bin in the Base trials.

during fine-tuning. Set (b) compares F with A3, which differs only in age jitter. The observation-only $\pi_{0.5}$ row (Black et al., 2025) uses the same demonstrations with a different initialization and learning rate. It provides the observation-only reference for MISSING guidance.

ERR and MISSING fine-tuning reduce sensitivity to guidance errors while retaining the error level under correct guidance. With correct guidance, F, A1, and A4 have similar errors. ERR raises A1's error by 0.88 and A4's error by 0.43, while F changes by 0.01. Paired over tasks, F has lower error than A1 by 0.88 with ERR in 20 of 25 tasks, and by 0.66 with MISSING in 23 of 25 tasks. A3 shares F's sources and changes by at most 0.03 in set (a). A4 is fine-tuned on the PERCEIVER's own errors and remains sensitive to ERR. F includes ERR corruptions during fine-tuning, consistent with the coverage motivation for $\beta(U)$ in Eq. (2); A1 and A4 give this source zero weight at that stage.

The largest failure occurs when A1, fine-tuned without ERR and MISSING, receives stale guidance. A1 retains age jitter during fine-tuning, yet its error on set (b) rises from 2.02 with fresh guidance to 5.81 with 3 s old guidance. This error exceeds the observation-only reference by 3.54, and A1's error with MISSING guidance exceeds it by 1.09. F stays below the reference with stale and MISSING guidance. These recipe comparisons include initialization and guidance training.

Age jitter reduces the loss from stale guidance at a small cost with fresh guidance. Stale guidance raises A3's error by 0.42 and F's error by 0.11. The difference between these increases is 0.31 in favor of F, in 21 of 23 tasks. The difference is 0.40 on states after a subtask change and 0.07 on trajectory states. With fresh guidance, A3 has lower error than F by 0.13; with stale guidance, F has lower error by 0.17. Replacing the true age with 0.2 s while keeping the stale content changes each recipe's error on subtask states by at most 0.01. The measured change is tied to stale subtask content, while both recipes have similar errors across the tested numerical ages.

F DETAILED METHOD

This section collects data annotation, PERCEIVER SFT, predicted CoT generation, WORKER training, hyperparameters, asynchronous inference, and prompt templates. General training uses DROID and AgiBot World; downstream adaptation uses the target benchmark's demonstrations. Downstream source probabilities, prediction counts, age ranges, and training budgets are configured per benchmark.

F.1 DATA ANNOTATION

The annotation pipeline produces affordance, 2D trajectory, subtask, and memory labels for model training. We applied it to six datasets: DROID (Khazatsky et al., 2024), AgiBotWorld-Beta (AgiBot-World-Contributors et al., 2025), RoboTwin 2.0 (Chen et al., 2025), OpenGalaxy (the Galaxea Open-World Dataset (Jiang et al., 2025)), LIBERO (Liu et al., 2023), and LIBERO-Plus (Fei et al., 2026). This annotation collection is broader than the data used for each model; the training stages and their data are specified below.

F.1.1 OVERVIEW

The pipeline produces frame-level labels for affordances and 2D trajectories, phase-level subtask labels, and memory summaries for both event segments and fixed time windows. Affordance labels give 2D bounding boxes for task-relevant entities. The category *object* denotes an entity the robot approaches to grasp; *location* denotes the target region for a grasped entity. Trajectory labels give

the image-space end-effector position at each frame, from which future trajectories are sampled during training. Subtask labels describe an action, target object, and spatial relation. Memory labels summarize observed events within each segment or window. These four label types form the chain-of-thought (CoT) supervision used in the annotation codebase.

The pipeline queries a VLM on selected frames and uses post-processing to produce dense labels. When simulator state and camera calibration are available, it projects the end-effector position directly into image space. Model queries use greedy decoding, and the parser checks the JSON format. Incremental outputs and episode checkpoints record progress so interrupted runs can resume.

