PROCVLM: LEARNING PROCEDURE-GROUNDED PROGRESS REWARDS FOR ROBOTIC MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon robotic manipulation requires dense feedback that reflects how a task advances through its procedural stages, not merely whether the final outcome is successful. Existing reward models often rely on trajectory-level success labels or time-based interpolation, which can conflate elapsed time with true task progress and therefore fail to capture unfinished steps, stagnation, and failure states. We present **ProcVLM**, a progress-aware vision-language model that learns procedure-grounded progress as a dense reward signal for manipulation. Rather than deriving progress from terminal outcomes or temporal proxies, ProcVLM grounds progress estimation in procedural structure and intra-stage visual change, and further adopts a reasoning-before-estimation paradigm that infers the remaining atomic actions before estimating task progress. Specifically, we construct this supervision by synthesizing frame-level subtask-semantic annotations, assigning progress budgets according to subtask structure, and distributing each budget based on intra-subtask visual change. To train ProcVLM at scale, we build a standardized procedural supervision synthesis pipeline and construct ProcCorpus-60M from 30 embodied datasets with 60M annotated frames, from which we derive ProcVQA for procedure-aware pretraining, with progress estimation as the central task alongside action segmentation and future planning. Experiments on ProcVQA and reward-model benchmarks show that ProcVLM improves embodied procedural reasoning and yields more discriminative trajectory-internal progress estimates than representative baselines, supporting its use as a dense reward model for downstream reward-guided policy optimization.

1 INTRODUCTION

Recent vision-language-action (VLA) models have substantially improved policy generalization from large-scale robot demonstrations (Brohan et al., 2023; Zitkovich et al., 2023; Collaboration et al., 2023; Ghosh et al., 2024; Kim et al., 2024; Black et al., 2024; Intelligence et al., 2025b; Bjorck et al., 2025; Team, 2025a;b; Yang et al., 2025; Li et al., 2025; Qu et al., 2025). However, demonstration-centric pretraining is constrained by the cost of collecting diverse robot data and the difficulty of adapting pretrained policies to new tasks, environments, and failure modes. This has motivated reward-guided policy improvement paradigms, where VLA policies improve beyond offline demonstrations through autonomous experience, expert corrections, learned reward feedback, or self-improvement signals (Intelligence et al., 2025a; Xiao et al., 2026; Fei et al., 2025; Zhai et al., 2025). Such paradigms require reward models that provide dense, task-conditioned feedback on task progress, remaining steps, stagnation, and failure.

A central challenge is that existing robot datasets rarely provide dense supervision for procedural progress. Most large-scale corpora contain task instructions, observations, actions, and sometimes trajectory-level success labels, but lack explicit annotations of procedural structure, such as subtask boundaries and remaining actions (Collaboration et al., 2023; Khazatsky et al., 2024; Walke et al., 2023). As a result, robotic reward or progress models often resort to indirect supervision from sparse outcomes, time-based interpolation, temporal-difference learning, or comparison- and preference-based signals (Zhai et al., 2025; Lee et al., 2026; Huang et al., 2025; Liang et al., 2026; Tan et al., 2025; Zhang et al., 2024). However, time is not progress in multi-step manipulation. Linear time-based labels alone provide limited guidance for understanding semantic action stages, task structure, within-stage advancement, and task completion. During deployment, a policy may spend many steps

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.

retrying a grasp, remain stuck in a local stage, or recover from an earlier failure. The lack of explicit stage information in linear time-based labels can undermine prediction reliability and limit transfer to new tasks and environments.

Motivated by this observation, we introduce **ProcVLM**, a progress-aware embodied VLM that predicts dense progress rewards grounded in task procedures. A transferable progress estimator should (i) identify the current execution stage and subtask boundaries, (ii) recognize atomic actions shared across tasks and environments, and (iii) understand task instructions and plan subsequent actions. To this end, ProcVLM follows a reasoning-before-estimation paradigm: it first infers the current execution stage and remaining atomic actions, and then predicts a continuous progress score conditioned on this explicit procedural reasoning. To support this learning process, we construct progress labels by assigning each subtask a duration-adjusted progress budget and distributing that budget according to intra-subtask perceptual change. This coupling provides procedural context for interpreting stage transitions, unfinished steps, stagnation, and failure states.

To train ProcVLM, we develop a highly efficient and scalable procedural supervision synthesis pipeline for embodied trajectory annotation. The pipeline decomposes raw manipulation trajectories into procedural subtasks and annotates frame-level cues such as state reasoning, completion status, and remaining actions. Using this pipeline, we construct **ProcCorpus-60M** from 30 embodied manipulation datasets, covering 400K trajectories and 60M annotated frames. We further convert these annotations into **ProcVQA**, a 20B-token VLM training corpus centered on task progress estimation and complemented by action segmentation and future planning as auxiliary process-reasoning tasks. ProcVLM is trained in two stages: large-scale procedure-aware pretraining on ProcVQA to learn general procedural representations, followed by refinement on a curated subset to sharpen subtask alignment and progress estimation. Empirically, ProcVLM improves procedural understanding on ProcVQA and enables one-shot adaptation on RoboFAC (Lu et al., 2025b), yielding more discriminative progress estimates than representative reward-model baselines. Moreover, ProcVLM improves policy performance on RoboTwin (Chen et al., 2025b) through on-policy RL after few-shot reward adaptation. Its rewards also stabilize fine-tuning on noisy real-robot demonstrations, yielding higher early-stage success scores than supervised fine-tuning. Our contributions are:

- We introduce *ProcVLM*, a progress-aware embodied VLM that predicts dense procedure-grounded progress rewards through reasoning-before-estimation, linking continuous progress estimation to subtask semantics, remaining actions, and within-stage advancement.
- We develop a scalable procedural supervision synthesis pipeline for converting raw manipulation trajectories into frame-level subtask-structured annotations. It yields *ProcCorpus-60M*, *ProcVQA*, and a two-stage training pipeline: large-scale pretraining followed by curated refinement.
- We systematically evaluate ProcVLM across procedure-aware understanding, reward modeling, cross-task adaptation, and downstream policy optimization. The results show that procedure-grounded progress estimation enables transferable reward modeling and supports stable policy improvement.

2 Related Work

Embodied VLMs for Procedural Reasoning. A parallel line of work studies how VLMs can support procedural understanding in robotics beyond direct action prediction. Early language-grounded robotics systems use foundation models for high-level planning, affordance grounding, program synthesis, and spatial-constraint reasoning (Ichter et al., 2022; Huang et al., 2022; Liang et al., 2023; Singh et al., 2023; Huang et al., 2023). Models such as PaLM-E, VIMA, and Gato further extend this idea by integrating language, vision, and embodied observations into general-purpose architectures for planning and interaction (Driess et al., 2023; Jiang et al., 2022; Reed et al., 2022). Robotics-oriented supervision and reasoning models further advance grounded long-horizon understanding, with RoboVQA providing robotics video-text supervision, RoboBrain unifying planning, affordance perception, and trajectory prediction, and PROGRESSLM evaluating task-progress reasoning in VLMs (Sermanet et al., 2024; Ji et al., 2025; Zhang et al., 2026b). Recent embodied reasoning models further extend VLMs toward spatial grounding, keypoint reasoning, task decomposition, and reinforced embodied reasoning, as represented by ReKep, Gemini Robotics-ER, and related systems (Team, 2025a;b; Google DeepMind, 2025; Yuan et al., 2024; Fang et al., 2024b; Huang et al., 2024; Yuan et al., 2026).

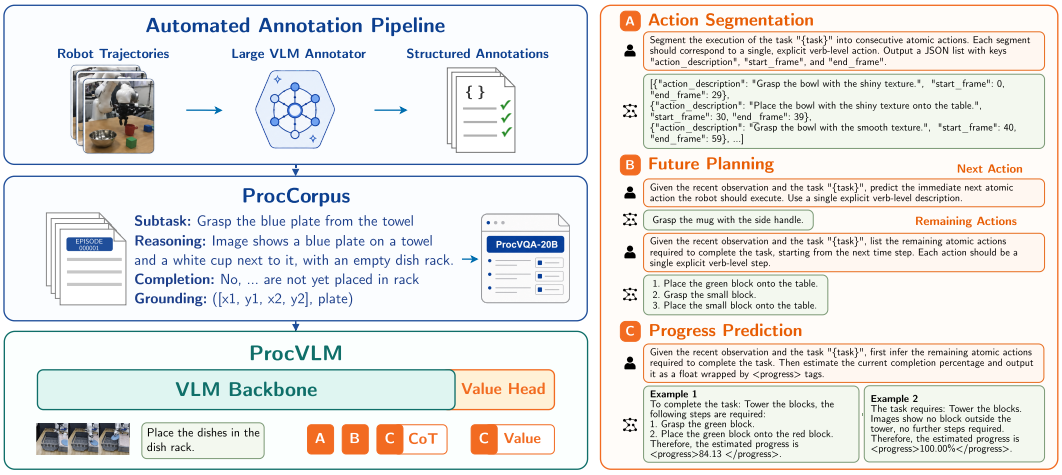


Figure 1: Overview of ProcVLM. A large VLM annotates robot trajectories to form ProcCorpus-60M, which is converted into ProcVQA for action segmentation, future planning, and task progress prediction. Training on ProcVQA enables ProcVLM to provide procedure-grounded progress rewards for downstream policy optimization.

Reward Modeling for Manipulation. Earlier work addresses sparse rewards through relabeling and value learning (Andrychowicz et al., 2017; Huang et al., 2025), or foundation-model feedback (Zhang et al., 2025; Lee et al., 2026; Chen et al., 2026b). Other frameworks use VLM/LLM feedback, VLA critics, or policy-internal signals for policy improvement (Ma et al., 2025; Wu et al., 2026; Yong et al., 2026; Zhai et al., 2025; Zhang et al., 2024; Lu et al., 2025a). For dense rewards, Robometer combines frame-level progress/success prediction with trajectory-comparison preferences, while RoboDopamine uses step-wise reward discretization and multi-view fusion (Liang et al., 2026; Tan et al., 2025). More closely related, SARM and SARM2 learn stage-aware rewards, but do not report a dedicated evaluation of reward transfer to held-out environments (Chen et al., 2025a; 2026a). PRIMO R1 elicits progress reasoning through GRPO, but does not report downstream reward-guided robot policy optimization (Liu et al., 2026). R^2VLM uses recurrent reasoning for progress estimation on ALFRED and Ego4D (Zhang et al., 2026c), while DenseReward is trained on a predominantly simulated corpus (Fang et al., 2026). Complementary approaches use retrieval-based label correction, retry preferences, or transition alignment (Zhao et al., 2026; Qin et al., 2026; Zhang et al., 2026a). Together, these advances leave large-scale dense procedural reasoning supervision for transferable reward estimation underexplored; we study this direction using 400K trajectories spanning 30 real-robot and simulated datasets.

3 Methods

3.1 Procedural Supervision Synthesis

Large-scale embodied manipulation datasets contain diverse task executions, but their supervision is usually limited to task instructions, action trajectories, or coarse episode-level outcomes (Collaboration et al., 2023; Khazatsky et al., 2024; Walke et al., 2023). To learn procedure-grounded progress rewards, we develop a scalable procedural supervision synthesis pipeline that uses large vision-language models as automatic annotators (Team, 2025c; Wang et al., 2025b) to convert raw trajectories into frame-wise annotations of subtask stages, completion states, and remaining actions, forming *ProcCorpus-60M*.

3.1.1 Hierarchical Annotation Pipeline

Figure 2 illustrates our asynchronous annotation pipeline, which decouples data reading, multimodal preprocessing, VLM inference, and post-processing into separate workers connected by queues. Annotator and runtime details are provided in Appendix A.

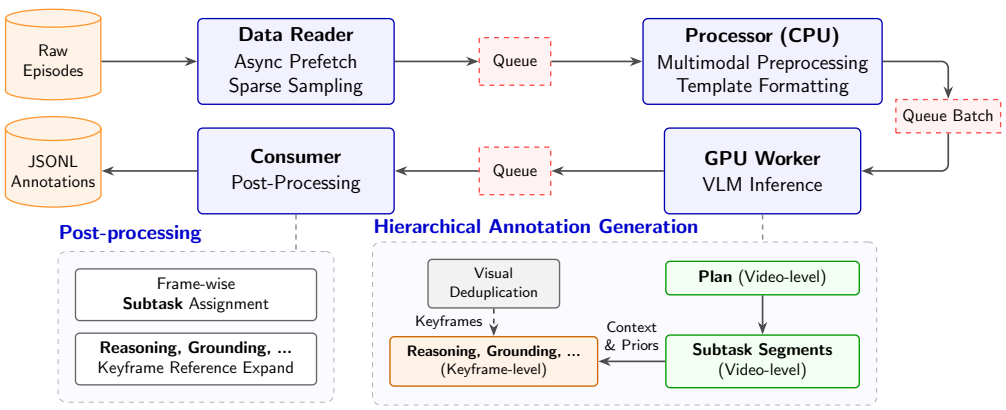


Figure 2: Procedural supervision synthesis pipeline. Asynchronous data reading, multimodal preprocessing, hierarchical VLM annotation, and post-processing convert robot episodes into frame-wise subtask labels and procedural reasoning.

The core annotation process follows a hierarchical, global-to-local design. For each episode, we first query the VLM on the complete video to infer a high-level task plan and localize the temporal span of each candidate subtask. These video-level segments define a shared subtask structure and are expanded into frame-wise assignments. Using the resulting task state and remaining subtasks as context, we generate concise visual reasoning on visually deduplicated keyframes. During post-processing, visually redundant neighboring frames reuse nearby keyframe reasoning, which preserves dense supervision with fewer annotation queries.

