GRVM: GOAL-RELATIVE VALUE MODELING BEYOND TEMPORAL PROGRESS FOR ROBOT MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Robot value models increasingly provide feedback for execution evaluation and policy improvement. However, supervision based on task advancement does not necessarily reflect whether the current physical state has become more favorable for task completion. This distinction is particularly important around failures and recoveries, where temporal progress and physical state quality can diverge. We introduce Goal-Relative Value Model (GRVM), which separately predicts task progress, goal-relative state quality, and trajectory outcome from visual observations and language. A human-audited annotation pipeline constructs dense goal-relative supervision using end-effector poses at successful stage completion as local goals and extends this supervision to failed and recovered executions through failure-aware relabeling. We pretrain GRVM on 286,316 annotated trajectories (2,612 hours), including 17,286 failure/recovery trajectories. We also construct a balanced, multi-embodiment benchmark of 1,000 held-out trajectories to evaluate continuous-label agreement and local state ordering around failures and recoveries. GRVM improves triplet accuracy by 58.0% and recovery-direction accuracy by 39.7% relative to the strongest competing baseline for each metric. With GRVM kept frozen, its feedback increases mean success across four LIBERO suites from 89.8% to 94.3% over sparse-reward RL while accelerating learning. GRVM-guided online RL raises mean success from 55.0% (SFT) to 95.0% across four real-robot tasks.

1 INTRODUCTION

Reinforcement learning enables robots to improve their behavior through interaction, with reward feedback guiding policy optimization (Kober et al., 2013). Providing informative feedback has motivated reward shaping (Ng et al., 1999) and reward learning from demonstrations (Abbeel & Ng, 2004), human preferences (Christiano et al., 2017; Lee et al., 2021), and reward sketches (Cabi et al., 2019). For robot policy improvement, this feedback should capture not only advancement through a task, but also whether the current state supports successful continuation. This distinction is especially important in manipulation, where grasp configuration and contact conditions affect the feasibility and success of subsequent actions (Fang et al., 2020; Liang et al., 2022). Two executions at the same nominal stage may therefore differ in gripper–object alignment or contact quality, with different implications for eventual task success. Effective value estimation therefore requires distinguishing **task advancement** from current **state quality**.

Recent robot reward and value models obtain scalable supervision from trajectory timing (Zhang et al., 2025; Huang et al., 2026) or trajectory-level comparisons (Liang et al., 2026). Timing-based targets use normalized frame indices or remaining time to completion as proxies for task advancement. Such targets can assign similar labels to physically different states and do not explicitly distinguish local state improvement from temporal advancement. Trajectory-level preferences provide complementary supervision across successful and failed executions, but these labels alone do not specify where an execution deteriorates or recovers. These limitations motivate a complementary form of dense supervision: the quality of the current physical state relative to the requirements of the ongoing manipulation.

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

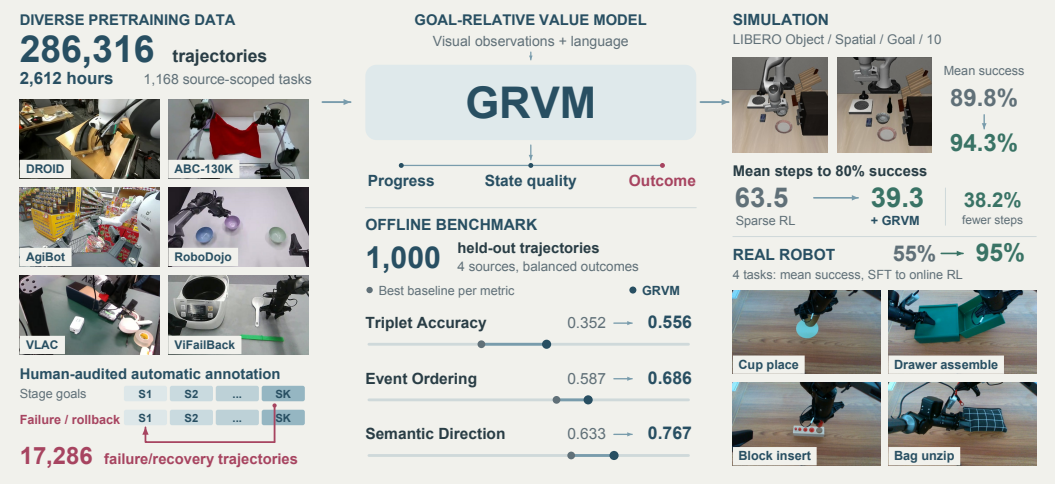


Figure 1: **GRVM overview.** Human-audited stage goals and failure/rollback annotations support pretraining on diverse robot trajectories (left). GRVM predicts progress, goal-relative state quality, and outcome, with evaluation on a held-out benchmark (center). Frozen GRVM feedback guides closed-loop policy learning in simulation and on real robots (right).

Our key observation is that end-effector poses at successful stage completion provide local goals for supervising physical state quality. We derive dense state-quality targets from gripper-to-goal relative poses. A human-audited GPT-based pipeline identifies stage boundaries, failure points, and rollback stages using task-specific prompts and human-provided examples. These annotations support target construction and failure-aware relabeling of failed and recovered executions.

Building on these annotations, we introduce **GRVM**, a value model built on Qwen3-VL that separately predicts frame-wise task progress and goal-relative state quality, together with a trajectory-level outcome. We pretrain GRVM on 286,316 annotated trajectories (2,612 hours) spanning 1,168 source-scoped tasks, including 17,286 failure/recovery trajectories that provide direct supervision on execution degradation and recovery. Robot poses and annotations are used only to construct training targets; at inference, GRVM predicts the three quantities from multi-view visual observations and language alone.

We construct a multi-embodiment benchmark of 1,000 held-out trajectories from four data sources, balanced between successful and non-successful executions, to assess continuous-label agreement and local state ordering around failures and recoveries. GRVM outperforms the strongest competing baseline for each metric by 58.0% in triplet accuracy and 39.7% in recovery-direction accuracy. With GRVM kept frozen, its feedback raises mean success across four LIBERO suites Liu et al. (2023) from 89.8% to 94.3% over sparse-reward RL and enables policies to reach high success rates earlier during training. On four real-robot tasks, GRVM-guided online RL raises mean success from 55.0% for the initial SFT policy to 95.0%.

We summarize our main contributions as follows:

- **Goal-relative value modeling.** We introduce and pretrain GRVM, which separately predicts task progress, goal-relative state quality, and trajectory outcome, distinguishing task advancement from physical state improvement.
- **Automated supervision construction.** We develop a human-audited automatic annotation pipeline that identifies manipulation stages, failures, and recoveries, and constructs dense supervision using stage-local end-effector goals and failure-aware relabeling.
- **Benchmark and policy validation.** We establish a balanced, multi-embodiment benchmark of 1,000 held-out trajectories for evaluation. Offline comparisons demonstrate more accurate local state ordering and recovery-direction prediction. With GRVM kept frozen, closed-loop learning across four LIBERO suites and four real-robot tasks achieves faster learning in simulation and higher task success rates in both settings.

2 Related Work

Robot Reward and Value Learning. Robot reward and value estimation has been studied through goal-conditioned representations in VIP and LIV (Ma et al., 2022; 2023), VLM-derived visual preferences in RL-VLM-F (Wang et al., 2024), in-context progress estimation in GVL (Ma et al., 2025), and temporal-distance prediction in RynnValue (Huang et al., 2026). Recent methods extend supervision beyond successful demonstrations. ReWiND (Zhang et al., 2025) synthesizes regressions through video rewinding, RoboReward (Lee et al., 2026) constructs negative and near-miss examples, and RoboMeter (Liang et al., 2026) combines frame-level progress with trajectory-level preferences. Other methods introduce more structured process supervision: VLAC (Zhai et al., 2025) predicts pairwise relative task progress, Robo-Dopamine (Tan et al., 2025) uses step-aware supervision and multi-view observations, and ARM (Mao et al., 2026) labels transitions as progressive, regressive, or stagnant. Robo-Dopamine 2.0 (Xu et al., 2026) further incorporates execution history and explicitly models failure and recovery. GRVM complements these approaches with dense state-quality targets derived from stage-local end-effector goals and failure-aware relabeling. By supervising task progress and goal-relative state quality separately, alongside task outcome, GRVM explicitly represents local physical alignment as a distinct dimension of value estimation.

Evaluation of Robot Value Models. Existing evaluations examine reward accuracy, ranking, and robustness. RoboRewardBench (Lee et al., 2026) measures agreement with human-verified reward labels, while RBM-EVAL (Liang et al., 2026) evaluates reward alignment, trajectory ranking, and instruction–trajectory matching. Progress-Bench (Zhang et al., 2026) tests progress reasoning across demonstration modalities and viewpoints, while ROBORMBENCH (Jeung et al., 2026) examines reward consistency under semantically equivalent instruction paraphrases. Robo-Dopamine 2.0 (Xu et al., 2026) evaluates trajectory-level ordering consistency under temporal ambiguity and out-of-distribution execution. We complement these protocols with a balanced, multi-embodiment benchmark that evaluates continuous-label agreement and state ordering, emphasizing the correctness of local value changes within trajectories around annotated failures and recoveries. Downstream control provides a further test of reward utility, as demonstrated by VIP (Ma et al., 2022) and robot reinforcement learning with learned rewards (Zhang et al., 2025; Lee et al., 2026; Liang et al., 2026). We therefore also evaluate frozen GRVM feedback through closed-loop policy learning in simulation and on real robots.