F.1.2 DATA SCALE

The annotated datasets include single- and dual-arm platforms. DROID contributes 74,740 episodes, LIBERO-Plus 14,347, LIBERO 1,693, and RoboTwin 2.0 2,500. We selected episodes from the two largest dual-arm real-robot collections according to scene richness. The AgiBotWorld-Beta subset contains 8,458 episodes. OpenGalaxy contains 22,384 raw episodes, of which approximately 7,800 were selected for annotation. Both geometric label types have frame-level records in all datasets.

F.1.3 ANNOTATION MODEL SELECTION

We compared commercial VLMs on a fixed evaluation set from DROID, AgiBotWorld, and OpenGalaxy. We assessed scene-entity detection precision and recall, bounding-box and gripper localization accuracy, and compliance with the JSON format. We selected doubao-seed-2-0-lite (ByteDance Seed, 2026) based on annotation quality and inference cost. In this comparison, the least expensive models had lower spatial precision, while the most expensive models gave small accuracy gains at a higher total cost. Queries used an OpenAI-compatible endpoint with greedy decoding.

F.1.4 AFFORDANCE ANNOTATION

Gripper-state phase segmentation The pipeline uses commanded gripper width to classify each frame as open or closed and divide each episode into phases. An open gripper defines an *object* phase, in which the arm approaches an entity to grasp. A closed gripper defines a *location* phase, in which the arm moves a grasped entity toward a placement region. The pipeline accepts a phase only after it persists for at least five frames and samples representative frames from longer phases. It segments the left and right grippers independently, so the two arms can have different phase labels at the same time.

Scene-entity detection For each episode, arm, phase, and camera combination, the pipeline queries the VLM on one representative frame. The query can include the episode's final frame as a reference. A worker pool runs the queries in parallel. The model returns each detected entity's name, category, bounding box in normalized 0–1000 image coordinates, and a short explanation in JSON format. The prompt applies five rules to category selection, spatial grounding, and arm assignment:

- **Excluded entities.** The prompt excludes the robot arm, gripper, and held items.
- **Detections must match the phase.** The prompt requests entities to be grasped during object phases and placement regions during location phases.
- **Containers must be task targets.** A cup is an object when the task is to pick it up. When the task is to pick a pen from the cup, the cup receives no object label.
- **The final state is used only to resolve ambiguity.** The final frame identifies the placement region and entities whose positions have not changed. The prompt skips these unchanged entities and requests boxes on the current frame.
- **Arm assignment.** Detections are assigned to the interacting gripper.

Dense forward-fill The pipeline copies the detections from each representative frame to all frames in the same phase. Each output record identifies an episode, frame, and camera, with bounding-box coordinates rounded to one decimal place. This procedure fills the phase without additional model queries; the boxes within a phase all come from the same source frame.

F.1.5 2D TRAJECTORY ANNOTATION

Real-robot data: VLM gripper localization For real-robot datasets without image-space trajectories, we use a VLM to locate the gripper. We sample every thirty frames from each camera stream and include the final frame. The pipeline discards the first few sampled frames because the gripper may be absent, which can lead to false detections. Each query includes closed- and open-gripper reference images and the target frame. The JSON output contains a visibility indicator and the point where the two fingers meet, in normalized 0–1000 coordinates. A dual-arm query returns one point per arm. The pipeline skips black frames from inactive cameras. The prompt marks a gripper visible only if both fingers are visible.

Gripper-point post-processing The pipeline removes invisible gripper points and rejects any point whose distance from the last accepted point exceeds the speed threshold. Agglomerative clustering groups nearby points and replaces each group with a representative point. Linear interpolation fills intervals between accepted points. Frames without valid observations receive a sentinel value.

Simulation data: camera projection For simulation datasets with end-effector states and camera calibration, the pipeline projects the 3D gripper position into the image. The first three state dimensions record the grip site, defined as the midpoint between the two fingers. We obtain camera poses from the simulator and compute intrinsics from the field of view. For LIBERO and LIBERO-Plus, the 256×256 images have a 45° vertical field of view, giving $f = 128/\tan(22.5^\circ) \approx 309.0$ and $c_x = c_y = 128$.