3.1.2 Data Scaling and Filtering

We apply the pipeline to real-robot datasets including DROID, BridgeData V2, Fractal, RH20T, Table30, and selected subsets of OXE (Khazatsky et al., 2024; Walke et al., 2023; Brohan et al., 2023; Fang et al., 2024a; Yakefu et al., 2025; Collaboration et al., 2023), and simulation datasets including LIBERO, RoboTwin 2.0, and GR00T-Teleop-Sim (Liu et al., 2023; Chen et al., 2025b; Bjorck et al., 2025), after screening for visual quality and instruction clarity (Appendix B). The resulting ProcCorpus-60M contains about 400K trajectories and over 60M annotated frames. We fully annotate RoboTwin but exclude it from pretraining and refinement to ensure reliable evaluation.

Each annotated sample contains *(1) task-centric scene reasoning*, including a concise description of task-relevant visual evidence and a completion-state assignment; *(2) subtask annotations*, including the current subtask and the trajectory-level subtask structure; and *(3) optional target grounding annotations* as 2D bounding boxes. Together, they link visual observations to progress, remaining steps, and manipulation targets. For refinement, we manually select about 15K trajectories whose subtask segmentation closely matches the original videos, reducing annotation noise while preserving task diversity, to further improve the model's subtask alignment beyond first-stage pretraining.

Shared trajectory-level subtask structure anchors local annotations, supporting consistent supervision for procedure-grounded progress reward learning. Asynchronous preprocessing and input buffering overlap CPU-side preparation with GPU-side inference, mitigating preprocessing-induced GPU idle time that can arise in native vLLM serving (Kwon et al., 2023) under image-heavy workloads, and supporting throughput of up to 4M keyframes per day on 8 H100 GPUs under our profiling setting.

3.2 Learning Procedure-Grounded Progress Rewards

Building on ProcCorpus-60M, we introduce *ProcVLM*, an embodied VLM for learning procedure-grounded progress rewards. Rather than assigning progress labels through linear time interpolation, ProcVLM learns from subtask-derived targets and conditions progress estimation on explicit reasoning about the current execution stage and remaining steps. Since ProcCorpus-60M provides structured annotations rather than ready-to-use training samples, we convert them into procedure-aware VQA tasks and jointly train ProcVLM for textual subtask reasoning and continuous progress estimation.

3.2.1 PROCEDURE-DEFINED PROGRESS TARGETS

To derive continuous progress labels from subtask annotations, we combine subtask-level structure with intra-subtask visual motion. Let T denote the trajectory length, K the number of valid subtasks, $k(t)$ the subtask index at time t, and $[s_k, e_k]$ the temporal span of the k-th subtask. We define progress $p(t)$ as the normalized accumulation of local visual change weighted by subtask duration:

$$p(t) = \frac{\sum_0^t w(\tau) r(\tau)}{\sum_0^T w(\tau) r(\tau)}, w(\tau) = \text{clip}\left(\frac{K\left(e_{k(\tau)} - s_{k(\tau)}\right)}{T}, 0.75, 1.25\right), r(\tau) = \frac{\|\dot{\phi}(\tau)\|}{\sum_{s_{k(\tau)}}^{e_{k(\tau)}} \|\dot{\phi}(u)\|}. \quad (1)$$

Here $\phi(\cdot)$ denotes the visual representation of the observation. The weight $w(\tau)$ assigns each subtask a global progress budget according to its relative duration, while the clipping range serves as a soft anchor around an equal-subtask prior, preventing unusually long or short subtasks from dominating or vanishing. Since $r(\tau)$ is normalized within each subtask, it distributes this budget over time according to local visual changes. In implementation, we approximate $\|\dot{\phi}(\tau)\|$ with adjacent-frame perceptual differences and add a small numerical stabilizer for each frame, yielding a lightweight progress label that preserves subtask structure without reducing the target to linear interpolation over time.

3.2.2 PROCEDURE-AWARE PRETRAINING

To turn the frame-wise annotations in ProcCorpus-60M into learnable supervision, we construct a procedure-aware VQA training set, termed *ProcVQA*, and formulate pretraining as multi-task VQA learning over robot observations and task instructions. Each training sample consists of a task instruction, one or more visual observations sampled from the trajectory, and a target response derived from the synthesized annotations. As illustrated in Figure 1, ProcVQA is centered on task progress estimation and further incorporates two auxiliary process-reasoning tasks to help the model understand task procedures beyond scalar progress prediction. Additional details of ProcVQA construction are provided in Appendix C.1.

Subtask-structured progress prediction. For progress prediction, we supervise the model to output the remaining atomic subtasks before estimating task completion. The final response ends with a continuous completion value wrapped by a structured progress tag: `<progress>p%</progress>`. This format grounds progress estimation in both visual evidence and explicit task structure, while also providing reasoning-formatted supervision for later adaptation.

Action segmentation. Given a task execution video and its task instruction, the model predicts a sequence of atomic subtask segments with semantic labels and temporal boundaries. This auxiliary task supervises the model to identify stage transitions in long-horizon manipulation and builds temporal understanding of task procedures.

Future planning. Given recent observations and task instruction, the model predicts the executable subtasks required in subsequent steps. This auxiliary task focuses on connecting the current manipulation state with future procedural actions, enabling the model to recover the remaining task structure needed for planning and failure recovery.

3.2.3 PROCVLM ARCHITECTURE AND TRAINING OBJECTIVES

ProcVLM is built on a compact vision-language backbone initialized from Qwen3-VL-2B-Instruct (Team, 2025c). The backbone takes task instructions and visual observations as input, and generates task-specific textual responses through the standard autoregressive language modeling head. This branch supports action segmentation, future planning, and reasoning-formatted progress prediction.

To enable continuous progress prediction, we attach a progress value head on top of the shared VLM representations. For progress-estimation samples, this branch is activated to regress a scalar completion score from contextual hidden states produced by the backbone. Specifically, the value head applies attention pooling over relevant multimodal and textual representations before predicting the progress value. This gated design preserves the original language generation pathway while adding a dedicated continuous regression route for progress estimation, helping mitigate the tendency of token-based numerical prediction to collapse into coarse or quantized anchors (Wang et al., 2025a; Ma et al., 2026; Golkar et al., 2023; Zhang et al., 2026b). By sharing the backbone with textual subtask reasoning, the value head grounds its prediction in structured visual-language representations rather than isolated numeric cues.

ProcVLM is trained with a joint objective that couples text generation with continuous progress regression. Across all three task families, we apply the standard autoregressive language modeling loss to the supervised textual response. This also includes progress-estimation samples: their reasoning context and formatted answer are learned through the language modeling objective, while the scalar progress value parsed from the target response's `<progress>` tag provides additional supervision for the value head.

For samples with progress supervision, the value branch predicts a continuous progress score $\hat{p}$, which is optimized against the ground-truth completion percentage p:

$$\mathcal{L}_{\text{value}} = \ell_{\text{reg}}(\hat{p}, p),$$

where ℓ_{reg} denotes the regression loss for continuous progress prediction. The final training objective is

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda \cdot \mathbb{I}_{\text{prog}} \mathcal{L}_{\text{value}}, \tag{2}$$

where $\mathcal{L}_{\text{LM}}$ is the standard autoregressive language modeling loss, $\mathbb{I}_{\text{prog}}$ indicates whether the sample contains progress supervision, and λ controls the weight of the value loss.

We train ProcVLM in two stages. The first stage uses the ProcVQA training split with approximately 20B tokens, while the second stage refines the model on a 2.8B-token curated set built from human-selected data to improve subtask alignment, procedural reasoning, and progress estimation. Further details on model configuration and training are provided in Appendices C.2 and C.3.

4 Experiments

We evaluate ProcVLM along three questions. *(Q1) Does ProcVLM improve embodied procedural understanding?* We evaluate this on ProcVQA, covering action segmentation, future planning, and task progress estimation. *(Q2) Can ProcVLM serve as a generalizable progress reward model?* We compare it with representative robotic reward models, test one-shot adaptation on RoboFAC, and conduct ablations on procedure-aware pretraining and reasoning-formatted supervision. *(Q3) Can ProcVLM improve downstream policy learning?* We use ProcVLM as a reward model for on-policy RL in RoboTwin simulation and reward-guided fine-tuning on a real robot.

4.1 Embodied Procedural Understanding

Setup. We evaluate ProcVLM on ProcVQA, a human-selected subtask-based VQA benchmark for embodied procedural understanding. ProcVQA includes three tasks: action segmentation, future planning, and task progress estimation. The ID split is derived from training-domain datasets such as DROID, Bridge, Table30, and OXE, while the OOD split is constructed from unseen RoboTwin tasks. We compare ProcVLM with mainstream VLMs, including GPT-5.4, Gemini 3.1 Pro, Qwen3VL (235B-A22B), and Qwen3.5 (27B). We report BF1@5/mMAE for action segmentation (Ishikawa et al., 2021), human-evaluated Success for future planning, and VOC/EPR@50 for progress estimation. Metric definitions are provided in Appendix D.2, with EPR defined in Eq. (3).

Findings. Table 1 shows that ProcVLM achieves the strongest overall performance on both ID and OOD splits. It obtains the best BF1@5, Success, VOC, and EPR@50, showing consistent gains in action segmentation, future planning, and progress estimation. Although its mMAE is not the lowest, ProcVLM remains comparable on this auxiliary localization metric while clearly improving the primary segmentation metric.

4.2 Generalizable Progress Reward Evaluation

4.2.1 Zero-Shot Comparison with Robotic Reward Models

Baselines and setup. We compare ProcVLM with Robometer and RoboDopamine on ProcVQA-OOD under a zero-shot setting. *Robometer* is trained on RBM-1M, a reward-learning dataset with over one million trajectories, which is larger in trajectory scale than our 400K-trajectory ProcCorpus-60M. It combines frame-level progress/success and trajectory-preference supervision without explicit reasoning over current and remaining subtasks. *RoboDopamine* predicts relative progress hops from multi-view comparisons of initial, goal, before, and after states through step-wise reward

Table 1: Embodied procedural understanding evaluation on ProcVQA. We compare ProcVLM with mainstream VLMs on ID and OOD splits across action segmentation, future planning, and task progress estimation.

		Action Segmentation		Future Planning	Progress Estimation	
Dataset	Name	BF1@5 ↑	mMAE ↓	Success ↑	VOC ↑	EPR@50 ↑
ID	Qwen3VL	0.3945	0.6033	0.6034	0.4420	4.4645
	Qwen3.5	0.4296	0.7247	0.7931	0.5475	4.8741
	Gemini 3.1 Pro	0.4698	0.6578	0.7413	0.5087	5.3725
	GPT-5.4	0.5221	**0.4864**	0.5517	0.5229	4.8058
	ProcVLM (Ours)	**0.6924**	0.6758	**0.8103**	**0.8058**	**5.7489**
OOD	Qwen3VL	0.4330	0.6263	0.6896	0.5361	4.9178
	Qwen3.5	0.4185	**0.5131**	0.7758	0.6397	5.0932
	Gemini 3.1 Pro	0.4584	0.6015	0.7241	0.5987	6.0399
	GPT-5.4	0.4941	0.5158	0.5862	0.6553	5.2751
	ProcVLM (Ours)	**0.5802**	0.5579	**0.8448**	**0.7282**	**6.1152**

Table 2: Zero-shot comparison with robotic reward models on the ProcVQA-OOD progress estimation subset.

Dataset	Metric	ProcVLM-2B	Robometer-4B	RoboDopamine-4B
ProcVQA-OOD	VOC ↑	**0.7282**	0.5296	<u>0.7156</u>
	EPR@50 ↑	<u>6.1152</u>	**6.7212**	4.5693

discretization and multi-perspective fusion. Our primary evaluation uses shuffled local-window queries, with short frame windows evaluated independently without chronological ordering. For RoboDopamine, we prepend a blank image as the before-state anchor; we acknowledge that this deviates from its intended input format and may limit its performance. EPR@50 is an auxiliary measure of progress-score resolution; it does not assess progress ordering, which is measured by VOC. Input construction and progress-score extraction are detailed in Appendix D.3.

Findings. Table 2 shows that ProcVLM achieves the best VOC under the shuffled local-window setting on the OOD split of ProcVQA. Robometer attains higher EPR@50, likely due to its carefully designed progress-regression head, but its lower VOC indicates weaker trajectory-internal progress ordering under zero-shot evaluation. RoboDopamine remains relatively stable with a blank contrastive anchor and shuffled VQA-window evaluation, but still lags behind ProcVLM in VOC. Its lower EPR@50 is consistent with the difficulty of representing continuous progress through discrete language tokens, which can favor coarse value anchors (Golkar et al., 2023; Wang et al., 2025a; Zausinger et al., 2025).

4.2.2 One-Shot Generalization

Setup and metrics. We evaluate one-shot reward-model generalization on successful and failed executions from the real-robot subset of RoboFAC, a failure-centric robotic VQA benchmark (Lu et al., 2025b). The 1-shot (Succ.) setting provides one successful demonstration per task, while 1-shot (Succ. + Fail.) additionally includes one demonstration for each task-specific failure type. Since one-shot demonstrations do not cover all camera views, the test set may contain unseen viewpoints, making the setting closer to real deployment. We report $\mathrm{VOC}_{\mathrm{succ}}$ for temporal ordering within successful trajectories, $\mathrm{MAE}_{\mathrm{fail}}$ (Eq. (4)) for fault localization on failed trajectories, MCC for success detection, and full-trajectory inference latency. Details are in Appendix E.