3 Method

Given a multi-view observation sequence $\mathcal{O} = \{o_t\}_{t=0}^{T-1}$ and a task instruction ℓ, GRVM predicts task progress $\hat{p}_t$, goal-relative state quality $\hat{c}_t$, and a trajectory-level outcome $\hat{q}$:

$$F_\theta(\mathcal{O}, \ell) = \left(\{\hat{p}_t, \hat{c}_t\}_{t=0}^{T-1}, \hat{q}\right). \tag{1}$$

We first annotate task stages and failure–recovery events, then construct dense supervision from stage timing and goal-relative end-effector poses with failure-aware relabeling (Fig. 2). We build GRVM on the pretrained Qwen3-VL multimodal backbone (Bai et al., 2025) and train it to predict task progress, goal-relative state quality, and trajectory outcome from multi-view observations and task instructions (Fig. 3). Structural annotations and end-effector poses are used only to construct supervision and are not required at inference.

3.1 Reference-Guided Trajectory Annotation

Stage and failure annotation. As illustrated in Fig. 2, we use a GPT-based annotator guided by task-specific prompts and human-annotated examples. Given visual observations and the task instruction, it segments both successful and failed trajectories into semantic manipulation stages. For a trajectory with K stages, each stage j is represented as $\mathcal{E}_j = [b_j, u_j]$, where b_j and u_j denote its start and end frames. For stages containing a failure, the annotator additionally identifies the failure frame f_j and the rollback stage r_j, which provides the reference for post-failure state-quality supervision. The prompts require explanations of the failure cause and the rollback-stage choice.

Iterative human audit and refinement. We annotate and audit trajectories separately for each task category, such as pick-and-place. At each iteration, humans inspect 10 sampled trajectories, and

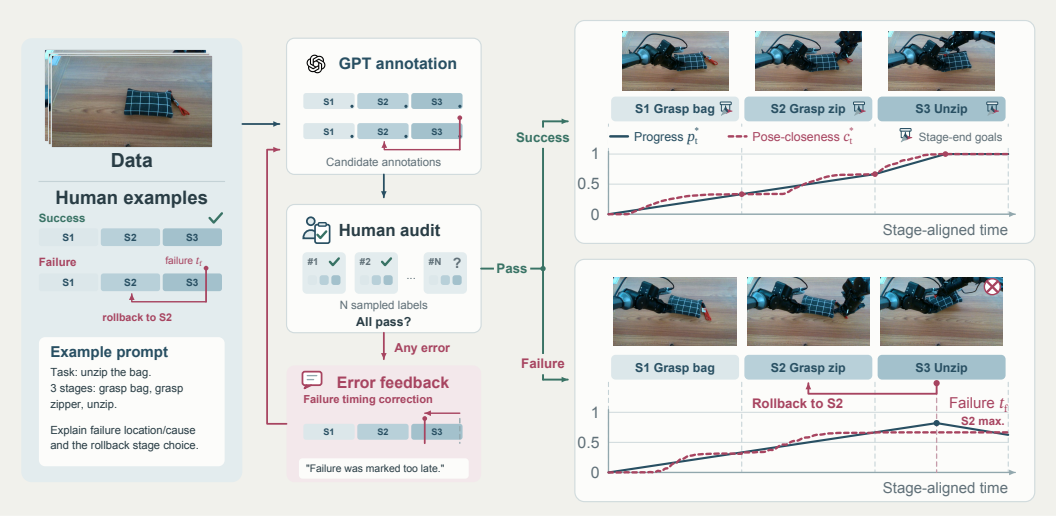


Figure 2: **Human-audited annotation pipeline.** We use task-specific prompts and human-annotated examples to guide GPT in identifying stage boundaries, failure points, and rollback stages. Humans audit sampled annotations and refine prompts with corrections and feedback. Accepted annotations and end-effector poses yield dense progress and pose-closeness targets.

the annotations for that task are accepted only if all 10 pass the audit. If any annotation fails, humans provide the corrected annotation together with an explanation of why the original was incorrect and how it was corrected. These corrections and explanations are incorporated into the task-specific prompt to guide GPT in the next annotation round. This annotation–audit–correction loop repeats until all 10 sampled trajectories pass. Additional annotation details are provided in Appendix A.

3.2 Goal-Relative Supervision

Using the stage annotations from Section 3.1, we construct two dense supervision signals: progress based on stage timing, and pose-closeness based on the end-effector pose relative to a local goal. We first define their nominal targets and then introduce failure-aware relabeling. We reserve nominal progress for the timing-based target p_t^{nom}, and use task progress for the supervised target $p_t^\star$ and its prediction $\hat{p}_t$, which incorporate failure-aware relabeling.

Nominal targets. We index the K stages by $j = 0, \ldots, K-1$. For a stage $\mathcal{E}_j = [b_j, u_j]$ without failure, its terminal end-effector pose defines the local goal $G_j = P_{u_j}$. We compute within-stage temporal progress and pose-closeness as

$$\alpha_t = \frac{t - b_j}{u_j - b_j}, \qquad r_t(G_j) = h\left(\text{clip}\left(1 - \frac{D(P_t, G_j)}{\max\big(D(P_{b_j}, G_j), \epsilon\big)}, 0, 1\right)\right), \tag{2}$$

where P_t is the recorded end-effector pose, D measures translational and rotational pose distance, and $\epsilon > 0$ is a numerical safeguard. Here, h is a sigmoid-based mapping that converts the normalized reduction in pose distance into within-stage pose-closeness. The nominal targets are

$$p_t^{\text{nom}} = \frac{j + \alpha_t}{K}, \qquad c_t^{\text{nom}} = \frac{j + r_t(G_j)}{K}. \tag{3}$$

Thus, progress increases with elapsed time within a stage, whereas pose-closeness reflects proximity to its physical goal. For stages without failure, we directly use $p_t^\star = p_t^{\text{nom}}$ and $c_t^\star = c_t^{\text{nom}}$.

Failure-aware targets. For a stage with annotated failure frame f_j, progress follows the nominal target through f_j, then decreases by $1/T$ per frame on the failed branch:

$$p_t^\star = \max\left(0, p_{f_j}^{\text{nom}} - \frac{t - f_j}{T}\right), \qquad t > f_j, \tag{4}$$

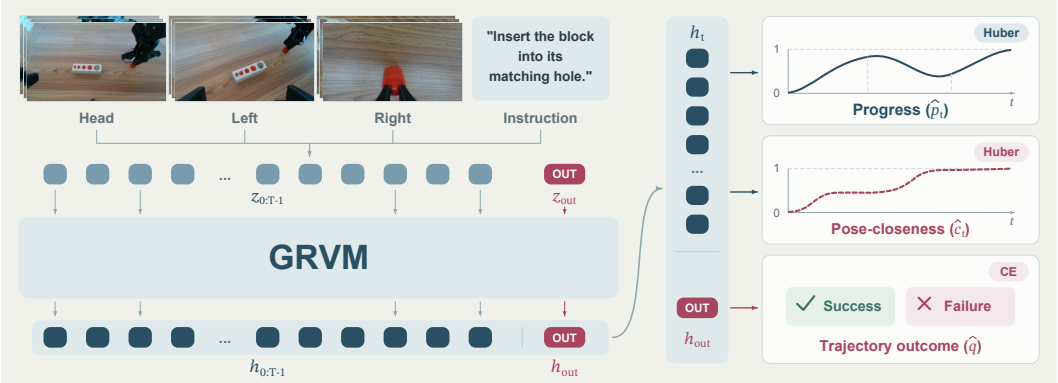


Figure 3: **GRVM architecture and training.** Multi-view visual features are fused with the instruction token sequence and processed by a causal trajectory transformer. Separate heads predict progress and pose-closeness from frame representations h_t, and trajectory outcome from the terminal-token representation h_{out}. Huber losses supervise the frame-wise predictions; cross-entropy supervises the outcome.

where T is the trajectory length.

For pose-closeness, the terminal pose of a failed execution is not treated as a valid goal. If a later recovery completes the interrupted stage, its terminal pose provides the goal for constructing pre-failure targets. Otherwise, pose-closeness is held at the preceding valid stage's maximum up to the failure frame. After f_j, the target is reset to the maximum pose-closeness of the annotated rollback stage r_j:

$$c_t^\star = \max_{s \in \mathcal{E}_{r_j}} c_s^\star, \qquad t > f_j. \tag{5}$$

This value is maintained until the execution returns to a valid task branch, after which subsequent stages follow the nominal construction. Additional details are provided in Appendix B.

3.3 GRVM Architecture and Training

Model architecture. As illustrated in Fig. 3, GRVM builds on pretrained Qwen3-VL-4B (Bai et al., 2025) visual features and language-conditioned trajectory modeling. Each frame is encoded independently; spatial pooling and masked view attention aggregate the available camera views into a state token. Cross-attention to the instruction token sequence yields one task-conditioned token z_t per time step. A separate causal trajectory transformer processes these tokens with a learnable terminal outcome token z_{out}:

$$(h_0, \ldots, h_{T-1}, h_{\text{out}}) = F_{\text{causal}}(z_0, \ldots, z_{T-1}, z_{\text{out}}). \tag{6}$$

Each h_t attends only to valid frame tokens up to time t, while h_{out} summarizes the valid trajectory through the terminal token. Separate MLP heads predict task progress $\hat{p}_t$ and pose-closeness $\hat{c}_t$ from h_t, and trajectory outcome $\hat{q}$ from h_{out}. Sigmoid activations constrain the progress and pose-closeness predictions to $[0, 1]$, while the outcome head produces two logits for success and failure. The heads retain distinct outputs and supervision; the composite value used for evaluation and policy learning is computed from the predictions, not from an additional scalar head.