Let $\mathbf{p}_w \in \mathbb{R}^3$ denote the gripper position in world coordinates. It is first transformed into the camera frame and then projected through the pinhole model,

$$\mathbf{p}_c = \mathbf{T}_{wc}\,\tilde{\mathbf{p}}_w, \quad u = f\,x_c/z_c + c_x, \quad v = f\,y_c/z_c + c_y, \qquad \mathbf{T}_{wc} = \mathrm{diag}(1,-1,-1)\,[\mathbf{R} \mid \mathbf{t}], \tag{17}$$

Here $\tilde{\mathbf{p}}_w = [\mathbf{p}_w^\top, 1]^\top \in \mathbb{R}^4$ is the homogeneous world point, and $\mathbf{p}_c = (x_c, y_c, z_c)^\top \in \mathbb{R}^3$ gives its camera coordinates. The rotation $\mathbf{R} \in \mathbb{R}^{3\times3}$ and translation $\mathbf{t} \in \mathbb{R}^3$ form the 3×4 extrinsic matrix $[\mathbf{R} \mid \mathbf{t}]$. Thus $\mathbf{T}_{wc} \in \mathbb{R}^{3\times4}$ maps homogeneous world points to camera coordinates. The diagonal factor converts the simulator's camera axes to the OpenCV convention. The projection gives pixel coordinates (u, v) using focal length f and principal point (c_x, c_y). A valid projection requires $z_c > 0$: the gripper must lie in front of the camera.

Dataset images mirror the rendered view horizontally. We correct the projected coordinates as

$$\tilde{u} = (W-1) - u, \qquad \tilde{v} = v, \tag{18}$$

Here W is the image width in pixels and $(\tilde{u}, \tilde{v})$ is the corrected pixel coordinate. Before label generation, we checked the world-to-camera transform against the simulator's extrinsic matrix. We also overlaid projected points on dataset frames to check their alignment with the finger midpoint and determine the flip direction. The projection took a few minutes and required no model queries.

F.1.6 SUBTASK ANNOTATION

Subtask labels divide an episode into contiguous segments. Each segment describes the robot's current operation and supplies a task boundary for training.

Boundary detection via gripper state and motion cues Phase boundaries are identified by combining the gripper-state analysis from Section F.1.4 with action continuity heuristics. Candidate boundaries arise from three sources:

- **Gripper transitions**: Open/closed changes mark candidate phase boundaries.
- **Motion direction**: Changes in end-effector direction mark additional boundaries.
- **Object state changes**: Events such as an object being picked up, placed, or pushed to a target location are inferred from the interaction with the affordance annotations.

The pipeline merges phases shorter than the default 15 frames (approximately 0.5 seconds) into neighboring phases. It combines consecutive phases of the same operation type, such as repeated grasp adjustments. It also merges open-gripper phases without interaction into the next phase. At episode boundaries, it removes trailing reset phases in which a closed gripper moves away from the workspace. It also removes initial idle phases with an open gripper lasting more than 30 frames.

Subtask naming convention Each subtask name specifies an action, object, spatial relation, and target using the format {action} {object} {relation} {target}. Examples include "pick the red cube from the table", "place the biscuit onto the right plate", and "push the green block towards the corner". The fixed field order supports consistent naming and parsing.

F.1.7 MEMORY ANNOTATION

Memory annotation combines event-based segments and fixed physical-time windows. Each label summarizes the observed operations in its segment or window. The event-based procedure is described below; at inference, summaries are generated for completed fixed time windows.

Fixed time windows. We choose a physical-time window length for each task before annotation and apply the same grid throughout its demonstrations. For each completed window, the annotator summarizes observed events from its video and the earlier accepted memory. Subtask boundaries are independent of these windows, so one operation may span several windows. The summary is appended to history only after the window ends. The current window's target summary is excluded from its input history. Inference uses this fixed-window procedure to update memory as observations arrive.