Fast one-shot adaptation. While zero-shot ProcVLM does not achieve the best progress ordering, adding a single successful trajectory per task improves $\mathrm{VOC}_{\mathrm{succ}}$ by 85.7%, increases MCC by 18.8%, and reduces $\mathrm{MAE}_{\mathrm{fail}}$ by 38.0% (Table 3). Although the $\mathrm{VOC}_{\mathrm{succ}}$ gain may partly reflect adaptation to the successful demonstration, the simultaneous improvements in $\mathrm{MAE}_{\mathrm{fail}}$ and MCC indicate that ProcVLM's adaptation extends beyond progress ordering on the demonstrated success case. In contrast, Robometer shows little improvement under the same one-shot setting, suggesting that preference supervision is less effective for few-demonstration adaptation. ProcVLM also maintains

Table 3: One-shot generalization on the real-robot subset of RoboFAC. We compare zero-shot and one-shot settings, where 1-shot (Succ.) provides one successful demonstration per task and 1-shot (Succ. + Fail.) additionally provides one demonstration for each task-specific failure type.

Setting	Name	$\text{VOC}_{\text{succ}} \uparrow$	$\text{MAE}_{\text{fail}} \downarrow$	MCC ↑	Latency (s) ↓
Zero-shot	Qwen3.5-27B	0.6191	0.1267	0.5678	1188
	RoboDopamine-8B	0.9095	0.2268	0.5506	157
	Robometer-4B	0.4129	<u>0.1218</u>	0.4516	139
	ProcVLM-2B	0.4920	0.2001	0.6665	**50**
1-shot (Succ.)	Robometer-4B	0.4094	0.1222	0.4435	136
	ProcVLM-2B	<u>0.9137</u>	0.1241	<u>0.7918</u>	81
1-shot (Succ. + Fail.)	Robometer-4B	0.4040	0.1221	0.4488	144
	ProcVLM-2B	**0.9301**	**0.1187**	**0.8053**	<u>80</u>

substantially lower inference latency than larger reward-model baselines, benefiting from its more compact model size.

Robust success detection and fault localization. Zero-shot ProcVLM achieves the best MCC (0.6665), which rises to the best overall MCC of 0.8053 with successful and failed demonstrations. For fault localization, one successful demonstration brings MAE_{fail} close to Robometer despite its larger scale and more extensive reward pretraining; adding failure demonstrations further improves MAE_{fail} to 0.1187.

4.2.3 Ablation Studies

Ablation setup. We conduct the ablation study under the RoboFAC 1-shot (Succ.) setting described in Section 4.2.2. Variants without ProcVLM pretraining are initialized from Qwen3-VL-2B-Instruct, the same backbone used before our procedure-aware pretraining.

Table 4: Ablation study on RoboFAC 1-shot adaptation. *PT* denotes ProcVLM pretraining and *Rsn* denotes reasoning-formatted supervision. Δ reports relative performance changes from ProcVLM, where negative values indicate degradation after accounting for metric direction.

Variant	PT	Rsn.	$\text{VOC}_{\text{succ}} \uparrow$		$\text{MAE}_{\text{fail}} \downarrow$		MCC ↑	
			Score	Δ	Score	Δ	Score	Δ
ProcVLM	✓	✓	**0.9137**	–	**0.1241**	–	**0.7918**	–
w/o Reasoning	✓	–	0.9040	-1.1%	0.1284	-3.5%	0.7598	-4.0%
w/o Pretrain	–	✓	0.8281	-9.4%	0.2343	-88.8%	0.5949	-24.9%
Base	–	–	0.8795	-3.7%	0.1985	-60.0%	0.6854	-13.4%

Procedure-aware pretraining enables one-shot transfer. Removing ProcVLM pretraining causes the largest degradation, increasing MAE_{fail} by 88.8% and reducing MCC by 24.9% (Table 4). This indicates that large-scale procedure-aware pretraining provides transferable representations for task progress perception, enabling the model to adapt to unseen RoboFAC tasks from only one successful demonstration.

Subtask-structured reasoning improves progress alignment. Removing reasoning-formatted supervision degrades all metrics, though less than removing pretraining, suggesting that reasoning helps align low-level subtask cues with downstream progress labels, improving one-shot generalization.

The Base variant further supports this conclusion, which retains reasonable progress ordering on successful trajectories after one-shot adaptation but performs worse on unseen failure cases (MAE_{fail}) and success detection (MCC), highlighting the **complementary roles of pretraining and reasoning**. Appendix G provides an annotation quality audit, a value head ablation, and qualitative reward modeling cases.

4.3 Reward-guided Policy Optimization

On-policy RL on RoboTwin. We evaluate ProcVLM, Robometer, and RoboDopamine on two RoboTwin tasks: Place Container Plate and Move Can Pot. Following the recommended adaptation protocols, we manually stage-annotate two of the 50 demonstrations per task to generate approximately 300 training samples for ProcVLM and 2,000 for RoboDopamine. Robometer uses 50 demonstrations without manual annotations, with five held out for validation. We then fine-tune each reward model on its corresponding adaptation data using its recommended configuration. The adapted reward models provide dense action-chunk rewards for OpenVLA-OFT (Kim et al., 2025) training with GRPO (Shao et al., 2024) in RLinf (Yu et al., 2025). Within each task, all methods share the SFT initialization, policy training/evaluation settings, and 480 trajectories per update. We compare success rates against active time on both tasks. Adaptation and evaluation details are in Appendix F.1.

Real-robot RFT. We conduct real-robot experiments on a stack-bowls task with a JAKA robot. We perform reward fine-tuning (RFT) with Evo-RL (Contributors, 2026), using ProcVLM's progress predictions as dense rewards to derive binary advantage labels (positive or negative) for advantage-conditioned policy training. Using $\pi_{0.5}$ (Intelligence et al., 2025b) as the base policy, we compare standard supervised fine-tuning (SFT) with RFT using the same policy initialization, batch size, and training set of 50 teleoperated demonstrations. Further details are provided in Appendix F.2.

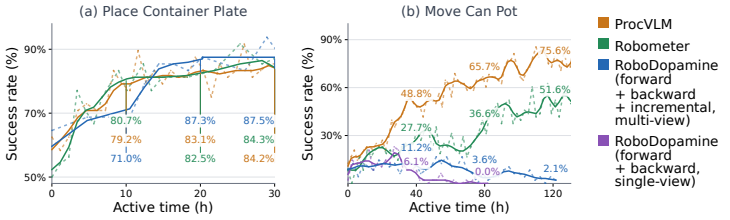


Figure 3: On-policy GRPO on RoboTwin.

Table 5: Soft success on the real-robot Stack Bowls task (%, ↑). Δ: gain over SFT in percentage points.

Steps	SFT	RFT	Δ
5k	37.5	62.5	+25.0
10k	70.8	83.3	+12.5

Findings. *On the RoboTwin benchmark,* ProcVLM and Robometer achieve comparable success rates on Place Container Plate, where SFT success is relatively high, while RoboDopamine attains slightly higher success near convergence. On the more challenging Move Can Pot task, where SFT success is low, ProcVLM achieves the highest final success rate, followed by Robometer. Both RoboDopamine variants remain near the low SFT baseline, suggesting limited transfer to this task under the same settings for fine-tuning the reward model (Figure 3). These results further support the effectiveness of ProcVLM's few-shot adaptation for downstream policy optimization.

On the real robot, ProcVLM-guided RFT improves soft success over SFT by 25.0 and 12.5 percentage points at 5k and 10k steps, respectively (Table 5). The larger early-stage gain is expected, as SFT may overfit to noisy demonstration behaviors such as repeated grasp retries (Figure 7). ProcVLM-based advantage conditioning helps reduce the influence of these less useful behaviors during fine-tuning.

5 Conclusion

This work presents ProcVLM as a procedure-grounded reward model for robotic learning. By combining subtask-structured supervision with reasoning over remaining steps, ProcVLM learns to estimate progress from procedural context, enabling transfer to new tasks and environments. Results on reward modeling, RoboTwin on-policy RL, and real-robot fine-tuning support its use in downstream policy optimization.

Limitations and Future Work. ProcVLM learns progress from procedure-defined supervision, so its estimates can be affected by the quality of subtask decomposition and temporal boundary localization (Lea et al., 2017; Sener & Yao, 2018; Ji et al., 2025). ProcVLM uses a lightweight progress-regression head, leaving calibration and robustness improvements to future work through distributional, stronger regression, or comparison-based objectives (Bellemare et al., 2017; Liang et al., 2026; Tan et al., 2025). Future work will also broaden real-world evaluation across robot platforms, tasks, and environments, with more comprehensive comparisons against other reward models.

AI USE STATEMENT

We used large vision-language models to generate procedural annotations for ProcCorpus-60M and ProcVQA, following the pipeline in Section 3.1 and Appendix A. These annotations undergo automated post-processing, with manual inspection used to curate the refinement subset. We also used generative AI tools for literature retrieval, manuscript drafting, and language polishing. The authors reviewed the AI-assisted text and references and take responsibility for the scientific claims, experimental results, and final manuscript.

ETHICS STATEMENT

ProcVLM is intended for research on reward modeling and robotic manipulation. It may reduce manual reward design, but inaccurate progress estimates can misguide policy learning and lead to unsafe robot behavior; real-world use therefore requires task-specific validation, safety checks, and human oversight. We credit the original sources of existing datasets, models, and software, respect their applicable licenses and terms of use, and do not redistribute raw third-party datasets. Data curation and human evaluation were conducted internally by project members, without crowdsourcing or research on human participants.

REPRODUCIBILITY STATEMENT

Section 3.2 describes the model, progress targets, and training objectives. Appendices A–C document annotation models and prompts, dataset construction, training configurations, and hyperparameters. Evaluation splits, baseline input protocols, and metrics are detailed in Appendices D and E. Appendix F specifies reward-model adaptation, policy training, and evaluation protocols for RoboTwin on-policy RL and real-robot RFT. We will provide anonymized implementation code for review. The full derived datasets are not included in the anonymous submission due to their scale; their sources and construction procedures are documented in the appendices above. After the anonymous review period ends, we will publicly release all redistributable portions of ProcCorpus-60M and ProcVQA, in accordance with the source datasets' licenses and terms of use, to support reproducibility and further research.

REFERENCES

Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5048–5058, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/453fadbd8a1a3af50a9df4df899537b5-Abstract.html.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 449–458. PMLR, 2017. URL http://proceedings.mlr.press/v70/bellemare17a.html.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith LLontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: an open foundation model for generalist humanoid robots. *CoRR*, abs/2503.14734, 2025. doi: 10.48550/ARXIV.2503.14734. URL https://doi.org/10.48550/arXiv.2503.14734.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey

Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *CoRR*, abs/2410.24164, 2024. doi: 10.48550/ARXIV.2410.24164. URL https://doi.org/10.48550/arXiv.2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong T. Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: robotics transformer for real-world control at scale. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu (eds.), *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi: 10.15607/RSS.2023.XIX.025. URL https://doi.org/10.15607/RSS.2023.XIX.025.

Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Fred Shentu, and Philipp Wu. SARM: stage-aware reward modeling for long horizon robot manipulation. *CoRR*, abs/2509.25358, 2025a. doi: 10.48550/ARXIV.2509.25358. URL https://doi.org/10.48550/arXiv.2509.25358.

Qianzhong Chen, Hau Zheng, Justin Yu, Suning Huang, Jiankai Sun, Ken Goldberg, Chuan Wen, Pieter Abbeel, Yide Shentu, Philipp Wu, and Mac Schwager. SARM2: multi-task stage aware reward modeling for self improving robotic manipulation. *CoRR*, abs/2606.10305, 2026a. doi: 10.48550/ARXIV.2606.10305. URL https://doi.org/10.48550/arXiv.2606.10305.

Shirui Chen, Cole Harrison, Ying-Chun Lee, Angela Jin Yang, Zhongzheng Ren, Lillian J. Ratliff, Jiafei Duan, Dieter Fox, and Ranjay Krishna. Topreward: Token probabilities as hidden zero-shot rewards for robotics. *CoRR*, abs/2602.19313, 2026b. doi: 10.48550/ARXIV.2602.19313. URL https://doi.org/10.48550/arXiv.2602.19313.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Qiwei Liang, Zixuan Li, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *CoRR*, abs/2506.18088, 2025b. doi: 10.48550/ARXIV.2506.18088. URL https://doi.org/10.48550/arXiv.2506.18088.

Open X-Embodiment Collaboration, Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alexander Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, Antonin Raffin, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Brian Ichter, Cewu Lu, Charles Xu, Chelsea Finn, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Chuer Pan, Chuyuan Fu, Coline Devin, Danny Driess, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Federico Ceola, Fei Xia, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Giulio Schiavi, Gregory Kahn, Hao Su, Haoshu Fang, Haochen Shi, Heni Ben Amor, Henrik I. Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Igor Mordatch, Ilija Radosavovic, and et al. Open x-embodiment: Robotic learning datasets and RT-X models. *CoRR*, abs/2310.08864, 2023. doi: 10.48550/ARXIV.2310.08864. URL https://doi.org/10.48550/arXiv.2310.08864.

Evo-RL Contributors. Evo-rl: Towards iterative policy improvement in real-world offline rl. https://github.com/MINT-SJTU/Evo-RL, 2026.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara

Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 8469–8488. PMLR, 2023. URL https://proceedings.mlr.press/v202/driess23a.html.

Haoshu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. RH20T: A comprehensive robotic dataset for learning diverse skills in one-shot. In *IEEE International Conference on Robotics and Automation, ICRA 2024, Yokohama, Japan, May 13-17, 2024*, pp. 653–660. IEEE, 2024a. doi: 10.1109/ICRA57147.2024.10611615. URL https://doi.org/10.1109/ICRA57147.2024.10611615.