Training objective. We supervise progress and pose-closeness with Huber losses (Huber, 1992) $\mathcal{L}_{\text{prog}}$ and $\mathcal{L}_{\text{qual}}$, using the dense targets constructed in Section 3.2. The trajectory outcome is supervised by a cross-entropy loss $\mathcal{L}_{\text{out}}$ against the success/failure label. The frame-level losses are averaged over valid supervised frames, excluding padding and frames without the corresponding targets. The overall objective is

$$\mathcal{L} = \lambda_p \mathcal{L}_{\text{prog}} + \lambda_c \mathcal{L}_{\text{qual}} + \lambda_o \mathcal{L}_{\text{out}}, \tag{7}$$

where $\lambda_p, \lambda_c, \lambda_o$ balance the three training objectives. Training hyperparameters are detailed in Appendix C.

Table 1: **Offline value evaluation on our 1,000-trajectory benchmark.** For GRVM, scalar agreement uses the progress prediction $\hat{p}_t$, while relative semantic consistency uses $\Phi_t = \hat{p}_t^{0.4}\hat{c}_t^{0.6}$. Arrows indicate the preferred direction. Best results are shown in bold.

Model	Scalar Agreement			Relative Semantic Consistency					
	VOC ↑	Pearson ↑	MAE ↓	SD-S ↑	SD-F ↑	SD-R ↑	TA ↑	AK ↑	EO ↑
VLAC-2B	0.719	0.354	0.293	0.483	0.168	0.287	0.066	0.715	0.193
VLAC-8B	0.688	0.402	0.268	0.564	0.279	0.381	0.124	0.707	0.293
RoboReward-8B	0.751	0.522	0.200	0.537	0.087	0.162	0.018	0.696	0.122
Robo-Dopamine-4B	0.540	0.299	0.316	0.278	0.052	0.109	0.025	0.607	0.059
Robo-Dopamine-8B	0.612	0.301	0.305	0.334	0.120	0.244	0.096	0.623	0.130
RoboMeter-4B	0.757	0.630	0.205	0.760	0.557	0.582	0.348	0.633	0.583
RynnValue-4B	0.419	0.337	0.243	0.668	0.632	0.465	0.352	0.358	0.587
RynnValue-8B	0.582	0.423	0.287	0.760	0.530	0.487	0.329	0.531	0.517
GRVM (ours)	**0.812**	**0.664**	**0.164**	**0.796**	**0.692**	**0.813**	**0.556**	**0.780**	**0.686**

4 EXPERIMENTS

We evaluate GRVM through offline value estimation and closed-loop policy improvement. On our offline benchmark, we evaluate how closely predicted scores agree with reference annotations and whether they correctly reflect the relative ordering of states during task progression, failure, and recovery. We then evaluate whether GRVM-derived rewards improve policy learning in simulation and on real robots. Finally, ablation studies examine the contributions of failure data and pose-closeness supervision.

4.1 OFFLINE VALUE EVALUATION

We evaluate trajectory-score reliability before using GRVM for policy optimization.

Benchmark and baselines. We construct a benchmark of 1,000 trajectories from AgiBot2026 (Team, 2026), DROID (Khazatsky et al., 2024), ViFailBack (Zeng et al., 2026), and VLAC-Cut (Zhai et al., 2026). Each source contributes 125 successful and 125 non-successful executions, with failure and recovery events included where available. The held-out subset is separated from GRVM training by leakage group, keeping related episodes, camera views, and trajectory slices in the same split. We compare GRVM with five publicly available baselines for robot reward and value estimation: VLAC (Zhai et al., 2025), RoboReward (Lee et al., 2026), Robo-Dopamine (Tan et al., 2025; Xu et al., 2026), RoboMeter (Liang et al., 2026), and RynnValue (Huang et al., 2026).

Evaluation protocol. All models use the same held-out manifest and up to 128 shared temporal endpoints per trajectory, with precomputed reference annotations. GRVM uses training-consistent multi-view preprocessing and a single causal forward pass over an event/phase-aware subsampled trajectory; baselines retain their model-specific input and inference procedures. Scores at sparse endpoints are causally held between queries when evaluating event-level comparisons. Recovery metrics require annotated recovery events, and Triplet Accuracy requires a complete pre-failure, failure, and recovery sequence. We aggregate comparisons within each trajectory and then average across eligible trajectories; Appendix D details sampling, score readouts, and baseline processing.

Evaluation metrics. We evaluate two aspects of trajectory scoring. *Scalar agreement* measures consistency with continuous reference annotations using Value-Order Correlation (VOC), Pearson correlation, and mean absolute error (MAE). For GRVM, these metrics use the progress prediction $\hat{p}_t$, since the references primarily describe nominal task advancement. *Relative semantic consistency* evaluates state ordering during task progression, failure, and recovery, using the composite score $\Phi_t = \hat{p}_t^{0.4}\hat{c}_t^{0.6}$ for GRVM. Semantic Direction measures whether scores change in the expected direction during advancement (SD-S), failure (SD-F), and recovery (SD-R). Triplet Accuracy (TA) evaluates the joint ordering of pre-failure, failed, and recovered states; Anchor Kendall (AK) mea-

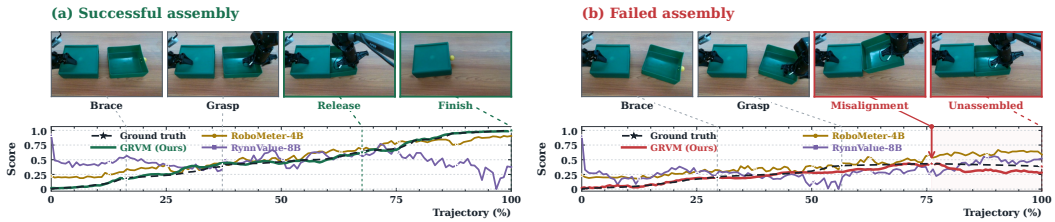


Figure 4: **Trajectory-level score comparison on real-robot drawer assembly.** We compare GRVM, RoboMeter-4B, and RynnValue-8B against the reference scores for (a) a successful execution and (b) a failed execution. GRVM scores use $\Phi_t = \hat{p}_t^{0.4}\hat{c}_t^{0.6}$, with the reference constructed using the same aggregation. The vertical marker in (b) indicates the annotated failure time.

sures ranking consistency among semantic anchors; and Event Ordering (EO) evaluates pairwise ordering around failure and recovery events. Detailed definitions are provided in Appendix D.

Results. GRVM improves both agreement with dense reference annotations and state ordering around failures and recoveries. As shown in Table 1, GRVM achieves the lowest MAE of 0.164 and the highest VOC and Pearson correlation.

More importantly, this overall agreement is accompanied by reliable responses to changes in execution quality: its SD-F and SD-R scores reach 0.692 and 0.813, respectively, outperforming all compared baselines. These advantages persist when evaluation requires comparing states across events rather than identifying individual score changes. GRVM achieves a Triplet Accuracy of 0.556 and an Event Ordering score of 0.686, compared with the strongest baseline scores of 0.352 and 0.587. Together, these results support the central objective of GRVM: tracking task advancement while distinguishing deterioration and recovery in the underlying execution state.

Qualitative trajectory analysis. Figure 4 illustrates how trajectory scores evolve during successful and failed drawer assembly. In the successful example, the GRVM curve closely follows the reference trend: scores increase through the initial bracing and grasping stages, change more slowly during the middle portion, and rise again toward completion. RoboMeter captures the overall upward trend but exhibits larger intermediate fluctuations, while RynnValue declines near the end despite successful completion. In the failed example, GRVM likewise assigns increasing scores to the valid early stages, but the curve gradually decreases after the annotated misalignment and finishes well below the successful trajectory. The score remains nonzero, reflecting partial advancement without indicating successful completion. By comparison, the baseline curves do not exhibit a sustained post-failure decline and remain approximately stable or increase over much of the remaining trajectory. Together, the two examples illustrate the distinction targeted by GRVM: recognizing valid task advancement while separating continued robot motion from effective progress toward completion.

4.2 Policy Improvement in Simulation

Experimental setup. We evaluate whether GRVM-derived rewards improve policy learning on four LIBERO suites: Object, Spatial, Goal, and LIBERO-10. Using the RLinf implementation of PPO for π_0 (Black et al., 2024), we compare a sparse task-success reward baseline with value-guided variants using RoboMeter and GRVM. All methods share the same policy initialization, PPO hyperparameters, interaction budget, and evaluation protocol, differing only in the reward signal used for policy optimization. After each rollout, GRVM and RoboMeter score the collected visual trajectory conditioned on the task instruction. For GRVM, we augment the environment reward with the temporal change in the composite score:

$$r_t = r_t^{\text{env}} + \Phi_t - \Phi_{t-1}, \tag{8}$$

where r_t^{env} is the sparse environment reward and $\Phi_t = \hat{p}_t^{0.4}\hat{c}_t^{0.6}$. For RoboMeter, we apply the same reward formulation in Eq. 8, replacing Φ_t with its predicted scalar score while retaining the sparse environment reward. GRVM remains frozen throughout policy optimization. We report task success rates over training to compare learning dynamics and final performance. The shared PPO configuration and suite-specific settings are detailed in Appendix F.