Event-based annotation. The following preprocessing and phase-generation steps produce the event-based portion of memory supervision.

Data preprocessing and quality filtering Before annotation, the pipeline checks corruption, blur, inactivity, and duration as follows:

- **Black frame detection:** Videos with pixel variance below 10 are rejected.
- **Blur detection:** A Laplacian variance below 30 marks blur and can trigger rejection.
- **Stagnation detection:** Periods where the structural similarity index (SSIM) between consecutive frames exceeds 0.98 for 30 consecutive frames are removed. If these intervals exceed 50% of the video duration, the video is rejected.
- **Short duration:** Videos shorter than 2 seconds are rejected.
- **Long duration:** Videos longer than 60 seconds are split for separate annotation.

For retained videos, image similarity identifies static periods and candidate phase boundaries. It samples keyframes at a stride of $\text{FPS}/2$, where FPS is the frame rate, giving two samples per second. An SSIM below 0.6 between consecutive keyframes marks a candidate boundary. It selects the maximum-change frame within a one-second window around the candidate boundary.

Vision-language model annotation A VLM segments each processed video into phases and writes a description of each phase. The prompt specifies the task, action vocabulary, timing constraints, and output format:

- **Task description**: The prompt includes the ground-truth task description and requests its substeps, including preparatory and corrective actions shown in the video.
- **Action vocabulary**: Phase names must use an action from the list: *reach*, *grasp*, *lift*, *move*, *place*, *release*, *push*, *pull*, *rotate*, *slide*, *adjust*, *approach*, and *retract*.
- **Timing**: Phase timestamps must lie within the video duration. The first phase starts at 0:00, and the last ends with the video. The prompt limits each phase to 5 seconds and requests consecutive phases for longer actions. Each phase must end when the next begins.
- **Output format**: The JSON object contains a `phases` array and `summary` string. Each phase records `phase_number`, `phase_name`, `start_time`, `end_time`, and `description`. Times use `"MM:SS"` format.

Parallel annotation and post-processing The pipeline runs annotation queries across worker threads. It divides videos longer than 60 seconds into clips shorter than 30 seconds using visual similarity or uniform splitting. Each clip is annotated separately, and the phase labels are then merged in time order.

Post-processing sets each phase end time to the next phase start time and flags durations above 5 seconds. It repairs common JSON errors, including trailing commas and unescaped quotes. If the model produces too many phases, for example more than 20, it merges adjacent short phases. A final check warns when no phase description mentions a key object from the task description.

F.1.8 ENGINEERING DETAILS

Video pre-decoding All annotation stages need random access to video frames. We decode each dataset once into shards of raw byte arrays, each covering a contiguous range of episodes. An index maps each episode to its shard, and the loader keeps recently used shards in memory. Later annotation stages retrieve frames directly from the cache, so each video is decoded once.

Multi-arm support Dual-arm outputs use arm-prefixed column names. Post-processing groups records by arm and camera, and the training files retain separate columns for the two arms.

Output schema and merging The pipeline merges each dataset's annotations into a CoT mapping file aligned with its LeRobot (Cadene et al., 2026) data. Each row identifies an episode and frame. It contains the affordance type, per-camera bounding boxes, and 2D end-effector position, together with subtask and memory labels from their respective annotation stages. These mapping files supply CoT training targets.

F.2 PERCEIVER TRAINING

Each annotated example becomes one of the four question–answer tasks in Tab. 2. The shared model is trained with SFT on the structured answer; prompt tokens are excluded from supervision. Geometric coordinates are normalized to $[0, 1000]$. The PERCEIVER predicts CoT separately from the WORKER's action training.

The general version used approximately 1.067 million prebuilt QA examples from DROID and AgiBot World, with all four task weights equal to one. Full-parameter AdamW (Loshchilov & Hutter, 2019) training used a global batch of 512 QA examples, 2,084 updates, weight decay 0.01, and a peak learning rate of 10^{-5}, followed after warmup by cosine decay to 10^{-6}. LoRA (Hu et al., 2022), action supervision, and the FAST (Pertsch et al., 2025) branch were disabled. Visual augmentation included lighting, color, and image degradation. Memory deletion and cross-episode replacement were disabled in this prebuilt version.