Kuan Fang, Fangchen Liu, Pieter Abbeel, and Sergey Levine. MOKA: open-world robotic manipulation through mark-based visual prompting. In Dana Kulic, Gentiane Venture, Kostas E. Bekris, and Enrique Coronado (eds.), *Robotics: Science and Systems XX, Delft, The Netherlands, July 15-19, 2024*, 2024b. doi: 10.15607/RSS.2024.XX.062. URL https://doi.org/10.15607/RSS.2024.XX.062.

Yu Fang, Wanxi Dong, Jiaqi Liu, Yue Yang, Mingxiao Huo, Yao Mu, Huaxiu Yao, Li Erran Li, Daniel Szafir, and Mingyu Ding. Densereward: Dense reward learning via failure synthesis for robotic manipulation. *CoRR*, abs/2607.13033, 2026. doi: 10.48550/ARXIV.2607.13033. URL https://doi.org/10.48550/arXiv.2607.13033.

Senyu Fei, Siyin Wang, Li Ji, Ao Li, Shiduo Zhang, Liming Liu, Jinlong Hou, Jingjing Gong, Xianzhong Zhao, and Xipeng Qiu. SRPO: self-referential policy optimization for vision-language-action models. *CoRR*, abs/2511.15605, 2025. doi: 10.48550/ARXIV.2511.15605. URL https://doi.org/10.48550/arXiv.2511.15605.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In Dana Kulic, Gentiane Venture, Kostas E. Bekris, and Enrique Coronado (eds.), *Robotics: Science and Systems XX, Delft, The Netherlands, July 15-19, 2024*, 2024. doi: 10.15607/RSS.2024.XX.090. URL https://doi.org/10.15607/RSS.2024.XX.090.

Siavash Golkar, Mariel Pettee, Michael Eickenberg, Alberto Bietti, Miles D. Cranmer, Géraud Krawezik, François Lanusse, Michael McCabe, Ruben Ohana, Liam Holden Parker, Bruno Régaldo-Saint Blancard, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. xval: A continuous number encoding for large language models. *CoRR*, abs/2310.02989, 2023. doi: 10.48550/ARXIV.2310.02989. URL https://doi.org/10.48550/arXiv.2310.02989.

Google DeepMind. Gemini robotics-er 1.5. Model card and technical report, 2025. URL https://deepmind.google/models/gemini-robotics/gemini-robotics-er/.

Dongchi Huang, Zhirui Fang, Tianle Zhang, Yihang Li, Lin Zhao, and Chunhe Xia. CO-RFT: efficient fine-tuning of vision-language-action models through chunked offline reinforcement learning. *CoRR*, abs/2508.02219, 2025. doi: 10.48550/ARXIV.2508.02219. URL https://doi.org/10.48550/arXiv.2508.02219.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In Karen Liu, Dana Kulic, and Jeffrey Ichnowski (eds.), *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782. PMLR, 2022. URL https://proceedings.mlr.press/v205/huang23c.html.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, volume 229 of *Proceedings of Machine Learning Research*, pp. 540–562. PMLR, 2023. URL https://proceedings.mlr.press/v229/huang23b.html.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4573–4602. PMLR, 2024. URL https://proceedings.mlr.press/v270/huang25g.html.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do as I can, not as I say: Grounding language in robotic affordances. In Karen Liu, Dana Kulic, and Jeffrey Ichnowski (eds.), *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 2022. URL https://proceedings.mlr.press/v205/ichter23a.html.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. ${\pi^*}_{0.6}$: a VLA that learns from experience. *CoRR*, abs/2511.14759, 2025a. doi: 10.48550/ARXIV.2511.14759. URL https://doi.org/10.48550/arXiv.2511.14759.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *CoRR*, abs/2504.16054, 2025b. doi: 10.48550/ARXIV.2504.16054. URL https://doi.org/10.48550/arXiv.2504.16054.

Yuchi Ishikawa, Seito Kasai, Yoshimitsu Aoki, and Hirokatsu Kataoka. Alleviating over-segmentation errors by detecting action boundaries. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2021, Waikoloa, HI, USA, January 3-8, 2021*, pp. 2321–2330. IEEE, 2021. doi: 10.1109/WACV48630.2021.00237. URL https://doi.org/10.1109/WACV48630.2021.00237.

Yuheng Ji, Huajie Tan, Jiayu Shi, Xiaoshuai Hao, Yuan Zhang, Hengyuan Zhang, Pengwei Wang, Mengdi Zhao, Yao Mu, Pengju An, Xinda Xue, Qinghang Su, Huaihai Lyu, Xiaolong Zheng, Jiaming Liu, Zhongyuan Wang, and Shanghang Zhang. Robobrain: A unified brain model for robotic manipulation from abstract to concrete. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 1724–1734. Computer Vision Foundation / IEEE, 2025. doi: 10.1109/CVPR52734.2025.00168. URL https://openaccess.thecvf.com/content/CVPR2025/html/Ji_RoboBrain_A_Unified_Brain_Model_for_Robotic_Manipulation_from_Abstract_CVPR_2025_paper.html.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. VIMA: general robot manipulation with multimodal prompts. *CoRR*, abs/2210.03094, 2022. doi: 10.48550/ARXIV.2210.03094. URL https://doi.org/10.48550/arXiv.2210.03094.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R. Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Donovon Jackson, Charlotte Le, Yunshuang Li, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J. Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. DROID: A large-scale in-the-wild robot manipulation dataset. In Dana Kulic, Gentiane Venture, Kostas E. Bekris, and Enrique Coronado (eds.), *Robotics: Science and Systems XX, Delft, The Netherlands, July 15-19, 2024*, 2024. doi: 10.15607/RSS.2024.XX.120. URL https://doi.org/10.15607/RSS.2024.XX.120.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Paul Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2024. URL https://proceedings.mlr.press/v270/kim25c.html.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *CoRR*, abs/2502.19645, 2025. doi: 10.48550/ARXIV.2502.19645. URL https://doi.org/10.48550/arXiv.2502.19645.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In Jason Flinn, Margo I. Seltzer, Peter Druschel, Antoine Kaufmann, and Jonathan Mace (eds.), *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pp. 611–626. ACM, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Colin Lea, Michael D. Flynn, René Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 1003–1012. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.113. URL https://doi.org/10.1109/CVPR.2017.113.

Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. Roboreward: General-purpose vision-language reward models for robotics. *CoRR*, abs/2601.00675, 2026. doi: 10.48550/ARXIV.2601.00675. URL https://doi.org/10.48550/arXiv.2601.00675.

Peiyan Li, Yixiang Chen, Hongtao Wu, Xiao Ma, Xiangnan Wu, Yan Huang, Liang Wang, Tao Kong, and Tieniu Tan. Bridgevla: Input-output alignment for efficient 3d manipulation learning with vision-language models. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/5c1a8aa04c1a2cf5013f28831870dafa-Abstract-Conference.html.

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S. Huang, Luke Zettlemoyer, Dieter Fox, Yu Xiang, Anqi Li, Andreea Bobu, Abhishek Gupta, Stephen Tu, Erdem Biyik, and Jesse Zhang. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. *CoRR*, abs/2603.02115, 2026. doi: 10.48550/ARXIV.2603.02115. URL https://doi.org/10.48550/arXiv.2603.02115.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pp. 9493–9500. IEEE, 2023. doi: 10.1109/ICRA48891.2023.10160591. URL https://doi.org/10.1109/ICRA48891.2023.10160591.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: benchmarking knowledge transfer for lifelong robot learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Yibin Liu, Yaxing Lyu, Daqi Gao, Zhixuan Liang, Weiliang Tang, Shilong Mu, Xiaokang Yang, and Yao Mu. From passive observer to active critic: Reinforcement learning elicits process reasoning for robotic manipulation. In Paolo Favaro, Zuzana Kukelova, Atsuto Maki, Anna Rohrbach, Konrad Schindler, and Federico Tombari (eds.), *Computer Vision - ECCV 2026 - 19th European Conference, Malmö, Sweden, September 8-12, 2026, Proceedings, Part XVI*, volume 17016 of *Lecture Notes in Computer Science*, pp. 572–589. Springer, 2026. doi: 10.1007/978-3-032-37383-0_33. URL https://doi.org/10.1007/978-3-032-37383-0_33.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Zi-wei Wang. VLA-RL: towards masterful and general robotic manipulation with scalable reinforcement learning. *CoRR*, abs/2505.18719, 2025a. doi: 10.48550/ARXIV.2505.18719. URL https://doi.org/10.48550/arXiv.2505.18719.

Weifeng Lu, Minghao Ye, Zewei Ye, Ruihan Tao, Shuo Yang, and Bo Zhao. Robofac: A comprehensive framework for robotic failure analysis and correction. *CoRR*, abs/2505.12224, 2025b. doi: 10.48550/ARXIV.2505.12224. URL https://doi.org/10.48550/arXiv.2505.12224.

Xingyuan Ma, Shuai He, Anlong Ming, Haobin Zhong, and Huadong Ma. Regression over classification: Assessing image aesthetics via multimodal large language models. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 7827–7835. AAAI Press, 2026. doi: 10.1609/AAAI.V40I10.37726. URL https://doi.org/10.1609/aaai.v40i10.37726.

Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, Osbert Bastani, Dinesh Jayaraman, Wenhao Yu, Tingnan Zhang, Dorsa Sadigh, and Fei Xia. Vision language models are in-context value learners. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=friHAl5ofG.

Xinyao Qin, Junjie Lu, Kaixin Wang, Chuheng Zhang, Sinjae Kang, Kimin Lee, Min Xu, Bin Liang, Jun Yang, and Li Zhao. Beyond monotonic progress: Retry-supervised value learning for robot imitation. *CoRR*, abs/2606.24633, 2026. doi: 10.48550/ARXIV.2606.24633. URL https://doi.org/10.48550/arXiv.2606.24633.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. Spatialvla: Exploring spatial representations for visual-language-action model. *CoRR*, abs/2501.15830, 2025. doi: 10.48550/ARXIV.2501.15830. URL https://doi.org/10.48550/arXiv.2501.15830.

Scott E. Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *Trans. Mach. Learn. Res.*, 2022, 2022. URL https://openreview.net/forum?id=1ikK0kHjvj.

Fadime Sener and Angela Yao. Unsupervised learning and segmentation of complex activities from video. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 8368–8376. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00873. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Sener_Unsupervised_Learning_and_CVPR_2018_paper.html.

Pierre Sermanet, Tianli Ding, Jeffrey Zhao, Fei Xia, Debidatta Dwibedi, Keerthana Gopalakrishnan, Christine Chan, Gabriel Dulac-Arnold, Sharath Maddineni, Nikhil J. Joshi, Pete Florence, Wei Han, Robert Baruch, Yao Lu, Suvir Mirchandani, Peng Xu, Pannag Sanketi, Karol Hausman, Izhak Shafran, Brian Ichter, and Yuan Cao. Robovqa: Multimodal long-horizon reasoning for robotics. In *IEEE International Conference on Robotics and Automation, ICRA 2024, Yokohama, Japan, May 13-17, 2024*, pp. 645–652. IEEE, 2024. doi: 10.1109/ICRA57147.2024.10610216. URL https://doi.org/10.1109/ICRA57147.2024.10610216.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pp. 11523–11530. IEEE, 2023. doi: 10.1109/ICRA48891.2023.10161317. URL https://doi.org/10.1109/ICRA48891.2023.10161317.

Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Yuheng Ji, Cheng Chi, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, Shaoxuan Xie, Guocai Yao, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Robo-dopamine: General process reward modeling for high-precision robotic manipulation. *CoRR*, abs/2512.23703, 2025. doi: 10.48550/ARXIV.2512.23703. URL https://doi.org/10.48550/arXiv.2512.23703.

Gemini Robotics Team. Gemini robotics: Bringing AI into the physical world. *CoRR*, abs/2503.20020, 2025a. doi: 10.48550/ARXIV.2503.20020. URL https://doi.org/10.48550/arXiv.2503.20020.

Gemini Robotics Team. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *CoRR*, abs/2510.03342, 2025b. doi: 10.48550/ARXIV.2510.03342. URL https://doi.org/10.48550/arXiv.2510.03342.

Qwen Team. Qwen3-vl technical report. *CoRR*, abs/2511.21631, 2025c. doi: 10.48550/ARXIV.2511.21631. URL https://doi.org/10.48550/arXiv.2511.21631.

Homer Rich Walke, Kevin Black, Tony Z. Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, Abraham Lee, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata V2: A dataset for robot learning at scale. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1723–1736. PMLR, 2023. URL https://proceedings.mlr.press/v229/walke23a.html.

Pei Wang, Zhaowei Cai, Hao Yang, Davide Modolo, and Ashwin Swaminathan. Enhancing numerical prediction of MLLMS with soft labeling. In *IEEE/CVF International Conference on Computer Vision, ICCV 2025, Honolulu, HI, USA, October 19-25, 2025*, pp. 3424–3434. IEEE, 2025a. doi: 10.1109/ICCV51701.2025.00327. URL https://doi.org/10.1109/ICCV51701.2025.00327.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, Guanzhou Chen, Zichen Ding, Changyao Tian, Zhenyu Wu, JingJing Xie, Zehao Li, Bowen Yang, Yuchen Duan, Xuehui Wang, Zhi Hou, Haoran Hao, Tianyi Zhang, Songze Li, Xiangyu Zhao, Haodong Duan, Nianchen Deng, Bin Fu, Yinan He, Yi Wang, Conghui He, Botian Shi, Junjun He, Yingtong Xiong, Han Lv, Lijun Wu, Wenqi Shao, Kaipeng Zhang, Huipeng Deng, Biqing Qi, Jiaye Ge, Qipeng Guo, Wenwei Zhang, Songyang Zhang, Maosong Cao, Junyao Lin, Kexian Tang, Jianfei Gao, Haian Huang, Yuzhe Gu, Chengqi Lyu, Huanze Tang, Rui Wang, Haijun Lv, Wanli Ouyang, Limin Wang, Min Dou, Xizhou Zhu, Tong Lu, Dahua Lin, Jifeng Dai, Weijie Su, Bowen Zhou, Kai Chen, Yu Qiao, Wenhai Wang, and Gen Luo. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *CoRR*, abs/2508.18265, 2025b. doi: 10.48550/ARXIV.2508.18265. URL https://doi.org/10.48550/arXiv.2508.18265.