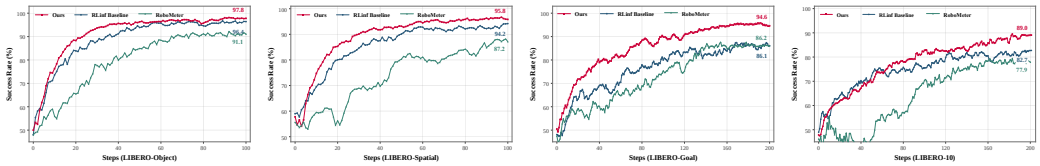


Figure 5: **Closed-loop policy improvement on four LIBERO suites.** Success rates during PPO training of π_0, comparing the RLinf sparse-reward baseline with value-guided variants using RoboMeter and GRVM.



Figure 6: **Real-robot evaluation tasks.** (a) Cup place, (b) Drawer assemble, (c) Block insert, and (d) Bag unzip.

Results. Figure 5 shows that GRVM improves policy learning across all four LIBERO suites, achieving a mean final success rate of 94.3%, compared with 89.8% for the sparse-reward baseline and 85.6% for RoboMeter. GRVM achieves the highest final success rate on every suite, with larger gains over sparse rewards on LIBERO-Goal and LIBERO-10. On LIBERO-Goal, it reaches 94.6%, compared with 86.1% for the sparse-reward baseline and 86.2% for RoboMeter; on LIBERO-10, the corresponding rates are 89.0%, 82.7%, and 77.9%. The improvements therefore extend across the evaluated suites rather than being concentrated in a single setting.

The learning curves further reveal benefits that are not fully captured by the final success rates. On LIBERO-Object and LIBERO-Spatial, GRVM reaches high success rates earlier and achieves the best final performance. On LIBERO-Goal, GRVM maintains a clear advantage over much of training. On LIBERO-10, GRVM establishes a sustained advantage in the later stages of training. These patterns show that GRVM feedback improves both learning efficiency and final policy performance.

4.3 Policy Improvement on Real Robots

Tasks and setup. We evaluate online policy improvement on four real-robot tasks: *Cup place*, *Drawer assemble*, *Block insert*, and *Bag unzip*, covering precise placement, articulated-object manipulation, insertion, and deformable-object manipulation. We compare the supervised fine-tuned (SFT) policy with the policy obtained after online reinforcement learning under the same task initialization and evaluation protocols. Following the two-stage RLT workflow implemented in RLinf, we first jointly fine-tune $\pi_{0.5}$ (Intelligence et al., 2025) and an RLT token transformer on 50 teleoperated demonstrations per task. This stage learns the initial SFT policy and a compact representation for subsequent reinforcement learning. Both the SFT and GRVM-guided policies are evaluated over 15 trials per task.

Online policy improvement. In the second stage, we freeze the fine-tuned $\pi_{0.5}$ and RLT module and train a lightweight MLP actor and critic through real-robot interaction. The actor is conditioned on the RLT representation, robot proprioception, and reference action chunk produced by the frozen VLA policy. We adapt RLinf's observation and control interfaces to our AgileX platform. After each rollout, the frozen GRVM scores the recorded visual trajectory conditioned on the task instruction. We convert these scores into dense rewards using Eq. 8 and use the resulting transitions for off-policy actor–critic updates. Stage boundaries, failure labels, and goal poses are not required during online interaction. Hardware integration and optimization details are provided in Appendix E.

Table 2: Real-robot closed-loop policy improvement. Each entry reports task success rate (%).

Method	Cup Place	Drawer Assemble	Block Insert	Bag Unzip	Avg.
SFT	60.00	46.67	53.33	60.00	55.00
GRVM-guided RL	**100.00**	**86.67**	**93.33**	**100.00**	**95.00**

Results. As reported in Table 2, online RL increases the average success rate from 55.0% to 95.0%. The resulting policies achieve 100% success on Cup place and Bag unzip, 86.67% on Drawer assemble, and 93.33% on Block insert. These results demonstrate consistent improvements over the SFT initialization across all four real-world manipulation tasks.

4.4 Ablation Studies

We ablate failure data and pose-closeness supervision in offline evaluation and closed-loop RL.

Offline supervision ablation. We evaluate the contributions of failure data and pose-closeness supervision on the offline benchmark (Table 3). Removing failure data substantially reduces SD-F from 0.692 to 0.330 and EO from 0.686 to 0.345, despite small improvements in MAE and SD-R. Removing pose-closeness supervision increases MAE from 0.164 to 0.190 and lowers Triplet Accuracy from 0.556 to 0.245. These results show that scalar agreement alone does not ensure reliable state ordering, highlighting the importance of failure examples and goal-relative supervision.

Table 3: **Offline supervision ablations.** Pose denotes pose-closeness supervision.

Variant	MAE ↓	SD-F ↑	SD-R ↑	EO ↑	TA ↑
GRVM	0.164	**0.692**	0.813	**0.686**	**0.556**
w/o Failure	**0.155**	0.330	**0.818**	0.345	0.344
w/o Pose	0.190	0.412	0.664	0.449	0.245

Closed-loop supervision ablation. All variants use the same policy initialization, RL algorithm, interaction budget, and evaluation protocol on LIBERO-Object (Table 4). Compared with sparse-reward RL, GRVM reduces the PPO updates needed to reach 95% success from 55 to 36 and improves mean success over the last ten evaluations from 96.23% to 97.82%. Removing failure data increases the updates to 50; additionally removing pose-closeness supervision increases them to 57 and lowers last-ten success to 95.51%. Both supervision sources therefore improve reward feedback for policy learning.

Table 4: **LIBERO-Object supervision ablations.** Last-10 / σ: last-ten mean / temporal std.; @95: PPO updates to 95% success (smoothing 0.6).

Method	Mean ↑	Last-10 ↑	@95 ↓	σ ↓
RLinf	89.39	96.23	55	0.83
GRVM	**91.92**	**97.82**	**36**	0.39
w/o Failure	90.72	97.01	50	**0.24**
w/o Failure & Pose	89.33	95.51	57	0.95

5 Conclusion

We presented GRVM, a robot value model that separately predicts task progress, goal-relative state quality, and trajectory outcome. A human-audited annotation pipeline constructs dense supervision from task stages and end-effector poses across successful, failed, and recovered executions, enabling pretraining on 286,316 robot trajectories. Evaluation on a balanced benchmark of 1,000 held-out trajectories demonstrates improved agreement with continuous reference annotations and more accurate state ordering around failures and recoveries. In closed-loop policy learning, GRVM-guided rewards increase mean success across four LIBERO suites from 89.8% to 94.3% while accelerating learning, and improve mean success across four real-robot tasks from 55.0% to 95.0% over the initial SFT policies. Together, these results support goal-relative supervision as an effective approach to learning robot value models that capture execution quality beyond nominal task progress and provide useful feedback for policy improvement.

AI USE STATEMENT

Generative AI tools assisted with drafting, language refinement, literature organization, and presentation of the manuscript. Their use in trajectory annotation is described in Section 3.1, including the human-audit procedure. The authors are responsible for the accuracy of the reported methods, results, interpretations, and references.

ETHICS STATEMENT

This work uses robot interaction data from multiple existing open-source datasets together with our collected and processed AgileX trajectories. We follow the applicable licenses and usage terms of the original data sources and do not intentionally use personally identifiable information as supervision. For real-robot experiments, all data collection and policy evaluation are conducted under controlled laboratory conditions with appropriate safety constraints. GPT-based trajectory annotation is used only to assist semantic stage, failure, and recovery annotation, and its outputs are subject to iterative human verification before being used for training. The training data sources and composition are summarized in Appendix A.1 and Table 5.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the training data composition and preprocessing, reference-guided trajectory annotation procedure, failure and recovery handling, dense-target construction, model architecture and training objectives, benchmark splits, evaluation metrics, and closed-loop policy-improvement protocols. The appendix further specifies the construction of progress and goal-relative pose supervision and the human-audit procedure used for automated annotations. We intend to release the implementation, evaluation scripts, and benchmark annotations and model checkpoints that can be redistributed under the licenses of their respective data sources.

REFERENCES

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

Arthur Allshire, Himanshu Gaurav Singh, Ritvik Singh, Adam Rashid, Hongsuk Choi, David McAllister, Justin Yu, Yiyuan Chen, Huang Huang, Pieter Abbeel, et al. Scalable behavior cloning with open data, training, and evaluation. *arXiv preprint arXiv:2606.27375*, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

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

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

Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, et al. Scaling data-driven robotics with reward sketching and batch reinforcement learning. *arXiv preprint arXiv:1909.12200*, 2019.

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, et al. Robodojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *The International Journal of Robotics Research*, 39(2-3):202–216, 2020.

Dongchi Huang, Hongyin Zhang, Bohan Hou, Siteng Huang, Zhian Su, Hang Guo, Tong Lu, Zhaofeng Xu, Jiahao Tang, Jianfei Yang, et al. Rynnvalue: Scaling robotic value foundation models with temporal distance. *arXiv preprint arXiv:2608.09853*, 2026.

Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518. Springer, 1992.