Memory supervision combines event segments and fixed physical-time windows across training stages. Inference uses fixed physical-time windows with predetermined treatment of the final window. Downstream subtask queries use about two seconds of recent video. Some downstream data packages resample query times within predefined segments or time bins. Geometric context can use an earlier subtask while retaining the current target. The processor uses the selected video frames directly.

F.3 PREDICTED COT GENERATION

After benchmark adaptation, we freeze the PERCEIVER and generate CoT over the training demonstrations. Generation follows the temporal dependencies: completed windows update predicted memory, recent observations and that memory produce subtasks, and the predicted subtasks condition affordance and trajectory generation. Each query uses only observations and predicted history available by its source time.

Predictions are stored alongside the original annotations and indexed by episode and source frame. Records retain predicted content, physical source time, and parsing status; explicit abstentions remain missing fields. The original observations and action targets are preserved. These predictions supply the PRED source during downstream WORKER training.

F.4 WORKER TRAINING

The WORKER uses a validity mask $W_{h,d} \in \{0,1\}$ for action step h and coordinate d. For chunks with $\sum_{h,d} W_{h,d} > 0$, it minimizes

$$\mathcal{L}_{\text{work}}(\theta) = \mathbb{E}\left[\frac{\sum_{h,d} W_{h,d}\left(v_\theta(A_\tau, \tau, X, U)_{h,d} - V_{h,d}\right)^2}{\sum_{h,d} W_{h,d}}\right]. \tag{19}$$

For a fixed mask, this is the squared loss on valid coordinates scaled by the constant inverse valid-coordinate count. With varying masks, flatten (h, d) to coordinate index k, and let the regression input T be S or (S, U). The optimum in domain D is the weighted conditional mean $m_{D,k}(T) = \mathbb{E}_D[w_k V_k \mid T]/\mathbb{E}_D[w_k \mid T]$ where $w_k = W_k / \sum_{k'} W_{k'}$ and $\mathbb{E}_D[w_k \mid T] > 0$. Transfer requires a bounded density ratio for the joint input–mask law (S, U, W).

Current observations and action targets remain aligned while each CoT field independently samples a source and historical time. Subtask enters as text; geometry is rendered on a copy of the front-camera image. The original views remain available.

Full-model AdamW training used 30,000 updates at learning rate 5×10^{-5}. Maximum chunks were 50 steps, with a target duration of about 1.67 seconds. Source-specific normalization and validity masks handled mixed single- and dual-arm data. Flow time was drawn from $\mathrm{Beta}(1.5, 1.0)$ with a small adjustment away from zero. Inference used 10 Euler steps. This version also applied an auxiliary weight of 0.1 to internal structural padding supervision, alongside the valid-action objective in Eq. (19). Downstream versions use their own padding-supervision settings.

Outside cold start and subject to source availability, the general condition marginals were 20% GT, 55% REWRITE/geometric noise, 10% ERR, and 15% MISSING. Groups were sampled independently, with text rewrites constrained by available rewrite coverage. The historical sampling range was about 0.1–2.0 seconds, and CoT remained empty for the first second. General training excluded PRED. Downstream training adds causal predictions from the adapted PERCEIVER and retains empty groups when prediction fails.

Starting from general-training weights, the downstream WORKER trains on GT, PRED, perturbed, and missing CoT from its first update, with visual augmentation disabled.

F.5 GUIDANCE HYPERPARAMETERS

Tab. 9 lists RoboTwin 2.0 settings. Sources are sampled independently per field before random emptying. Lookback is sampled in physical time. Annotated and perturbed fields have the sampled age; PRED keeps the age of the selected source record, which can be older than the lookup time.

Table 9: RoboTwin 2.0 CoT training settings. Source weights precede independent field dropout.