Kun Wu, Chengkai Hou, Jiaming Liu, Zhengping Che, Xiaozhu Ju, Zhuqin Yang, Meng Li, Yinuo Zhao, Zhiyuan Xu, Guang Yang, Zhen Zhao, Guangyu Li, Zhao Jin, Lecheng Wang, Jilei Mao, Xinhua Wang, Shichao Fan, Ning Liu, Pei Ren, Qiang Zhang, Yaoxu Lyu, Mengzhen Liu, Jingyang He, Yulin Luo, Zeyu Gao, Chenxuan Li, Chenyang Gu, Yankai Fu, Di Wu, Xingyu Wang, Sixiang Chen, Zhenyu Wang, Pengju An, Siyuan Qian, Shanghang Zhang, and Jian Tang. Robomind: Benchmark on multi-embodiment intelligence normative data for robot manipulation. *CoRR*, abs/2412.13877, 2024. doi: 10.48550/ARXIV.2412.13877. URL https://doi.org/10.48550/arXiv.2412.13877.

Yanru Wu, Weiduo Yuan, Ang Qi, Vitor Guizilini, Jiageng Mao, and Yue Wang. Large reward models: Generalizable online robot reward generation with vision-language models. *CoRR*, abs/2603.16065, 2026. doi: 10.48550/ARXIV.2603.16065. URL https://doi.org/10.48550/arXiv.2603.16065.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Zhengyi Luo, Yuqi Xie, Fengyuan Hu, Linxi "Jim" Fan, Guanya Shi, and Yuke Zhu. Self-improving vision-language-action models with data generation via residual RL. In Carl Vondrick, Bharath Hariharan, Colin Raffel, Lerrel Pinto, Diyi Yang, and Aleksandra Faust (eds.), *The Fourteenth International Conference on Learning Representations, ICLR 2026, Rio de Janeiro, Brazil, April 23-27, 2026*. proceedings.iclr.cc, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/163a875ab0c2631ed30b7435c4c9b8f1-Abstract-Conference.html.

Adina Yakefu, Bin Xie, Chongyang Xu, Enwen Zhang, Erjin Zhou, Fan Jia, Haitao Yang, Haoqiang Fan, Haowei Zhang, Hongyang Peng, Jing Tan, Junwen Huang, Kai Liu, Kaixin Liu, Kefan Gu, Qinglun Zhang, Ruitao Zhang, Saike Huang, Shen Cheng, Shuaicheng Liu, Tiancai Wang, Tiezhen Wang, Wei Sun, Wenbin Tang, Yajun Wei, Yang Chen, Youqiang Gui, Yucheng Zhao, Yunchao Ma, Yunfei Wei, Yunhuan Yang, Yutong Guo, Ze Chen, Zhengyuan Du, Ziheng Zhang, Ziming Liu, and Ziwei Yan. Robochallenge: Large-scale real-robot evaluation of embodied policies. *CoRR*, abs/2510.17950, 2025. doi: 10.48550/ARXIV.2510.17950. URL https://doi.org/10.48550/arXiv.2510.17950.

Shuai Yang, Hao Li, Yilun Chen, Bin Wang, Yang Tian, Tai Wang, Hanqing Wang, Feng Zhao, Yiyi Liao, and Jiangmiao Pang. Instructvla: Vision-language-action instruction tuning from understanding to manipulation. *CoRR*, abs/2507.17520, 2025. doi: 10.48550/ARXIV.2507.17520. URL https://doi.org/10.48550/arXiv.2507.17520.

Silong Yong, Stephen Sheng, Carl Qi, Xiaojie Wang, Evan Sheehan, Anurag Shivaprasad, Yaqi Xie, Katia Sycara, and Yesh Dattatreya. Generalizable dense reward for long-horizon robotic tasks. *CoRR*, abs/2604.00055, 2026. doi: 10.48550/ARXIV.2604.00055. URL https://doi.org/10.48550/arXiv.2604.00055.

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, Zixiao Huang, Mingjie Wei, Yuqing Xie, Ke Yang, Bo Dai, Zhexuan Xu, Xiangyuan Wang, Xu Fu, Zhihao Liu, Kang Chen, Weilin Liu, Gang Liu, Boxun Li, Jianlei Yang, Zhi Yang, Guohao Dai, and Yu Wang. Rlinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation. *CoRR*, abs/2509.15965, 2025. doi: 10.48550/ARXIV.2509.15965. URL https://doi.org/10.48550/arXiv.2509.15965.

Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *CoRR*, abs/2406.10721, 2024. doi: 10.48550/ARXIV.2406.10721. URL https://doi.org/10.48550/arXiv.2406.10721.

Yifu Yuan, Haiqin Cui, Yaoting Huang, Yibin Chen, Fei Ni, Zibin Dong, Pengyi Li, Yan Zheng, Hongyao Tang, and Jianye Hao. Embodied-r1: Reinforced embodied reasoning for general robotic manipulation. In Carl Vondrick, Bharath Hariharan, Colin Raffel, Lerrel Pinto, Diyi Yang, and Aleksandra Faust (eds.), *The Fourteenth International Conference on Learning Representations, ICLR 2026, Rio de Janeiro, Brazil, April 23-27, 2026*. proceedings.iclr.cc, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0f7ad318868d228b980b13cf827e85f4-Abstract-Conference.html.

Jonas Zausinger, Lars Pennig, Anamarija Kozina, Sean Sdahl, Julian Sikora, Adrian Dendorfer, Timofey Kuznetsov, Mohamad Hagog, Nina Wiedemann, Kacper Chlodny, Vincent Limbach, Anna Ketteler, Thorben Prein, Vishwa Mohan Singh, Michael M. Danziger, and Jannis Born. Regress, don't guess: A regression-like loss on number tokens for language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/zausinger25a.html.

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. *CoRR*, abs/2509.15937, 2025. doi: 10.48550/ARXIV.2509.15937. URL https://doi.org/10.48550/arXiv.2509.15937.

Haoyu Zhang, Zecui Zeng, Bin Wang, Lusong Li, Liang Lin, and Long Cheng. Dream2reward: Transition-alignment reward models from positive demonstrations for robotic manipulation. *CoRR*, abs/2608.18787, 2026a. doi: 10.48550/ARXIV.2608.18787. URL https://doi.org/10.48550/arXiv.2608.18787.

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. *CoRR*, abs/2505.10911, 2025. doi: 10.48550/ARXIV.2505.10911. URL https://doi.org/10.48550/arXiv.2505.10911.

Jianshu Zhang, Chengxuan Qian, Haosen Sun, Haoran Lu, Dingcheng Wang, Letian Xue, and Han Liu. Progresslm: Towards progress reasoning in vision-language models. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 11243–11271. Association for Computational Linguistics, 2026b. doi: 10.18653/V1/2026.ACL-LONG.516. URL https://doi.org/10.18653/v1/2026.acl-long.516.

Yuelin Zhang, Sijie Cheng, Chen Li, Zongzhao Li, Yuxin Huang, Yang Liu, and Wenbing Huang. Recurrent reasoning with vision-language models for estimating long-horizon embodied task progress. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2026, Denver, CO, USA, June 3-7, 2026*, pp. 41150–41159. Computer Vision Foundation, 2026c. URL https://openaccess.thecvf.com/content/CVPR2026/html/Zhang_Recurrent_Reasoning_with_Vision-Language_Models_for_Estimating_Long-Horizon_Embodied_Task_CVPR_2026_paper.html.

Zijian Zhang, Kaiyuan Zheng, Zhaorun Chen, Joel Jang, Yi Li, Chaoqi Wang, Mingyu Ding, Dieter Fox, and Huaxiu Yao. GRAPE: generalizing robot policy via preference alignment. *CoRR*, abs/2411.19309, 2024. doi: 10.48550/ARXIV.2411.19309. URL https://doi.org/10.48550/arXiv.2411.19309.

Lirui Zhao, Modi Shi, Li Chen, Qi Liu, Ping Luo, and Hongyang Li. UR-VC: unsupervised robotic value correction for time-derived progress proxies. *CoRR*, abs/2607.12892, 2026. doi: 10.48550/ARXIV.2607.12892. URL https://doi.org/10.48550/arXiv.2607.12892.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong T. Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

APPENDIX CONTENTS

A Annotation Pipeline Details **21**
A.1 Annotator Models . . . 21
A.2 Pipeline Execution and Post-processing . . . 21
A.3 Pipeline Profiling . . . 21
A.4 Prompt Templates . . . 22

B Preprocessed Dataset Details **25**

C ProcVLM Training Details **26**
C.1 ProcVQA Construction . . . 26
C.1.1 Construction from Procedural Annotations . . . 26
C.1.2 Task Templates . . . 26
C.1.3 Training Sample Statistics . . . 27
C.2 ProcVLM Configuration . . . 28
C.2.1 Backbone and Output Branches . . . 28
C.2.2 Progress-Value Gating . . . 28
C.2.3 Value-Head Pooling . . . 28
C.3 Training Pipeline and Implementation Details . . . 29
C.3.1 Training Objectives . . . 29
C.3.2 Tail Dropout for Leakage Mitigation . . . 29
C.3.3 Two-stage Training Pipeline . . . 30

D ProcVQA Evaluation Details **31**
D.1 Benchmark and Splits . . . 31
D.2 Metrics . . . 31
D.3 Reward-Model Evaluation . . . 32

E RoboFAC Evaluation Details **32**
E.1 Implementation Details . . . 32
E.2 Metrics . . . 33

F Reward-guided Policy Optimization Details **33**
F.1 On-Policy Reinforcement Learning on RoboTwin . . . 34
F.2 Real-Robot Reward Fine-tuning . . . 34

G Additional Analyses **36**
G.1 Annotation Quality Audit . . . 36
G.2 Ablation of the Value Head . . . 37
G.3 Reward Modeling Cases . . . 37

A Annotation Pipeline Details

This section provides additional details of the annotator models, hierarchical annotation process, pipeline profiling results, and prompt templates used in Section 3.1.

A.1 Annotator Models

We use different large VLMs for different annotation stages. For video-level planning and subtask temporal localization, we use Qwen3-VL-235B-A22B-Instruct as the annotator model, since these tasks require long-context video understanding and global procedural reasoning. For frame-level reasoning and grounding annotations, we use InternVL3.5-38B, which provides efficient and reliable single-frame visual reasoning. All generated annotations are post-processed into JSONL files and used to construct procedure-aware VQA tasks for ProcVLM pretraining.

A.2 Pipeline Execution and Post-processing

The annotation pipeline is implemented as a queue-based asynchronous system to avoid serialized CPU/GPU execution. It consists of four decoupled modules: data reading, CPU-side preprocessing, GPU-side VLM inference, and post-processing. Each module runs independently and communicates through bounded queues, so that slow I/O, image preprocessing, and VLM inference do not block each other. This design overlaps CPU preparation with GPU execution, reduces pipeline bubbles, and keeps the annotator GPUs continuously supplied with ready-to-run batches.

Data reader. The data reader scans raw robot episodes, organizes them into task-consistent trajectories, and performs sparse video sampling according to the annotation stage. It also prefetches images and metadata asynchronously, including task instructions, camera keys, frame indices, and episode identifiers. This module isolates storage and decoding latency from the rest of the pipeline, preventing GPU workers from waiting on raw data loading.

CPU processor. The CPU processor converts sampled episodes into VLM-ready inputs. It performs image loading, resizing, multimodal preprocessing, prompt construction, and template formatting for different annotation tasks, including plan generation, subtask temporal localization, frame-level reasoning and target grounding. Because each annotation query often contains multiple images or video frames, we move image loading, multimodal preprocessing, and template formatting to CPU workers ahead of GPU inference, keeping ready-to-run batches available for GPU workers and reducing idle GPU time.

GPU worker. The GPU worker consumes preprocessed batches and runs large VLM inference. Depending on the annotation stage, it performs video-level task planning, subtask temporal localization, or keyframe-level reasoning and grounding. We use optimized inference backends such as vLLM and LMDeploy, together with local batched inference APIs, to keep large multimodal requests densely packed on the GPU and maximize annotation throughput.

Consumer and post-processing. The consumer parses VLM outputs, validates their format, and writes normalized JSONL annotations indexed by dataset name, episode id, frame id, and camera key. It expands video-level subtask segments into frame-wise assignments, propagates keyframe reasoning to neighboring non-keyframes when appropriate, checks temporal consistency for grounding box outputs and removes invalid boxes. This stage converts heterogeneous VLM outputs into a standardized annotation format, allowing different robot datasets to be merged into ProcCorpus and later converted into ProcVQA training samples.

Overall, the pipeline separates I/O-bound, CPU-bound, GPU-bound, and format-normalization workloads into independent stages. This modular design improves annotation throughput, reduces idle time caused by CPU/GPU synchronization, and makes large-scale frame-wise procedural annotation practical over hundreds of thousands of trajectories.

A.3 Pipeline Profiling

In our profiling experiment, the pipeline is deployed on 8 NVIDIA H100 GPUs with 80GB HBM3 memory. Under this setting, it processes about 4M keyframes per day. Figure 4 shows the throughput and GPU utilization of the annotation pipeline.