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

Wonje Jeung, Sangyeon Yoon, Hyesoo Hong, Yoonjun Cho, Dongjae Jeon, Bumjun Kim, Jean Oh, Youngjae Yu, and Albert No. Same trajectory, contradictory rewards (robormbench): Paraphrase fragility in vision language reward models. *arXiv preprint arXiv:2609.05401*, 2026.

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

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Kimin Lee, Laura M Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6152–6163. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/lee21i.html.

Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. Roboreward: General-purpose vision-language reward models for robotics. *arXiv preprint arXiv:2601.00675*, 2026.

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S Huang, Luke Zettlemoyer, Dieter Fox, et al. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. *arXiv preprint arXiv:2603.02115*, 2026.

Jacky Liang, Xianyi Cheng, and Oliver Kroemer. Learning preconditions of hybrid force-velocity controllers for contact-rich manipulation. *arXiv preprint arXiv:2206.12728*, 2022.

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

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

Yecheng Jason Ma, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning*, pp. 23301–23320. PMLR, 2023.

Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, et al. Vision language models are in-context value learners. In *International Conference on Learning Representations*, volume 2025, pp. 33984–34009, 2025.

Yiming Mao, Zixi Yu, Weixin Mao, Yinhao Li, Qirui Hu, Zihan Lan, Minzhao Zhu, and Hua Chen. Arm: Advantage reward modeling for long-horizon manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 4468–4477, June 2026.

Soroush Nasiriany, Sep Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations*, volume 2026, pp. 98643–98667, 2026.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

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

Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Yuheng Ji, Cheng Chi, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, et al. Robo-dopamine: General process reward modeling for high-precision robotic manipulation. *arXiv preprint arXiv:2512.23703*, 2025.

AgiBot World Team. Agibot world 2026. `https://huggingface.co/datasets/agibot-world/AgiBotWorld2026`, 2026.

Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson. Rl-vlm-f: Reinforcement learning from vision language foundation model feedback. *arXiv preprint arXiv:2402.03681*, 2024.

Yijie Xu, Haopeng Jin, Run Zhou, Shengbang Liu, Sixiang Chen, Hongyang Cheng, Sicheng Hu, Peterson Co, Jinwen Luo, Huajie Tan, et al. Robo-dopamine 2.0: History-conditioned and ood-aware process reward modeling for robotic manipulation. *arXiv preprint arXiv:2608.15680*, 2026.

Xianchao Zeng, Xinyu Zhou, Youcheng Li, Jiayou Shi, Tianle Li, Liangming Chen, Lei Ren, and Yong-Lu Li. Diagnose, correct, and learn from manipulation failures via visual symbols. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 42386–42395, 2026.

Shaopeng Zhai, Qi Zhang, Tianyi Zhang, Fuxian Huang, Haoran Zhang, Ming Zhou, Shengzhe Zhang, Litao Liu, Sixu Lin, and Jiangmiao Pang. A vision-language-action-critic model for robotic real-world reinforcement learning. *arXiv preprint arXiv:2509.15937*, 2025.

Shaopeng Zhai, Qi Zhang, Tianyi Zhang, Haoran Zhang, Fuxian Huang, Zhanhui Lin, Zijun Xu, and Weinan Zhang. HELP: Human-efficient large-scale robot post-training with rollout segmentation. *arXiv preprint arXiv:2607.09776*, 2026.

Jiahui Zhang, Yusen Luo, Abrar Anwar, Sumedh Anand Sontakke, Joseph J Lim, Jesse Thomason, Erdem Biyik, and Jesse Zhang. Rewind: Language-guided rewards teach robot policies without new demonstrations. *arXiv preprint arXiv:2505.10911*, 2025.

Jianshu Zhang, Chengxuan Qian, Haosen Sun, Haoran Lu, Dingcheng Wang, Letian Xue, and Han Liu. Progresslm: Towards progress reasoning in vision-language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11243–11271, 2026.

A TRAJECTORY ANNOTATION DETAILS

This section provides implementation details for the reference-guided trajectory annotation pipeline described in Section 3.1. The pipeline converts recorded trajectories into semantic stage boundaries, failure anchors, rollback-stage assignments, and recovery relations. These structural annotations are subsequently used to construct the dense progress and goal-relative pose supervision introduced in Section 3.2.

A.1 TRAINING DATA COMPOSITION

Our training mixture contains 286,316 trajectories spanning approximately 2,612 hours and 1,168 source-scoped tasks. The data combine large-scale successful robot demonstrations with trajectories containing failures and recoveries. Table 5 summarizes the composition of the training mixture.

A.2 TRAJECTORY PREPROCESSING

Each trajectory contains temporally ordered visual observations and a task instruction, together with robot-state information when provided by the source dataset. Before annotation, we remove trajectories that are too short to contain a meaningful manipulation event.

We denote the minimum retained trajectory duration by

$$D_{\min} = 5\ \mathrm{s}, \tag{9}$$

and discard trajectories satisfying

$$D < D_{\min}, \tag{10}$$

where D denotes the trajectory duration.

For pose-based supervision, we additionally require temporal correspondence between visual observations and robot-state measurements. Frames without a valid synchronized end-effector pose are excluded from pose-closeness supervision, while the remaining valid observations can still be retained for semantic trajectory annotation.

A.3 REFERENCE-GUIDED GPT ANNOTATION

We use a GPT-based annotator guided by task-specific prompts and human-annotated reference examples. For each trajectory, the annotator receives the task instruction and temporally ordered visual observations and predicts its semantic manipulation structure.

For a trajectory containing K stages, the annotator produces

$$\mathcal{E} = \{\mathcal{E}_j = [b_j, u_j]\}_{j=0}^{K-1}, \tag{11}$$

where b_j and u_j are the start and end frames of stage j. The ordered stages do not overlap:

$$0 = b_0 \le u_0 < b_1 \le u_1 < \cdots < b_{K-1} \le u_{K-1} = T - 1. \tag{12}$$

The stage decomposition is semantic rather than determined by a fixed low-level state threshold. Depending on the task, stages may correspond to interaction phases such as approaching, grasping, transporting, aligning, inserting, or releasing an object. The resulting stage structure is used only to construct supervision and is not provided as an additional input to GRVM.

A.4 FAILURE AND ROLLBACK-STAGE ANNOTATION

For each annotated stage $\mathcal{E}_j = [b_j, u_j]$, the GPT annotator determines whether the execution remains on a valid task branch.

If no failure occurs within the stage, we set

$$f_j = \varnothing. \tag{13}$$

Otherwise, the annotator identifies a failure frame

$$f_j \in [b_j, u_j], \tag{14}$$

Table 5: **Composition of the training data used in this work.** The mixture contains 286,316 trajectories spanning approximately 2,612 hours and 1,168 tasks. Non-success data include failure and recovery trajectories.

Data Source	# Trajectories	Hours	# Success	# Non-success	Non-success Type
AgiBot2026 (Team, 2026)	45,102	265.5	43,199	1,903	Failure / recovery
DROID (Khazatsky et al., 2024)	68,625	362.4	59,175	9,450	Failure
RoboDojo (Chen et al., 2026)	5,300	35.6	5,300	–	–
ViFailback (Zeng et al., 2026)	5,186	12.0	654	4,532	Failure
VLAC-Cut (Zhai et al., 2026)	9,586	117.2	8,185	1,401	Failure / recovery
ABC-130K (Allshire et al., 2026)	46,282	1,210.0	46,282	–	–
RoboCasa (Nasiriany et al., 2026)	57,137	609.0	57,137	–	–
OXE Subset (O'Neill et al., 2024)	18,304	–	18,304	–	–
AgiBot2025 Subset (Bu et al., 2025)	30,794	–	30,794	–	–
Total	**286,316**	**2,611.6**	**269,030**	**17,286**	–
Duration by outcome: success $\approx 2,510.4$ h; non-success ≈ 101.2 h.					

defined as the earliest frame at which the execution visibly departs from valid task progression. The failure frame therefore represents the onset of the failed branch rather than the final visible consequence of the failure.

The annotator additionally identifies a rollback-stage index

$$r_j < j, \tag{15}$$

which specifies the latest previously valid stage whose state provides an appropriate reference for post-failure supervision. For example, a failure observed during placement may require returning to an earlier grasp or alignment stage before the task can proceed correctly.

For each detected failure, the annotator is also required to explain the failure cause and the selected rollback stage. These explanations are used only for human auditing and prompt refinement and are not used as GRVM training targets.

A.5 Recovery Identification

A failure does not necessarily terminate the useful portion of a trajectory. We therefore additionally identify whether the execution subsequently returns to a valid task branch.

For a stage containing failure frame f_j, a recovery is identified when the later execution re-enters valid task progression and reaches a valid realization of the interrupted or downstream manipulation stage. We denote the corresponding recovered stage by

$$\mathcal{E}_j^{\mathrm{rec}} = [b_j^{\mathrm{rec}}, u_j^{\mathrm{rec}}]. \tag{16}$$

The structural annotation for a trajectory can therefore be written as

$$\mathcal{A}(\tau) = \left\{ \left(b_j, u_j, f_j, r_j, b_j^{\mathrm{rec}}, u_j^{\mathrm{rec}} \right) \right\}_{j=0}^{K-1}, \tag{17}$$

where failure- and recovery-specific entries are defined only when applicable.

A.6 Iterative Human Audit and Refinement

Annotation and auditing are conducted separately for each dataset and task category. At each refinement round, humans randomly inspect 10 annotated trajectories. A round passes only when all 10 sampled trajectories are judged correct.