Setting	Value
PERCEIVER context dropout	0.3
WORKER subtask sources	GT/PRED/ERR: 0.125/0.75/0.125
WORKER geometry sources	GT/NOISE/PRED/ERR: 0.08/0.24/0.60/0.08
WORKER field dropout	0.2 per field
Initial interval without CoT	1 second
WORKER temporal lookback	2–11 rows, or 0.12–0.66 s at $50/3$ Hz
Coordinate noise	Gaussian, standard deviation 5% of the short image side
Subtask error	Subtask from another time in the same episode
Affordance error	Box displacement of 30% of the image size
Trajectory error	Displacement of 20–40% of the short side and rotation of 15–45°

LIBERO guidance. The subtask source weights for GT, REWRITE, PRED, and ERR are 0.25, 0.35, 0.30, and 0.10. Geometry uses GT, NOISE, PRED, and ERR with weights 0.25, 0.25, 0.40, and 0.10. Independent field dropout is 0.15, and all fields are empty during the first second. The lookback is uniform on 0.1–1.0 s. Coordinate noise has standard deviation 3% of the short image side; the geometric error magnitudes follow Tab. 9.

Downstream optimization. The archived LIBERO and RoboTwin WORKER runs start from the general 30,000-update checkpoint. Both use AdamW with global batch size 256, seed 42, and eight A100 GPUs. LIBERO uses 30,000 updates, a peak learning rate of 10^{-5}, 500 warmup updates, and cosine decay to 10^{-6}. Its joint training view contains 1,689 demonstrations; LIBERO-Long receives a sampling multiplier of 1.30 relative to row-proportional sampling, and the other suites use 1.0. RoboTwin uses 5,000 updates, a peak learning rate of 5×10^{-5}, 250 warmup updates, and cosine decay to 2.5×10^{-7}. The action chunks contain 10 steps for LIBERO and 50 for RoboTwin. Both runs attend to the full action chunk during training and retain loss masks for padded action targets. All downstream visual augmentation is disabled.

F.6 ASYNCHRONOUS INFERENCE

A shared vLLM service hosts the PERCEIVER. Independently scheduled request loops update memory, subtask, affordance, and trajectory. Memory summaries accumulate in fixed-window order. For the other three fields, parsed content and its source frame and time are published together; a response from an older source cannot replace a newer cached result.

At each action query, the WORKER reads the latest available fields with the current observation and computes their ages from the source times. CoT generation continues while actions are predicted. Return time determines availability, while source time determines age; an explicit abstention clears the corresponding field.

F.7 PROMPT TEMPLATES

The PERCEIVER uses the shared system prefix below, followed by a task-specific system message and user input. Braced fields are filled with observations and available CoT; media placeholders denote attached images or video. Role headings organize the presentation. The geometric templates show the bimanual form, with single-arm adaptations noted below.

Prompt 1: Shared system prefix: relative time and image coordinates

You are a visual reasoning module for a manipulation robot.

Notation:
– ‘<now>‘ marks an image or item from the current query time.
– A tag like ‘<X.X seconds ago>‘ gives only the delay from the current query; it never reveals time since episode start.
– Memory entries are ordered from older to newer. Their order is meaningful even when exact timing is omitted.
– Video timestamps are local to the supplied clip and start at 0; they are not episode timestamps.
– Past CoT context may be slightly stale; use its relative delay and the current visual evidence when deciding how much to trust it.
– All bbox / point / trajectory coordinates are integers in [0, 1000] for both x and y, expressed in a normalized image space.

F.7.1 SUBTASK

The recent video ends at the query time; memory contains summaries of earlier completed segments. An empty or dropped memory block is rendered as `N/A`.

Prompt 2: Subtask: the manipulation goal active at the query time

SYSTEM CONTINUATION
Task: infer the robot's CURRENT manipulation subtask at the present timestamp.

You are given:
– the task instruction,
– an ordered observation history from oldest to newest; it contains causally sampled real frames and the last image is NOW,
– summaries of earlier observations (memory; may be empty).