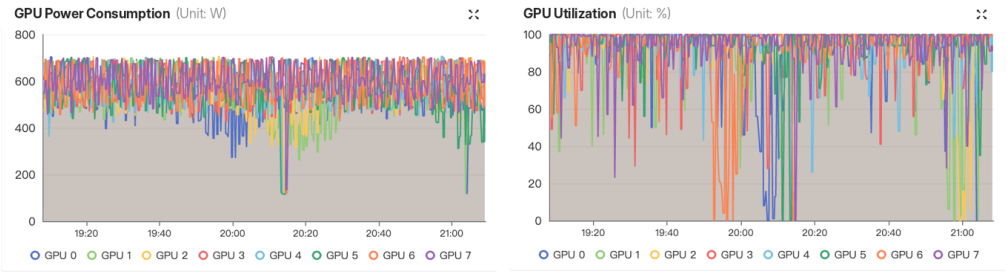


Figure 4: GPU power consumption and utilization over time across multiple GPUs during the annotation pipeline execution. The GPUs maintain consistently high utilization and stable power draw for most of the run, indicating efficient parallel workload scheduling with only brief transient drops.

A.4 Prompt Templates

We use task-specific prompt templates for plan generation, subtask temporal localization, and frame-level reasoning. The templates are shown below.

Plan Generation.

```
I will give you a robot task and a video showing the robot arm performing
the task.
You need to analyze actions of the robot arm from the video and decompose
the task
into a sequence of detailed sub-tasks. The sequence of sub-tasks should
lead to the
completion of the overall task.

###
Grasp [specific object]
e.g. "Grasp the red block"
Explain: Use this pattern when the robot isn't holding anything and needs
to pick up
an object, before performing other actions like placing or lifting.
###
Place [specific object] onto / into [specific location]
e.g. "Place the cup onto the table" or "Place the screwdriver into the
tool rack."
Explain: Use this pattern when the robot is holding an object and needs
to put it
down at a specific location.
###
Push [specific object] [forward / backward / left / right]
e.g., "Push the blue block forward"
Explain: Use this pattern when the robot needs to move an object in a
specific
direction by applying force to it, without lifting or grasping it. "Push"
and
("Grasp", "Place") are mutually exclusive.
###
Tilt the gripper
e.g. "Tilt the gripper to pour the liquid" or "Tilt the gripper slightly
to the left"
Explain: Use this pattern when the robot needs to adjust the angle of its
gripper,
either to pour liquid or position the gripper for some specific task.
###
Hang [specific object] on / above [specific location]
e.g. "Hang the coat on the hook" or "Hang the cup above the table"
Explain: Use this pattern when the robot needs to suspend an object from
a specific
```

```
location, such as hanging a coat on a hook or a cup above a table.
###
Press [specific object]
e.g., "Press the button" or "Press the power switch until it clicks."
###
Open [specific object]
e.g. "Open the door slowly"
###
Close [specific object]
e.g., "Close the lid securely"
###
Rotate [specific object]
e.g., "Rotate the knob clockwise"
###

All sub-tasks must be in exactly one of the patterns above, and should
follow the
Explain for each pattern. There is no need to include the robot arm
itself as an
object in the sub-tasks.

You should output sub-tasks in a numbered list format, starting from 1.
Each line
contains one sub-task with a leading number and a period. No extra text
or
explanation.

Task: {task}
Output:
```

Subtask Segmentation.

```
You will be shown a VIDEO of a robot task and an UNORDERED list of
planned sub-tasks.

Task: "{task}"
Planned sub-tasks:
{plans}

OBJECTIVE:
For each planned sub-task, if present, mark the frames where it starts
and finishes.
If a sub-task is started but not finished, set complete_frame=null. If
not present
at all, set both start_frame=null and complete_frame=null, and notes="not
present".
If the video shows any action was interrupted and the overall task was
not completed,
set overall_notes="task not completed".

OUTPUT FORMAT:
{
  "task": "<same as input task>",
  "subtasks": [
    {"id":1, "notes":"<<=60 words optional>",
     "start_frame":<int|null>, "complete_frame":<int|null>,
     "name":"<same text from plans>"},
    ...
  ],
  "overall_notes":"<<=30 words optional>"
}

HINTS:
- Find the changes in effector pose and object motion as candidate
start/complete frames.
```

```
- The start frame can be picked slightly earlier and the complete frame
slightly later
  to ensure the action is fully captured.
- If multiple candidates exist, pick the final success. If retries occur,
record the
  final success.
- Use the last frame of the video as reference for overall_notes.

Now process the provided video and planned sub-tasks and return the JSON
result ONLY.
```

Frame-level Reasoning. We use three reasoning templates depending on the task state: unfinished, finished, and give-up/failure. These templates are intentionally short and task-oriented, encouraging the annotator VLM to focus on task completion and remaining procedural steps rather than open-ended scene description.

Unfinished state:

```
The image shows a robot performing a task: '{task}', which may be
incomplete.
Remaining subtasks: '{rest_sub_task}'.
Explain why it's unfinished and briefly describe the next steps based on
image details.

Output (<=150 words, 3 sentences):
<analysis with image details>. This task is not finished <short reason>.
<one-sentence summary of next steps>.

Example:
Task: 'put all the green objects on the pink plate.'
Image: a green apple in robot arm, a green pear on blue plate.
Output:
Image shows a green apple held by the robot and a green pear on the blue
plate.
This task is not finished because both green objects are not yet on the
pink plate.
The robot should place the green apple on the pink plate, then move the
green pear
from the blue plate to the pink plate.
```

Finished state:

```
The image shows a robot performing a task: '{task}', which is finished.
Explain briefly why it's completed based on image details.

Output (<=50 words, 2 sentences):
<analysis with image details>. This task is finished <short reason>.

Example:
Task: 'put all the green objects on the pink plate.'
Image: both green apple and pear on pink plate.
Output:
Image shows a green apple and pear on the pink plate. This task is
finished because
all green objects are placed correctly.
```

Give-up or failed state:

```
The image shows a robot performing a task: '{task}', which is not
finished.
Explain briefly why it's unfinished based on image details.

Output (<=50 words, 2 sentences):
<analysis with image details>. This task is not finished <short reason>.

Example:
```

```
Task: 'put all the green objects on the pink plate.'
Image: a green apple held by the robot, a green pear on blue plate.
Output:
Image shows a green apple in the robot arm and a green pear on the blue
plate.
This task is not finished because neither object has been placed on the
pink plate.
```

Table 6: Datasets used for ProcCorpus construction. Annotated frames and annotation rates are computed based on subtask annotations only.

Dataset	Frames	Traj.	Ann. Frames	Ann. Rate
Real-robot datasets				
DROID	27,630,375	95,658	26,157,676	94.67%
BridgeData V2	2,863,587	78,637	2,790,172	97.44%
Fractal	3,786,400	87,212	3,778,827	99.80%
RH20T	1,699,138	4,433	1,621,815	95.45%
Table30	5,184,355	25,610	5,167,461	99.67%
OXE / Austin BUDS	34,112	50	34,112	100.00%
OXE / Austin SAILOR	353,094	240	353,094	100.00%
OXE / Austin SIRIUS	279,939	559	279,939	100.00%
OXE / Berkeley AUTOLab UR5	86,887	896	86,583	99.65%
OXE / Berkeley FANUC Manipulation	62,613	415	59,764	95.45%
OXE / CMU Play Fusion	235,922	576	235,096	99.65%
OXE / Columbia PushT	24,802	122	23,220	93.62%
OXE / DLR EDAN Shared Control	8,928	104	8,597	96.29%
OXE / DLR SARA Pour	12,971	100	12,771	98.46%
OXE / Dobb-E	1,139,911	5,208	1,087,931	95.44%
OXE / FMB	1,137,340	8,611	959,915	84.40%
OXE / IAMLab CMU Pickup-Insert	146,105	522	145,053	99.28%
OXE / Jaco Play	70,127	976	68,872	98.21%
OXE / NYU Door Opening	17,761	435	17,670	99.49%
OXE / QUT Dexterous Manipulation	176,278	200	175,538	99.58%
OXE / Stanford HYDRA	358,234	570	353,326	98.63%
OXE / TOTO	294,139	902	294,139	100.00%
OXE / UCSD Kitchen	3,970	150	3,491	87.93%
OXE / UT Austin Mutex	361,883	1,500	350,737	96.92%
Internal / JAKA Teleop	23,774	100	22,271	93.68%
Internal-xArm / Real-world Tabletop	366,152	2,005	366,152	100.00%
Real-robot total	**46,358,797**	**315,791**	**44,454,222**	**95.89%**
Simulation datasets				
GR00T-Teleop-Sim	5,820,277	24,000	5,807,497	99.78%
Internal-xArm / Simulation Tabletop	2,001,729	43,720	1,998,726	99.85%
LIBERO v2.1	273,465	1,693	273,465	100.00%
RoboTwin 2.0 (*held out from both training stages for OOD evaluation*)	6,012,086	27,200	5,997,357	99.76%
Simulation total	**14,107,557**	**96,613**	**14,077,045**	**99.78%**
Overall total	**60,466,354**	**412,404**	**58,531,267**	**96.80%**

B PREPROCESSED DATASET DETAILS

This section provides the detailed list of real-robot and simulation datasets preprocessed and annotated for ProcCorpus, as described in Section 3.1.2. We merge multiple task-level or configuration-level subsets belonging to the same dataset family, and report annotation statistics based only on subtask annotations.

Table 6 summarizes the dataset sources and subtask annotation coverage of ProcCorpus. In total, the corpus contains more than 60M raw frames from over 400K real-robot and simulated trajectories, with

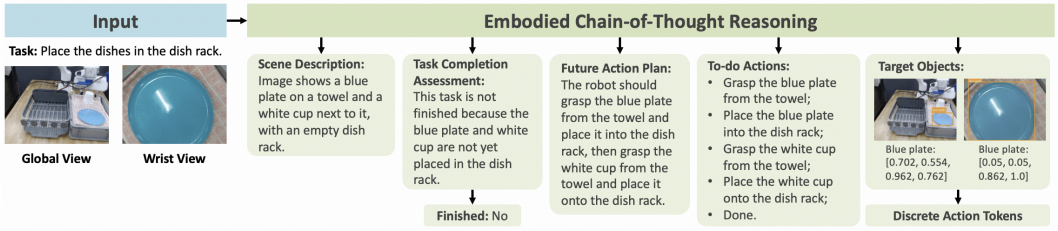


Figure 5: Example of *Embodied Chain-of-Thought (ECoT)* annotation in ProcCorpus. Given multi-view robot observations and a task instruction, ECoT enriches the raw frame with task-centric scene reasoning, completion assessment, future action planning, remaining to-do actions, target-object grounding, and optional discrete action tokens for VLA training.

most datasets achieving high subtask annotation coverage. These statistics show that the annotation pipeline can scale to heterogeneous manipulation data while maintaining dense procedure-level supervision for ProcVLM pretraining.

As illustrated in Figure 5, each ProcCorpus annotation enriches a raw manipulation frame with task-centric reasoning, completion-state assignment, subtask structure, and target grounding.

C ProcVLM Training Details

This section provides additional training details for ProcVLM, including the construction of ProcVQA, the model configuration for progress-value prediction, and the two-stage training pipeline, as described in Section 3.2.

C.1 ProcVQA Construction

C.1.1 Construction from Procedural Annotations

ProcVQA is constructed from the frame-wise annotations in ProcCorpus. Each raw trajectory is represented by a task instruction and a robot execution video, where the video is treated as an ordered image sequence. The synthesized annotations provide subtask labels, temporal boundaries, frame-level procedural reasoning, and progress estimates.

We convert these annotations into supervised multimodal question-answering samples. Each sample consists of a textual query, one or more visual observations sampled from the trajectory, and a target response derived from the corresponding procedural annotations. This conversion turns dense frame-wise annotations into a unified VQA-style supervision format for multimodal instruction tuning.

We define an **atomic action** as an executable subtask that can be described by a **single explicit verb-level** action. A high-level task may admit multiple valid decompositions under this definition, but we regard them as acceptable as long as each atomic action is temporally coherent and verb-maximal, i.e., it covers a complete continuous execution phase whenever possible. Under this principle, different decompositions can still provide meaningful low-level and generalizable action descriptions.

For instance, the high-level instruction “clean the table” is not directly executable as a single verb-level action. When two tissues are on the table, a valid decomposition may include grasping the left tissue, placing it into the trash can, grasping the right tissue, and placing it into the trash can. ProcVQA therefore supervises the model to reason over explicit procedural steps rather than only imitate trajectory-level task descriptions.

C.1.2 Task Templates

These templates instantiate the pretraining tasks introduced in Section 3.2.2, with five concrete variants denoted as (a.1), (a.2), (b.1), (b.2), and (c). They use either video-level interleaved frame-index sequences for action segmentation or recent local observation windows for next-step prediction, remaining-step planning, and progress estimation.

An interleaved frame-index sequence represents a video as an ordered multimodal sequence, where each frame image is followed by its frame identifier, e.g., `<image><frame_id: 1><image><frame_id: 2>`. This format preserves the temporal order of sampled frames while remaining compatible with the image-text input interface of the VLM.

(a.1) Action segmentation with task instruction. Given an interleaved frame-index sequence and its task instruction, the model segments the execution into consecutive atomic actions:

```
Segment the execution of the task "{task}" into consecutive
atomic actions.  Each segment should correspond to a single,
explicit verb-level action.  Output a JSON list with keys
"action_description", "start_frame", and "end_frame".
```

The target response is a JSON-style list of action segments, where each segment contains an action description and its start and end frames.

(a.2) Task-free action segmentation. We also construct a task-free segmentation variant, where the model receives only the visual sequence and predicts the atomic action segments without the original task instruction:

```
Segment the actions shown in the image sequence into
consecutive atomic actions, each described by a
single explicit verb.  Output a JSON list with keys
"action_description", "start_frame", and "end_frame".
```

This variant encourages the model to infer procedural structure directly from visual observations.