For each trajectory, the audit checks whether (i) the semantic stage decomposition matches the observed manipulation process, (ii) the failure frame correctly identifies the onset of the failed branch, (iii) the rollback stage represents an appropriate previously valid state, and (iv) the recovery relation, when present, matches the subsequent execution.

If any of the 10 annotations fails the audit, the human annotator provides the corrected annotation together with an explanation of why the original annotation was incorrect and how it should be

corrected. These corrections and explanations are incorporated into the corresponding task-specific prompt and reference examples before the next annotation round.

We perform 3–5 annotation–audit–refinement rounds for each dataset, and no dataset requires more than five rounds. The resulting annotations are used for dense-target construction only after satisfying the 10/10 audit criterion.

A.7 Automatic Consistency Checks

Before dense-target construction, we additionally apply deterministic checks to the structural annotations. We require

$$b_j \le u_j, \tag{18}$$
$$u_j < b_{j+1}, \tag{19}$$
$$b_j \le f_j \le u_j \quad \text{if } f_j \neq \varnothing, \tag{20}$$
$$r_j < j \quad \text{if } f_j \neq \varnothing. \tag{21}$$

For a recovered trajectory, $\mathcal{E}_j^{\mathrm{rec}}$ must occur after the corresponding failure frame and must be compatible with the annotated task progression. We also verify that every end-effector pose used for geometric supervision is available in the synchronized robot-state record.

Annotations that violate these constraints are returned for re-annotation before training.

A.8 Pose Grounding

GPT is used only to infer the discrete semantic structure of the trajectory. It does not predict continuous geometric targets. Both the current end-effector pose and the target pose used to construct pose-closeness supervision are obtained directly from recorded robot states.

For a stage without failure, the terminal end-effector pose is used as the local stage goal:

$$G_j = P_{u_j}. \tag{22}$$

For a failed stage followed by a valid recovery, the failed terminal pose is not treated as a valid target. Instead, the terminal pose of the corresponding recovered stage is used:

$$G_j = P_{u_j^{\mathrm{rec}}}. \tag{23}$$

This target is used to construct pose-closeness supervision for the portion of the interrupted stage preceding f_j.

If no valid recovery exists, the terminal pose of the failed branch is not used as a stage goal. In this case, the pose-closeness target remains anchored to the preceding valid stage, as detailed in Appendix B.

Thus, GPT determines which observed state constitutes a valid semantic reference, while all continuous poses used to construct supervision are grounded directly in recorded robot states.

B Dense Target Construction Details

This section provides additional implementation details for the progress and goal-relative pose-closeness targets introduced in Section 3.2.

Pose distance. We represent the end-effector pose as $P_t = (\mathbf{x}_t, \mathbf{R}_t)$ and the stage goal as $G_j = (\mathbf{x}_j^\star, \mathbf{R}_j^\star)$, where $\mathbf{x} \in \mathbb{R}^3$ denotes position and $\mathbf{R} \in \mathrm{SO}(3)$ denotes orientation. Both poses are expressed in the same coordinate frame.

The pose distance combines translational and rotational discrepancies:

$$D(P_t, G_j) = \frac{\left\|\mathbf{x}_t - \mathbf{x}_j^\star\right\|_2}{s_{\mathrm{pos}}} + \frac{d_{\mathrm{rot}}(\mathbf{R}_t, \mathbf{R}_j^\star)}{s_{\mathrm{rot}}}, \tag{24}$$

where

$$d_{\mathrm{rot}}(\mathbf{R}, \mathbf{R}^\star) = \arccos\left(\mathrm{clip}\left(\frac{\mathrm{tr}(\mathbf{R}^\top \mathbf{R}^\star) - 1}{2}, -1, 1\right)\right) \tag{25}$$

is the geodesic angular distance on $\mathrm{SO}(3)$.

The positive scale parameters s_{pos} and s_{rot} balance position errors measured in meters and orientation errors measured in radians, yielding a dimensionless pose distance. The same distance definition is used for all trajectories. We set $s_{\mathrm{pos}} = 0.10\,\mathrm{m}$ and $s_{\mathrm{rot}} = 0.50\,\mathrm{rad}$.

The same metric is used in both the numerator and denominator of the normalized pose-closeness target. Therefore, the current pose error is measured relative to the pose error at the beginning of the stage. Including both translational and rotational terms prevents positional alignment alone from implying full pose-closeness when substantial orientation error remains.

Valid stage goals. For a valid stage without failure, the local target is the recorded end-effector pose at the terminal frame:

$$G_j = P_{u_j}. \tag{26}$$

For a failed stage followed by recovery, the target for its pre-failure portion is instead

$$G_j = P_{u_j^{\mathrm{rec}}}. \tag{27}$$

Although the recovered state is available during offline target construction, GRVM remains causal: future observations are not provided when predicting the value of the current frame.

For a failed stage without a valid recovery, no goal pose is taken from its terminal state.

Within-stage pose-closeness. Given a valid stage goal G_j, we first compute the normalized reduction in pose error:

$$z_t(G_j) = \mathrm{clip}\left(1 - \frac{D(P_t, G_j)}{\max\left(D(P_{b_j}, G_j), \epsilon\right)}, 0, 1\right), \tag{28}$$

where $\epsilon > 0$ prevents numerical instability when the stage begins close to its goal.

The within-stage pose-closeness is

$$r_t(G_j) = h(z_t(G_j)), \tag{29}$$

where $h : [0,1] \to [0,1]$ is the sigmoid-based monotonic mapping used in the main formulation. It redistributes intermediate values while preserving their ordering by goal-relative pose error.

Nominal targets. For frame t in stage $\mathcal{E}_j = [b_j, u_j]$, the within-stage temporal progress is

$$\alpha_t = \frac{t - b_j}{u_j - b_j}. \tag{30}$$

The corresponding nominal targets are

$$p_t^{\mathrm{nom}} = \frac{j + \alpha_t}{K}, \qquad c_t^{\mathrm{nom}} = \frac{j + r_t(G_j)}{K}. \tag{31}$$

For a stage without failure, we directly use

$$p_t^\star = p_t^{\mathrm{nom}}, \qquad c_t^\star = c_t^{\mathrm{nom}}. \tag{32}$$

Failure-aware progress. For a stage containing failure frame f_j, progress follows the nominal target up to the failure. For frames remaining on the failed branch after f_j, we use

$$p_t^\star = \max\left(0, p_{f_j}^{\mathrm{nom}} - \frac{t - f_j}{T}\right), \qquad t > f_j, \tag{33}$$

where T denotes the trajectory length.

Thus, the target decreases by $1/T$ for every subsequent frame on the failed branch. The lower bound prevents the progress target from becoming negative.

Failed stage without recovery. When a failed stage has no later valid recovery, its terminal pose is not treated as a physical goal. Let $\mathcal{E}_{j^-}$ denote the most recent valid stage preceding the failed stage. We define its valid pose-closeness level as

$$c_j^{\text{prev}} = \max_{s \in \mathcal{E}_{j^-}} c_s^\star. \tag{34}$$

The pose-closeness target is held at this value before the failure:

$$c_t^\star = c_j^{\text{prev}}, \qquad b_j \le t \le f_j. \tag{35}$$

This prevents the supervision from assigning increasing state quality toward an endpoint that is known to belong to a failed execution.

Rollback-stage pose target. After failure, the annotated rollback stage r_j determines the pose-closeness level associated with the state to which the execution should return. We define

$$c_j^{\text{back}} = \max_{s \in \mathcal{E}_{r_j}} c_s^\star. \tag{36}$$

For frames remaining on the failed branch,

$$c_t^\star = c_j^{\text{back}}, \qquad t > f_j. \tag{37}$$

Using a fixed rollback-stage anchor avoids interpreting arbitrary motion within the failed branch as meaningful improvement toward an invalid local goal.

Recovery transition. Let t_j^{rec} denote the first frame at which the trajectory returns to a valid task branch. The failure-aware correction applies only while the trajectory remains on the failed branch.

For $t \ge t_j^{\text{rec}}$, the progress and pose-closeness targets are reconstructed from the corresponding valid stage using Eq. 31. A recovered execution can therefore resume increasing progress and pose-closeness once it returns to valid task progression.

C GRVM Training Details

Feature encoding and fusion. We freeze the Qwen3-VL-4B visual tower and the language encoder, while optimizing the remaining model parameters. Each time–view pair is encoded independently by the visual tower. After the vision merger, spatial tokens are mean-pooled, projected to 384 dimensions, and augmented with a camera embedding. A learned query pools the available views through masked multi-head attention, producing one state token per time step. This token then queries the full instruction token sequence through cross-attention, yielding the task-conditioned token z_t. Language conditioning uses token-level features rather than a pooled sentence embedding.

Causal processing and readout. The trajectory transformer receives $[z_0, \ldots, z_{T-1}, z_{\text{out}}]$, where z_{out} is a learnable outcome token. An upper-triangular causal mask blocks attention to future positions, and a key-padding mask excludes invalid frame tokens. Thus, ordinary frame tokens cannot attend to future frames or to the terminal outcome token, whereas the outcome token can attend to all valid frames. Frame-wise visual encoding has no cross-time attention, so future RGB observations cannot enter earlier visual features. The complete sequence is processed in a single masked forward pass rather than by separately forwarding every prefix. After the final LayerNorm, the first T positions provide h_t for the progress and pose-closeness heads; the last position provides h_{out} for the two-class outcome head.