A subtask identifies the direct manipulation goal active NOW: an operation and its target object or destination. Approach, grasp, transport, and release may belong to the same direct operation goal.

Rules:
– Use changes across the ordered images when useful, but label the subtask at NOW.
– Use recent observations and earlier memory to identify the active goal.
– Do NOT predict a distant future subtask or describe the entire task.
– Do not create a new subtask merely because a motion phase changes or a memory window ends.
– Name the concrete operation and its target object or destination.
– Describe the goal without assuming successful completion.
– Keep the answer concise and do not narrate unrelated goals.

Output exactly one answer block in this format:
<answer>Subtask: <action + target object/location></answer>

If the given inputs do not provide enough information to identify any active manipulation step, output exactly:
<answer>Subtask: no_subtask</answer>
– The ordered history is a native video whose timeline starts at clip–local 0 seconds and whose final frame is NOW.

USER
Task Instruction
{instruction}

Observation Video
Observation History ({camera}, oldest to NOW) video (clip–local 0.0 to {clip_duration_s} seconds; final frame is NOW):
[VIDEO: recent observation history]

Memory
Current Memory:
[memory_1] {earlier_summary_1}
...
[memory_k] {earlier_summary_k}

F.7.2 MEMORY

A memory query summarizes one completed window and receives only earlier summaries. The `Previous Memory Segments` block excludes the window being summarized and becomes `N/A` when no history is available. Video timestamps are relative to the supplied clip.

Prompt 3: Memory: an incremental summary of the completed window

SYSTEM CONTINUATION

Task: summarize the visible task–relevant activity in the given video window into one memory entry.

You are given:
1. the task instruction,
2. a recent video clip covering the last few seconds of the episode,
3. any older memory from earlier in the task (may be empty).

Your job is to write a concise memory entry describing what happens in the given video clip.

The memory should capture:
– the main manipulation subtasks performed in the clip,
– the manipulated object(s) or target location(s),
– important progress updates,
– any clear changes in object state or scene state relevant to the task.

Guidelines:
– Describe only activity or relevant states visible inside the given video window.
– An operation may continue across window boundaries; a window ending does not imply completion.
– Do not assert object identity, grasp, release, or placement without visual support.
– If there is no clear new progress, describe the visible ongoing activity or relevant stable state.
– Use older memory for continuity, not as evidence of new progress in this window.
– Do not predict the next step.
– Do not give a frame–by–frame description.
– Keep the memory short, concrete, and useful for later reasoning about task progress.
– Output only this window's incremental summary, not the accumulated history or the current subtask.

Output exactly one answer block:
<answer>Recent memory: <compact task–relevant event summary></answer>

If the given video clip does not provide enough information to summarize any task–relevant event, output exactly:
<answer>Recent memory: no_memory</answer>

USER

Task Instruction
{instruction}

Past Video
Recent Video (clip–local 0 to {clip_duration_s} seconds):
[VIDEO: completed memory window]

Memory
Previous Memory Segments:
[memory_1] {earlier_summary_1}
...
[memory_k] {earlier_summary_k}

F.7.3 AFFORDANCE

Geometry uses the current image and the available subtask. Its time tag gives the subtask's age relative to the query; a zero-age subtask uses <now>. Missing or dropped subtask context becomes N/A. The user input below is shared by affordance and trajectory queries.

Prompt 4: Affordance: object or target-region boxes for each arm

SYSTEM CONTINUATION
Task: identify the manipulation target relevant to EACH robot arm's CURRENT action.

You are given:
– the task instruction,
– the current image,
– the current subtask (may be empty).

Output requirements (per arm, left then right):
– For each arm, identify the single most relevant target object or target region for the current subtask.
– Predict one or more 2D bounding boxes for that target.
– If an arm has no valid manipulation target, output no_affordance for that arm.