(b.1) Immediate next-step prediction. Given a recent observation window and the task instruction, the model predicts the immediate next atomic action:

```
Given the recent observation and the task "{task}", predict
the immediate next atomic action the robot should execute.
Use a single explicit verb-level description.
```

The target response is a single executable verb-level action.

(b.2) Future planning. Given a recent observation window and the task instruction, the model predicts the remaining atomic actions required to complete the task:

```
Given the recent observation and the task "{task}", list
the remaining atomic actions required to complete the task,
starting from the next time step.  Each action should be a
single explicit verb-level step.
```

The target response is an ordered list of remaining subtasks.

(c) Subtask-structured progress prediction. For progress prediction, the model first infers the remaining atomic actions and then estimates the current task completion percentage:

```
Given the recent observation and the task "{task}", first
infer the remaining atomic actions required to complete the
task.  Then estimate the current completion percentage and
output it as a float wrapped by <progress> tags.
```

The response ends with a structured progress tag:

$$\texttt{<progress>}\ p\ \texttt{\%</progress>},$$

where p is a continuous completion percentage. This format provides both token-level supervision for procedural reasoning and scalar supervision for the progress value head.

C.1.3 Training Sample Statistics

ProcVQA is split into a large-scale pretraining set and a curated refinement set. The first-stage set is constructed from the training split of the annotated corpus and provides broad coverage across

Table 7: Summary of ProcVQA training data used for the two-stage ProcVLM training pipeline.

Statistic	Stage 1 Pretraining	Stage 2 Refinement
Data source	ProcVQA training split	Human-selected subset
Token scale	∼20B	2.8B
Trajectory scale	∼400K	∼15K
Main purpose	Broad procedural pretraining	High-quality refinement

robots, environments, viewpoints, and task types. The second-stage set is built from human-selected trajectories with more accurate subtask alignment and clearer procedural structure.

To further characterize the procedural structure of the curated refinement set, we report the distribution of subtask counts per trajectory in Table 8. The Stage 2 set contains 13,688 trajectories, with an average of 2.87 subtasks per trajectory and a median of 3. This indicates that the curated set contains substantial multi-step procedural structure rather than only single-action executions, supporting its use for refinement-stage training on subtask alignment and progress reasoning.

Table 8: Distribution of subtask counts per trajectory in the Stage 2 curated refinement set.

# Subtasks	1	2	3	4	5	6	7	8	9	10	11	12	≥ 14
# Trajectories	1455	5303	4187	1523	210	479	123	197	19	153	2	15	22

Summary statistic	Value
Total trajectories	13,688
Mean # subtasks	2.87
Median # subtasks	3
Minimum # subtasks	1
Maximum # subtasks	24

C.2 ProcVLM Configuration

C.2.1 Backbone and Output Branches

ProcVLM is initialized from Qwen3-VL-2B-Instruct. The backbone takes visual observations and task instructions as multimodal input and generates textual responses through the standard autoregressive language modeling head. This language branch is used for all ProcVQA task families.

The value head is implemented as a three-layer MLP with hidden dimensions $d_h \to 4d_h \to d_h \to 1$, where d_h denotes the hidden size of the VLM backbone.

C.2.2 Progress-Value Gating

The progress value head is controlled by a semantic gating mechanism. During training, this branch is activated when the ground-truth response contains a `<progress>` tag. For samples without progress supervision, only the language modeling objective is applied. During inference, the value branch can be activated on demand for responses containing a `<progress>` tag. This gated design separates general procedure-aware text generation from continuous value prediction. It allows ProcVLM to learn from all task families through language modeling, while using progress-prediction samples to additionally supervise the value head.

C.2.3 Value-Head Pooling

Let $H = \{h_i\}_{i=1}^{L}$ denote the sequence hidden states produced by the VLM backbone. For progress-prediction samples, the value head applies attention pooling over the valid sequence positions:

$$\alpha_i = \frac{\exp(s(\tilde{h}_i)) \cdot m_i}{\sum_{j=1}^{L} \exp(s(\tilde{h}_j)) \cdot m_j},$$

$$h_{\text{pool}} = \sum_{i=1}^{L} \alpha_i \tilde{h}_i, \qquad \hat{p} = f_{\text{value}}(h_{\text{pool}}),$$

where $s(\cdot)$ is the attention scoring function, f_{value} is the regression head, and m_i denotes the attention mask that excludes padding tokens. During training, $\tilde{h}_i$ is obtained by applying feature-level dropout to the tail hidden states that contain the progress answer, while keeping other hidden states unchanged. This corruption reduces direct access to the ground-truth progress tokens during teacher forcing and encourages the value head to infer progress from the visual observation, task instruction, and prior procedural reasoning.

C.3 Training Pipeline and Implementation Details

C.3.1 Training Objectives

Language Modeling Loss. Given the ProcVQA templates above, all task families are trained with the standard autoregressive language modeling objective over the supervised assistant response. The input prompt, including task instructions and visual observations, is used as conditioning context, while the loss is applied only to the target response tokens. Formally, given a multimodal input x and a target response $y = \{y_t\}_{t=1}^{T}$, we compute

$$\mathcal{L}_{\text{LM}} = -\frac{1}{|\mathcal{M}_{\text{text}}|} \sum_{t \in \mathcal{M}_{\text{text}}} \log P_\theta(y_t \mid y_{<t}, x),$$

where $\mathcal{M}_{\text{text}}$ denotes the set of target response tokens used for language modeling supervision. Prompt tokens and padding tokens are masked out in the label tensor and are therefore excluded from the loss.

Progress-prediction samples are also included in this language modeling objective. In these samples, the reasoning context and the formatted progress answer are learned as text, while the scalar value inside the `<progress>` tag is additionally used as supervision for the value head.

Progress Regression Loss. For progress-prediction samples, we parse the scalar completion value from the structured `<progress>` tag and use it as the ground-truth progress label. Although the textual response expresses progress as a percentage, the regression loss is computed on a normalized scale. Let p denote the parsed completion percentage and $\bar{p} = p/100$ denote its normalized value. The value head predicts a normalized progress score $\hat{p} \in [0, 1]$, and the value loss is defined as

$$\mathcal{L}_{\text{value}} = \ell_{\text{reg}}(\hat{p}, \bar{p}),$$

where ℓ_{reg} is the regression loss used for continuous progress prediction.

Joint Objective. The overall objective is

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda \cdot \mathbb{I}_{\text{prog}} \mathcal{L}_{\text{value}},$$

where $\mathbb{I}_{\text{prog}}$ indicates whether the sample contains progress supervision and λ controls the contribution of the value regression loss. In minibatches containing both progress and non-progress samples, the value loss is averaged only over samples with valid progress labels.

C.3.2 Tail Dropout for Leakage Mitigation

During teacher forcing, the ground-truth progress value appears near the end of the textual target. If the value head directly relies on the hidden states of the `<progress>` answer tokens, it may recover the target value from the answer span rather than infer task progress from the visual observation and procedural context. To reduce this leakage, we apply a soft tail-masking strategy before value-head pooling.

Specifically, for each training sequence, we identify the last valid token according to the attention mask and apply feature-level dropout to the final N_{tail} hidden states:

$$\tilde{h}_i = \begin{cases} \text{Dropout}(h_i; p_{\text{tail}}), & i \in \mathcal{T}_{\text{tail}}, \\ h_i, & \text{otherwise}, \end{cases}$$

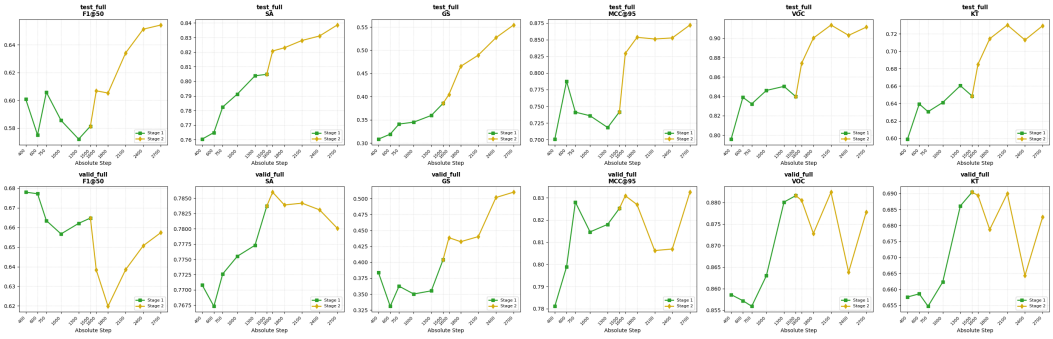


Figure 6: Training curves of ProcVLM under the two-stage training pipeline. Green denotes Stage 1 pretraining on the ProcVQA training split, and yellow denotes Stage 2 refinement on the curated subset. Metrics are evaluated on the in-distribution `test_full` split and the out-of-distribution `valid_full` split. Stage 1 mainly establishes generalizable procedural representations from large-scale diverse supervision, while Stage 2 further improves prediction precision with higher-quality refinement data.

where $\mathcal{T}_{\text{tail}}$ denotes the tail token region containing the progress answer. In our implementation, we set $N_{\text{tail}} = 14$, which covers the numerical progress answer, the surrounding `<progress>` tags, and the sequence ending tokens in the default response format. We use $p_{\text{tail}} = 0.5$ for feature-level dropout during training.

The corrupted hidden states $\tilde{H} = \{\tilde{h}_i\}_{i=1}^{L}$ are then passed to the attention pooler together with the standard attention mask. Importantly, the tail tokens are not removed from pooling by setting their attention mask to zero. Instead, their hidden features are partially corrupted. This soft masking weakens direct access to the ground-truth progress tokens while keeping the tail positions in the computation graph, allowing gradients to still flow through the complete response sequence.

At inference time, no tail dropout is applied. When the value branch is enabled, the model runs a forward pass over the generated sequence and predicts the progress value from the resulting hidden states.

C.3.3 Two-stage Training Pipeline

As described in Section 3.2.3, ProcVLM is trained in two stages: large-scale pretraining on the ProcVQA training split, followed by refinement on a curated subset constructed from human-selected trajectories. We provide the training curves in Figure 6 to analyze the distinct roles of the two stages.

Figure 6 reports representative metrics throughout training on both the in-distribution `test_full` split and the out-of-distribution `valid_full` split. F1@50 directly measures action segmentation quality by matching predicted and ground-truth subtask intervals. SA evaluates the semantic accuracy of predicted subtask plans. GS measures progress regression accuracy. MCC@95 evaluates success-state prediction by converting progress into a binary decision, where $p \geq 95$ and $\hat{p} \geq 95$ indicate ground-truth and predicted success, respectively. VOC and KT both measure intra-trajectory ordering ability, i.e., whether predicted progress values preserve the temporal order of states within a trajectory.

The Stage 1 curves show that large-scale pretraining is crucial for generalization. Training on the ProcVQA training split consistently improves planning, progress estimation, success prediction, and trajectory-ordering metrics on both in-distribution and out-of-distribution splits, indicating that diverse procedure-aware supervision helps learn transferable task-progress representations.

Stage 2 further refines the pretrained model on the curated subset. After switching to higher-quality data, the model gains precision on metrics requiring accurate subtask boundaries, calibrated progress values, and reliable temporal ordering, with stronger improvements on the in-distribution split and smaller gains or fluctuations on the out-of-distribution split. Together, these trends support the two-stage design: Stage 1 builds generalizable procedural representations, while Stage 2 sharpens subtask and progress prediction.

Training Hyperparameters. The main training hyperparameters are summarized in Table 9. Both stages use the same joint objective described above and train the VLM backbone together with the progress value head. They differ mainly in data distribution, optimization schedule, and value-head regularization: Stage 1 uses a larger learning rate and lower value-head regularization for broad pretraining, while Stage 2 uses a smaller learning rate and stronger refinement regularization on the curated subset.

Table 9: Training hyperparameters for the two-stage ProcVLM training pipeline.

Hyperparameter	Stage 1 Pretraining	Stage 2 Refinement
Training data	`train_s1_8k`	`train_s2_8k`
Token scale	∼20B	2.8B
Backbone initialization	Qwen3-VL-2B-Instruct	Stage 1 checkpoint
Trainable modules	VLM backbone, vision tower, multimodal projector, value head	VLM backbone, vision tower, multimodal projector, value head
LoRA	Disabled	Disabled
Precision	bf16	bf16
Context length	8192	8192
Data packing	Enabled	Enabled
Per-device batch size	1	1
Gradient accumulation	64	64
Effective batch size	$64 \times N_{\mathrm{GPU}}$	$64 \times N_{\mathrm{GPU}}$
Base learning rate	1×10^{-5}	5×10^{-6}
Multimodal projector learning rate	1×10^{-5}	4×10^{-6}
Vision tower learning rate	5×10^{-6}	2×10^{-6}
Optimizer	AdamW	AdamW
Learning-rate schedule	Cosine	Cosine
Warmup ratio	0.03	0.05
Weight decay	0.01	0.01
Training length	∼1.1–1.2 epochs	∼3–4 epochs
Image pixel range	32^2–512^2	32^2–512^2
Video sampling	2 FPS, up to 512 frames	2 FPS, up to 512 frames
Regression loss ℓ_{reg}	L_1	L_1
Value-loss weight λ	0.2	0.3
Value-head dropout	0	0.05
Value noise std.	0.03	0.01
Distributed training	DeepSpeed ZeRO-2	DeepSpeed ZeRO-2 with offload
Gradient checkpointing	Enabled	Enabled

D ProcVQA Evaluation Details

This section describes the ProcVQA benchmark and evaluation metrics used in Section 4.1, together with the reward-model evaluation protocol for the ProcVQA comparison in Section 4.2.