Input sampling and masking. Training trajectories contain at most 128 sampled frames. We retain all frames when the sequence length is at most 128; for longer trajectories, sampling preserves the first and last frames and annotated event/phase-change points, then fills the remaining slots by iteratively sampling the largest temporal gaps. RGB images are resized to 224×224 using bicubic interpolation and normalized with channel-wise mean and standard deviation 0.5. Source camera keys are mapped to up to three canonical slots: main, wrist, and auxiliary views. Missing views are skipped by the visual encoder and masked in view attention; each valid frame must contain at least one valid view. Instructions contain at most 77 tokens, and padded frames are excluded from attention and frame-level supervision.

Optimization. We train GRVM on the annotated multi-source dataset using distributed data parallel on 16 GPUs with bfloat16 mixed precision. Each batch is balanced across sources and contains approximately 50% ordinary-success, 25% terminal-failure, and 25% recovery trajectories; these are sampling proportions rather than the raw dataset proportions. We minimize Eq. 7 with $(\lambda_p, \lambda_c, \lambda_o) = (1.0, 1.0, 1.0)$; Huber terms use PyTorch `smooth_l1` with $\beta = 1$. There is no additional loss on the composite score $\hat{p}_t^{0.4}\hat{c}_t^{0.6}$. Optimization uses AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 10^{-3}), linear warmup for 500 steps, cosine decay to 10% of the peak learning rate, and global gradient clipping at unit ℓ_2 norm. The peak learning rate is 5×10^{-5}; the per-device batch size is 2, giving a global batch size of 32. Training runs for 10,000 steps.

Supervision ablations. Removing failure data leaves the number of training steps and batch size unchanged. For variants without pose-closeness supervision, we use the progress prediction $\hat{p}_t$ directly as the scalar readout for offline evaluation and as the potential Φ_t in Eq. 8, replacing $\hat{p}_t^{0.4}\hat{c}_t^{0.6}$. Thus, the unsupervised pose-closeness output contributes to neither evaluation scores nor rewards.

D Offline Benchmark Protocol and Evaluation Metrics

Benchmark protocol. The offline benchmark contains 1,000 trajectories drawn equally from AgiBot2026 (Team, 2026), DROID (Khazatsky et al., 2024), ViFailBack (Zeng et al., 2026), and VLAC-Cut (Zhai et al., 2026). Each source contributes 125 successful and 125 non-successful trajectories. The held-out manifest is separated from training by leakage group, with related episodes, views, temporal crops, and derived slices kept in the same split. Reference labels are loaded from pre-generated per-episode NPZ files and metadata; they are not recomputed during training or evaluation.

Trajectory sampling and GRVM inference. GRVM uses the same multi-view preprocessing as training and the event/phase-aware frame selection described in Appendix C, retaining at most 128 frames per trajectory. The entire sampled sequence is processed in one forward pass with a causal attention mask. We then select up to 128 endpoints by `linspace` on the subsampled time axis and map them back to the original trajectory timestamps. Each endpoint's composite score is computed from the sigmoid outputs as $\hat{v}_t = \hat{p}_t^{0.4}\hat{c}_t^{0.6}$, without additional normalization. For metrics requiring full-length curves or values at event frames between queried endpoints, we use a causal hold:

$$\hat{y}_t = \hat{y}_{t_{k(t)}}, \qquad k(t) = \max\{k : t_k \le t\}, \tag{38}$$

where $t_1 < \cdots < t_M$ denote the queried endpoints on the original time axis. Frames preceding the first valid endpoint are omitted.

Baseline inference. Baselines use the same benchmark manifest and temporal endpoints, but retain their model-specific observation and inference interfaces. Most vision–language reward models operate on a pre-rendered frame cache and score causal prefixes or successive hops, rather than using GRVM's multi-view, whole-sequence forward pass. Only observations available up to the queried time are supplied for causal prefix scoring. Their native outputs are mapped to $[0, 1]$ using model-specific mappings before computing the common scalar metrics, including MAE and Pearson correlation, against the stored reference labels. Thus, the shared protocol aligns trajectories, endpoints, and references, but does not impose identical view inputs, inference calls, or score mappings across models.

Evaluation readouts. The benchmark evaluates two complementary properties of a value model. Scalar agreement measures compatibility with existing continuous annotations, while relative semantic consistency measures whether predicted values preserve the correct ordering around normal advancement, failure, and recovery.

For GRVM, scalar metrics use the progress prediction

$$\hat{y}_t^{\text{scalar}} = \hat{p}_t, \tag{39}$$

because the available continuous references primarily encode nominal task advancement. Event-sensitive metrics instead use the downstream composite value

$$\hat{v}_t = (\hat{p}_t)^{0.4}(\hat{c}_t)^{0.6}. \tag{40}$$

The evaluator also computes semantic-direction, anchor-ranking, and triplet metrics separately for the composite-value, progress, and pose-closeness readouts, denoted `official_value`, `progress`, and `pose_closeness` in the evaluation outputs. The composite reference uses the same exponents, $v_t^* = (p_t^*)^{0.4}(c_t^*)^{0.6}$. These are evaluation readouts, not additional network heads. For single-output baselines, the mapped scalar prediction is used for both metric groups.

Reference annotations determine valid comparison frames, event phases, and semantic anchors. The event-sensitive metrics depend only on relative ordering and therefore do not require predictions to match a particular numerical calibration.

Scalar agreement. Let $\mathcal{I}_\tau$ denote the valid evaluated frames of trajectory τ, $\hat{y}_t$ the predicted scalar value, and y_t the corresponding continuous reference annotation.

Value-Order Correlation (VOC) is the Spearman rank correlation between prediction and reference:

$$\mathrm{VOC}(\tau) = \rho_{\mathrm{S}}\left(\{\hat{y}_t\}_{t\in\mathcal{I}_\tau}, \{y_t\}_{t\in\mathcal{I}_\tau}\right). \tag{41}$$

VOC measures whether the predicted values preserve the ordering of the continuous reference while being insensitive to absolute calibration.

Pearson correlation measures linear agreement:

$$\mathrm{Pearson}(\tau) = \rho_{\mathrm{P}}\left(\{\hat{y}_t\}_{t\in\mathcal{I}_\tau}, \{y_t\}_{t\in\mathcal{I}_\tau}\right), \tag{42}$$

and mean absolute error evaluates numerical calibration:

$$\mathrm{MAE}(\tau) = \frac{1}{|\mathcal{I}_\tau|}\sum_{t\in\mathcal{I}_\tau} |\hat{y}_t - y_t|. \tag{43}$$

Trajectories for which a correlation is undefined because either sequence has zero variance are omitted from that metric.

Semantic Direction. Contiguous valid-advancement, failure, and recovery regions are obtained from the reviewed event annotations. We use $\epsilon = 0.02$ to exclude comparisons whose reference difference is too small to define an unambiguous direction.

For a valid-advancement region $r = [s_r, e_r]$ satisfying $|y_{e_r} - y_{s_r}| > \epsilon$, we define

$$h_r^S = \mathbb{1}\left[\mathrm{sign}(\hat{v}_{e_r} - \hat{v}_{s_r}) = \mathrm{sign}(y_{e_r} - y_{s_r})\right]. \tag{44}$$

For each failure event, the reference annotation identifies a pre-failure anchor a_r and a degraded-state anchor b_r. Failure direction is correct when

$$h_r^F = \mathbb{1}[\hat{v}_{a_r} > \hat{v}_{b_r}], \tag{45}$$

provided that $y_{a_r} - y_{b_r} > \epsilon$.

For a subsequent recovery ending at anchor c_r, recovery direction is correct when

$$h_r^R = \mathbb{1}[\hat{v}_{c_r} > \hat{v}_{b_r}], \tag{46}$$

provided that $y_{c_r} - y_{b_r} > \epsilon$.

SD-S, SD-F, and SD-R are obtained by averaging the corresponding indicators within each trajectory and then macro-averaging over trajectories containing at least one eligible region.

Triplet Accuracy. For a complete pre-failure–failure–recovery triplet (a_r, b_r, c_r), we require both degradation and subsequent recovery to be ordered correctly:

$$h_r^{\mathrm{tri}} = \mathbb{1}\left[\hat{v}_{a_r} > \hat{v}_{b_r} \ \wedge\ \hat{v}_{c_r} > \hat{v}_{b_r}\right]. \tag{47}$$

Triplet Accuracy is the mean of h_r^{tri} over complete eligible triplets, first within each trajectory and then across trajectories. Unlike the individual Semantic Direction metrics, TA requires the model to identify both deterioration and recovery within the same execution structure.

Anchor Kendall. We construct a sparse semantic anchor set containing the first and last valid frames, pre-failure anchors, degraded-state anchors, and recovery endpoints. We then compute tie-corrected Kendall correlation between reference and predicted values over these anchors:

$$\tau_b = \frac{N_{\mathrm{con}} - N_{\mathrm{dis}}}{\sqrt{N_{\mathrm{untied},y} N_{\mathrm{untied},\hat{v}}}}, \tag{48}$$

where N_{con} and N_{dis} denote the numbers of concordant and discordant anchor pairs, and the denominator accounts for ties in the reference and predicted sequences. Anchor Kendall therefore evaluates global ordering over semantically important states without weighting every trajectory frame equally.