Affordance type vocabulary (choose the one that best matches the predicted target):
– 'object': the affordance is on a specific object that the end–effector should act on (e.g. a cup to grasp, a brush to pick up).
– 'location': the affordance is on a spatial region rather than a specific object (e.g. a placement area, a target hook, a tray to place into).
– 'contact': the affordance is a contact patch or contact path on a surface where the end–effector should make sustained or sweeping physical contact (e.g. a surface to wipe, a path to push along).
– 'no_affordance': no valid manipulation target can be determined for that arm.

Rules:
– Focus only on the CURRENT subtask / current image.
– Do not predict future targets.
– The bbox should tightly cover the manipulation–relevant target.
– All coordinates must be integers in [0, 1000].

Output exactly one answer block containing a JSON object with 'left' and 'right'.
Each arm value is either an affordance object with the same schema as the single–arm task, or the string 'no_affordance':
<answer>{"left": {"affordance_type": "object", "boxes": [[x_min, y_min, x_max, y_max]]}, "right": "no_affordance"}</answer>

If neither arm has a localizable manipulation target, output exactly:
<answer>no_affordance</answer>

USER
Task Instruction
{instruction}

Current Image
Current Obs Image ({camera}): <now> [IMAGE: current camera view]

Subtask
<{subtask_age_s} seconds ago> {subtask}

For a single arm, the query returns one affordance_type/boxes object for the current manipulation target inside <answer>. The left/right wrapper is omitted. The query returns <answer>no_affordance</answer> when it cannot localize a target.

F.7.4 TRAJECTORY

The trajectory query uses the same user input as Prompt 4. The horizon and first future-point offset are filled from the dataset configuration; the RoboTwin recipe uses 4 s, 0.4 s, and five points.

Prompt 5: Trajectory: five future image-space points for each arm

SYSTEM CONTINUATION
Task: predict BOTH robot arms' FUTURE MOTION TRAJECTORIES over the next ˜{horizon_s} seconds of robot motion, expressed in image space.

You are given:
– the task instruction,
– the current image,
– the current subtask (may be empty).

Trajectory definition (applies to each arm independently):
– A trajectory is a sequence of 5 image–space points sampled along that arm's FUTURE motion path.
– Do NOT include the end–effector position at the CURRENT timestamp; the trajectory starts strictly in the future.
– [x1, y1] is the first future position, roughly +{future_start_s} seconds after CURRENT; [x5, y5] ends near +{horizon_s} seconds after CURRENT.
– The 5 points are approximately arc–length–uniform along the actual future path; they are NOT copies of the current position and NOT forced onto a straight line.
– Coordinates are integers in [0, 1000].

Each arm has exactly three possible states:
– A 5–point trajectory: the arm moves and stays visible.
– static: the arm is visible but barely moves over the window (its future path is shorter than a small threshold).
– no_2dtraj: the arm's end–effector is not visible / cannot be localized.

Output exactly one answer block containing a JSON object with 'left' and 'right'.
Each arm is a JSON trace, 'static', or 'no_2dtraj':
<answer>{"left": [[x1, y1], [x2, y2], [x3, y3], [x4, y4], [x5, y5]], "right": "static"}</answer>

If both arms are 'no_2dtraj', collapse the whole answer to:
<answer>no_2dtraj</answer>

For a single arm, the answer contains one point list, `static`, or `no_2dtraj`. These distinguish motion, a stationary visible arm, and an unlocalizable end effector.

F.7.5 ACTION GENERATION

The WORKER receives this text with the current camera views and a front-view copy carrying the geometric overlay. The state field contains space-separated quantized robot-state values; the action expert predicts a continuous action chunk.

Prompt 6: Worker: text prefix accompanying the current images and CoT overlay

Task: {instruction}. Subtask: {subtask} <age {subtask_age_s}s> <overlay age: affordance {affordance_age_s}s, trajectory {trajectory_age_s}s>, State: {discretized_state};
Action:

Each field's age is measured from its source observation to the current action query. Unavailable CoT text, age tags, and overlay drawings are omitted. With all fields missing, the text contains only the instruction, state, and `Action:` prefix. Memory reaches this policy through the PERCEIVER's generated guidance. The same text builder is used in training and inference.