Event Ordering. Semantic Direction evaluates only selected anchor states. To measure whether the same ordering holds over local neighborhoods around each event, we additionally define window-level pairwise ordering.

For two local frame windows A and B, let

$$\mathrm{Ord}(A, B) = \frac{1}{|A||B|} \sum_{i \in A} \sum_{j \in B} \mathbb{1}[\hat{v}_i > \hat{v}_j]. \tag{49}$$

Using up to $w = 5$ valid frames around each semantic anchor, failure ordering is computed as

$$\mathrm{Ord}(W_{\mathrm{pre}}, W_{\mathrm{fail}}), \tag{50}$$

while recovery ordering is

$$\mathrm{Ord}(W_{\mathrm{rec}}, W_{\mathrm{fail}}). \tag{51}$$

Event Ordering is the mean of the available failure and recovery terms. We additionally record $\mathrm{Ord}(W_{\mathrm{rec}}, W_{\mathrm{pre}})$ as a diagnostic of whether the recovered execution returns to the pre-failure value level, but this diagnostic is not included in the main benchmark score.

Aggregation. All metrics are computed using only finite and valid entries. Event- and region-level measurements are first averaged within each canonical trajectory so that trajectories containing more frames or more events do not receive greater weight. Benchmark results are then obtained by macro-averaging the resulting trajectory-level scores.

E Real-Robot Experimental Setup

Robot platform. Our real-robot experiments use a dual-arm AgileX PiperX manipulation platform (Fig. 7). Each arm is equipped with a parallel gripper and is controlled independently, allowing both single-arm and coordinated bimanual manipulation. Three RealSense D435IF cameras provide synchronized main, left, and right RGB views at 640×360 resolution and 30 Hz. Depth images are not used by the policy or GRVM. During policy improvement, GRVM receives the deployment-time RGB observations and task instruction; no robot pose, object pose, semantic stage, or failure/recovery annotation is provided as input.

Demonstration collection. Initial demonstrations are collected through master–slave teleoperation. Two master PiperX arms are manually operated by a human demonstrator, and their joint motions are mapped to the corresponding slave arms on the robot platform. This interface allows the demonstrator to directly control arm motion and gripper state while observing the task execution from the same workspace used for policy deployment. The collected trajectories contain synchronized visual observations and robot actions and are used to train the initial SFT policy.

Observation and action interface. The observation space and camera configuration are kept fixed throughout SFT, online policy optimization, and final evaluation. Robot proprioception and each action are 14-dimensional: six joint values and one gripper value per arm. Joint actions are represented as normalized deltas and decoded into absolute joint targets, while gripper actions specify absolute binary targets. The frozen VLA generates a 30-step reference chunk, and the Stage 2 actor predicts a 10-step chunk conditioned on its first ten steps. Actions are spaced at 30 Hz and interpolated for the 200 Hz hardware command stream. Inactive-arm commands are masked for single-arm tasks. The observation and action interfaces are shared by the compared policies. GRVM scores the recorded visual trajectory conditioned on the task instruction and does not require privileged robot or object states at inference time.

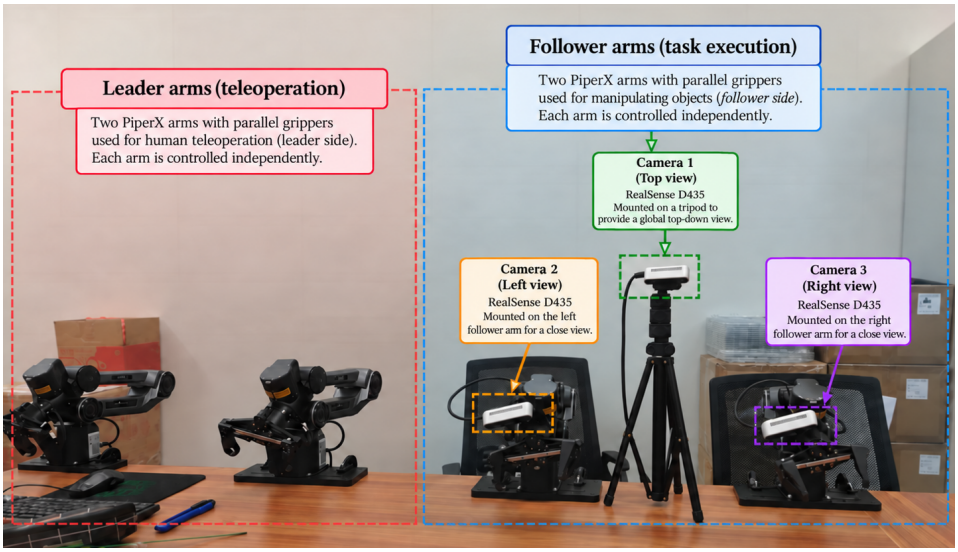


Figure 7: Dual-arm AgileX PiperX platform.

Task setup and resets. We evaluate four manipulation tasks: *Cup place*, *Drawer assemble*, *Block insert*, and *Bag unzip*. They respectively emphasize precise placement, articulated-object manipulation, constrained insertion, and deformable/contact-rich interaction. For each task, object identities, workspace layout, camera placement, language instruction, and reset distribution are fixed across methods. After each rollout, the workspace is manually reset according to the same task-specific initialization protocol.

Stage 1: supervised adaptation. We jointly fine-tune $\pi_{0.5}$ and the RLT token transformer using 50 teleoperated demonstrations per task. The RLT module has two layers, eight attention heads, a hidden dimension of 2,048, and one RL token. Its reconstruction objective is combined with the VLA objective with weight 1.0. Stage 1 uses AdamW with learning rate 10^{-5}, $(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay 10^{-10}, and gradient clipping at 1.0. The global batch size is 64. The learning-rate schedule has 1,000 warmup steps followed by cosine decay, with a minimum learning-rate ratio of 0.25.

Stage 2: online policy improvement. We freeze the Stage 1 VLA and RLT module and optimize an MLP actor and twin-Q critic using replayed real-robot transitions. The actor receives the 2,048-dimensional RLT representation, robot proprioception, and the reference action chunk; the critic receives the representation, proprioception, and candidate action chunk. The actor objective combines Q maximization and a reference-action behavioral-cloning term, weighted by 0.1 and 5.0, respectively. Reference-action dropout is 0.5 during actor training. Actor and critic learning rates are both 3×10^{-4}, with gradient clipping at 10.0 and one critic update per actor update. The discount factor is 0.96, and target critics use soft updates with coefficient 0.005. The configured learner batch size is 1, with a replay sampling window of 20 and a minimum replay size of 1; target and rollout-policy weights are updated every learner step. Entropy regularization is disabled. For GRVM-guided learning, each completed rollout is scored by the frozen GRVM conditioned on the task instruction. The scores are converted into dense rewards using Eq. 8; neither the VLA, the RLT module, nor GRVM is updated in this stage. Task resets, safety constraints, and the final evaluation protocol remain the same across the compared policies.

Implementation details. RGB observations and robot states are recorded at 30 Hz. The 30-step VLA reference horizon is distinct from the 10-step Stage 2 action chunk described above. The final

Table 6: PPO configuration for the four LIBERO suites, following the official RLinf π_0 setup. Suite-specific values are ordered as Object / Spatial / Goal / Long.

Parameter	Setting
Policy / value-head learning rate	5×10^{-6} / 10^{-4}
Adam (β_1, β_2); optimizer ϵ	$(0.9, 0.95)$; 10^{-8}
Weight decay	0.01
Global / micro batch size	2,048 / 128
PPO update epochs	4
Policy-ratio / value clipping	0.2 / 0.2
Discount γ / GAE λ	0.99 / 0.95
Entropy bonus / KL penalty coefficient	0 / 0
Parallel training environments	64
Rollout epochs per collection	8
Parallel evaluation environments	500
Maximum episode length	240 / 240 / 320 / 480
Executed action-chunk length	5 / 5 / 5 / 10
Gradient-clipping threshold	1.0 / 1.0 / 1.0 / 2.0

performance is measured over 15 evaluation trials per task, using the same task-specific success criterion and initialization protocol for SFT and GRVM-guided RL.

F Simulation Implementation Details

We follow the official RLinf π_0–LIBERO PPO configurations for Object, Spatial, Goal, and LIBERO-Long (LIBERO-10). Object, Spatial, and Goal use the released `RLinf-Pi0-LIBERO-Spatial-Object-Goal-SFT` initialization; Long uses `RLinf-Pi0-LIBERO-Long-SFT`. Within each suite, sparse-reward RL, RoboMeter-guided RL, and GRVM-guided RL share the policy initialization and PPO settings, differing only in the reward signal specified in Section 4.2.

Policy and optimization. The policy receives main-view and wrist-view RGB observations, robot state, and the task instruction, and outputs seven-dimensional end-effector and gripper actions. We follow the official action-expert fine-tuning configuration and attach a PPO value head. Flow-SDE exploration uses noise level 0.5 and four denoising steps. PPO uses normalized GAE advantages and chunk-level rewards and log probabilities. Table 6 summarizes the shared settings and suite-specific differences. The executed action chunk is distinct from the VLA network's prediction horizon.

Reward comparison. The sparse baseline uses the environment task-success reward. The value-guided variants retain that reward and add the change in the corresponding frozen model's score, using Eq. 8. The PPO critic is optimized together with the policy and is distinct from the frozen GRVM or RoboMeter used to score rollouts. All reward conditions within a suite use the same environment reset and evaluation settings.