D.1 Benchmark and Splits

ProcVQA is a human-selected ECoT-based VQA benchmark for embodied procedural understanding. It covers three manipulation-understanding tasks: action segmentation, future planning, and task progress estimation. To evaluate both in-domain performance and cross-domain generalization, we divide ProcVQA into an in-distribution (ID) split and an out-of-distribution (OOD) split. The ID split is derived from training-domain datasets, including DROID, Bridge, Table30, and selected OXE subsets, while the OOD split is constructed from unseen RoboTwin tasks.

D.2 Metrics

Boundary F1 score (BF1). BF1 is calculated as follows. Given predicted and ground-truth action segments, we first extract segment boundaries and remove duplicated boundary positions. A predicted boundary is matched to a ground-truth boundary if their temporal distance is within 5% of the sequence length. Each boundary can be matched at most once. BF1@5 is then computed as the

boundary-level F1 score from matched and unmatched boundaries. We use BF1@5 as the primary segmentation metric because it penalizes both missing and redundant boundaries.

Matched boundary localization error (mMAE). mMAE measures the temporal localization error of matched boundaries. It is computed as the mean absolute distance between each matched predicted boundary and its corresponding ground-truth boundary. Since mMAE is calculated only over matched boundaries, it does not penalize false positives or false negatives. We therefore report it as an auxiliary metric rather than the primary action segmentation metric.

Future planning success (Success). Success measures whether the model-generated future task plan can be successfully executed and satisfy the given instruction. We compute this metric through anonymized and randomly ordered human evaluation, where annotators compare model outputs without knowing the model identity.

Value-Order Correlation (VOC). VOC evaluates within-trajectory progress ordering. We compute Spearman's rank correlation between predicted progress values and the temporal order of the corresponding observations. It does not measure numerical agreement with the procedure-defined progress targets.

Effective Progress Resolution (EPR). EPR measures the density of continuous progress regression. It discourages models from obtaining high VOC by predicting only a few discrete progress anchors, such as 0.25, 0.5, 0.75, and 1.0. For predicted progress values $\hat{p}$, EPR is defined as

$$\mathrm{EPR}_\tau(\hat{p}) = -\log_2 \min\left\{\Delta \in \{1/k \mid k \in \mathbb{N}_+\} \mid \Delta \cdot |\mathcal{B}_\Delta(\hat{p})| \geq \tau\right\}, \tag{3}$$

where $\mathcal{B}_\Delta(\hat{p})$ denotes the set of occupied quantization bins after quantizing $\hat{p}$ with bin width Δ. In our experiments, we report EPR@50 by setting $\tau = 0.5$.

D.3 Reward-Model Evaluation

This subsection details the zero-shot reward-model comparison on the ProcVQA-OOD progress-estimation subset in Section 4.2.1. We use the benchmark split and metrics described above, and specify the input construction and progress-score extraction for each baseline below.

Evaluation setting. We compare ProcVLM with Robometer and RoboDopamine using short local-window queries, typically containing 1–16 consecutive frames within about one second. Queries from the same trajectory are evaluated independently without chronological ordering. We report VOC and EPR@50 as defined above.

RoboDopamine. RoboDopamine trains a step-aware vision-language reward model to estimate fine-grained manipulation progress from multi-view state observations by predicting relative progress hops and fusing them into dense rewards. For each local-window query, we retain the temporal order of its frames and prepend a blank image as a neutral contrastive start anchor. This construction provides the contrastive input required by RoboDopamine without introducing observations from other query windows. We use the 4B variant of Robo-Dopamine-GRM-2.0 for this comparison.

Robometer. Robometer trains a Qwen3VL-based video-language reward model with both frame-level progress/success supervision and trajectory-comparison preference supervision, enabling dense reward prediction from robot videos and language instructions. In our evaluation, Robometer receives the instantaneous input window from each VQA query, and we use its last-frame prediction as the final progress estimate.

E RoboFAC Evaluation Details

This section provides additional details of the RoboFAC evaluation protocol and metric definitions used in Section 4.2.2.

E.1 Implementation Details

Benchmark. RoboFAC is a failure-centric robotic manipulation VQA benchmark for robotic failure analysis and correction. It contains diverse successful and failed manipulation executions, with structured annotations for task understanding, failure analysis, and correction. RoboFAC consists

of both real-robot and simulated data, and we use only the real-robot subset for evaluation. We use RoboFAC to evaluate whether ProcVLM can generalize to unseen tasks with only a small number of demonstrations.

One-shot split construction. We first split all trajectories into success and failure sets. We then construct task-level one-shot training sets separately from these two sets. For the success set, we select one trajectory for each task as the one-shot successful demonstration. For the failure set, we select one trajectory for each task–failure-type pair, so that different failure modes are covered with minimal supervision. All remaining trajectories are used for testing.

Viewpoint coverage. The one-shot demonstrations are selected at the trajectory level and do not explicitly cover all camera viewpoints. As a result, the test set may contain observations from viewpoints that are absent from the one-shot demonstrations. This design makes the evaluation more challenging and better reflects practical reward-model deployment, where a small number of demonstrations cannot exhaustively cover all visual variations.

Success detection. For binary success detection, each model performs inference on the full test set using the visual observation from the last temporal window of each trajectory. Robometer predicts completion with its success head, whose probability is discretized into a binary success label. For RoboDopamine and ProcVLM, we use the predicted progress value and apply a fixed success threshold to obtain the binary completion label.

Full-trajectory progress evaluation. For full-trajectory progress evaluation, running all test trajectories is computationally expensive. We therefore sample 100 test trajectories in total, evenly from the success and failure sets, and evaluate progress prediction along the sampled trajectories.

E.2 METRICS

VOC on successful trajectories. $\mathrm{VOC}_{\mathrm{succ}}$ is computed only on successful trajectories and follows the same definition as VOC in ProcVQA. It measures whether predicted progress values preserve the correct temporal order along a successful execution.

MAE on failed trajectories. $\mathrm{MAE}_{\mathrm{fail}}$ is computed only on failed trajectories. For each failed trajectory, we identify the earliest frame that reaches the maximum predicted progress value:

$$t^* = \min \arg\max_t \hat{p}_t,$$

where $\hat{p}_t$ denotes the predicted progress value at frame t. We treat t^* as the predicted turning point, corresponding to the estimated failure onset. Given the human-annotated cutoff index t_{cut}, $\mathrm{MAE}_{\mathrm{fail}}$ is computed as

$$\mathrm{MAE}_{\mathrm{fail}} = \frac{1}{N_{\mathrm{fail}}} \sum_{i=1}^{N_{\mathrm{fail}}} \frac{\left| t^{*(i)} - t_{\mathrm{cut}}^{(i)} \right|}{T_i - 1}. \tag{4}$$

Here, $T_i \geq 2$ is the number of evaluated frames in trajectory i, and both cutoff indices are zero-based. The error is normalized within each trajectory before averaging across failed trajectories.

MCC for success detection. We use the Matthews correlation coefficient (MCC) to evaluate binary success detection. Predicted progress scores or success probabilities are converted into completion labels using a fixed threshold, and MCC is then computed against the ground-truth success/failure labels.

Latency. Latency is measured in seconds during full-trajectory progress evaluation and reports the average inference time required to process one trajectory. We measure latency on NVIDIA RTX A6000 GPUs, using single-GPU deployment for all models except Qwen3.5-27B, which is deployed with 4-way tensor parallelism.

F REWARD-GUIDED POLICY OPTIMIZATION DETAILS

This section provides additional details on the reward-guided policy optimization experiments in Section 4.3.

F.1 On-Policy Reinforcement Learning on RoboTwin

Tasks and shared RL setup. We optimize OpenVLA-OFT with GRPO in RLinf on Place Container Plate and Move Can Pot. Within each task, all methods share the SFT initialization and policy training/evaluation settings. Each update uses 480 trajectories (48 environments $\times$ 10 rollout epochs; 60 groups of eight), with rewards computed after each 25-step action chunk.

Reward-model adaptation and inference. ProcVLM and RoboDopamine use two stage-annotated demonstrations per task; Robometer uses 50 without manual annotations. For ProcVLM and Robometer, `w8` denotes an ordered window of up to eight front-camera frames.

- **ProcVLM (procedure-grounded, w8).** We LoRA-tune on 335 procedure-grounded examples for Move Can Pot and 344 for Place Container Plate; progress is parsed from the model's generated output.
- **Robometer (linear, w8).** We LoRA-tune Robometer-4B on 45 demonstrations, holding out five for validation, with the official rank 32 and $\alpha = 64$ configuration. Progress labels discretize normalized trajectory time into ten bins.
- **RoboDopamine (Forward + Backward + Incremental, multi-view).** We fully fine-tune on 2,000 comparison samples per task using synchronized front and left/right wrist images; inference fuses all three modes across these views.
- **RoboDopamine (Forward + Backward, single-view).** On Move Can Pot, we also evaluate forward–backward fusion using only the front camera.

Reward construction and optimization. Progress predictions are normalized to $[0, 1]$; consecutive differences are clipped to $[-0.2, 0.2]$. The progress reward has weight 1.0 and the environment success reward 0.1, emphasizing the learned signal. Following RLinf's official configuration, the policy uses LoRA rank 32, with global batch size 480, weight decay 0.01, and gradient clipping at 1.0. GRPO uses lower/upper clipping parameters 0.2/0.28, sampling temperature 1.6, and zero KL and entropy coefficients. The learning rate, discount factor, and episode horizon are $(10^{-5}, 0.99, 150)$ for Place Container Plate and $(2 \times 10^{-4}, 1.0, 200)$ for Move Can Pot.

Evaluation and active-time comparison. Every five updates, we evaluate 48 trajectories and report episodic success (`eval/success_once`). We compare success against active time on both tasks, using the first recorded evaluation as the time origin. Active time excludes downtime associated with policy restarts and includes evaluation and checkpointing overhead, with all models following the same evaluation and checkpointing protocol.

Curve presentation. In Figure 3, dashed lines show raw evaluation success rates; solid lines show rates smoothed by a three-point median filter followed by a weighted moving average, assigning weights of 0.25, 0.5, and 0.25 to the previous, current, and next points, respectively.

F.2 Real-Robot Reward Fine-tuning

Task and data setup. We conduct real-robot experiments on a tabletop stack-bowls task with a single-arm JAKA robot. The task requires the robot to pick up a bowl on the table and place it into a target bowl, forming a properly nested bowl stack. The real-robot training set contains 50 human teleoperated demonstration trajectories, approximately evenly distributed across scenes with two, three, and four bowls. We train both SFT and RFT policies for 10k steps with a batch size of 64, and visualize representative rollouts at both 5k and 10k training steps. Due to the higher difficulty of the three-bowl and four-bowl settings and our limited real-robot evaluation budget, preliminary runs showed very low success rates on these settings. Therefore, we restrict the real-robot evaluation to the two-bowl setting, where each rollout requires one bowl to be placed into the target bowl.

Reward-guided fine-tuning. Both SFT and RFT start from the same $\pi_{0.5}$ policy initialization and use the same demonstration data. We use SJTU Evo-RL, an offline RL framework supporting value inference and advantage-conditioned policy training. SFT follows standard supervised fine-tuning. For RFT, ProcVLM assigns progress scores as dense rewards to the training trajectories, and Evo-RL estimates advantages over a 50-step horizon. The top 30% of samples by advantage receive positive labels and the remaining samples receive negative labels; these labels serve as auxiliary conditions during fine-tuning. The teleoperated demonstrations contain noisy local behaviors, including repeated



SFT, 5k training steps, grasp failed.



SFT, 5k training steps, partial success.



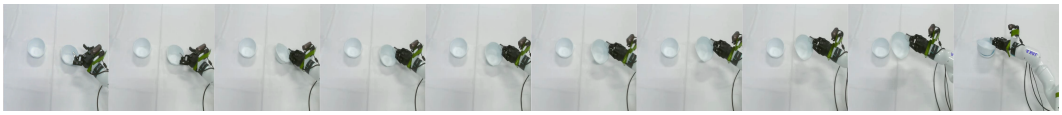
RFT, 5k training steps, partial success.



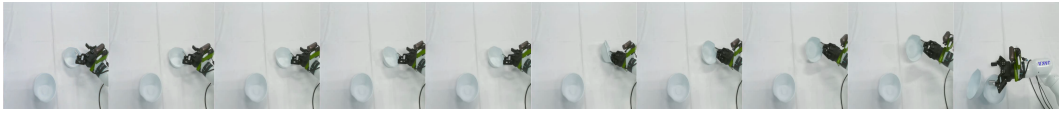
RFT, 5k training steps, full success.



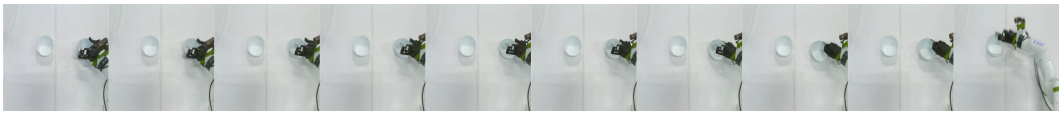
SFT, 10k training steps, partial success.



SFT, 10k training steps, full success.



RFT, 10k training steps, partial success.



RFT, 10k training steps, full success.

Figure 7: Representative real-robot rollout records on the JAKA tabletop stack-bowls task. We show examples from SFT and RFT policies at both 5k and 10k training steps. Each method is evaluated with 12 rollouts under the two-bowl setting, and the final score is computed from the nesting quality of the placed bowl.

grasp retries. Advantage conditioning helps distinguish higher-advantage segments from less useful behaviors, offering a possible explanation for the larger RFT gain at 5k steps.

Evaluation protocol. For each method, we conduct 12 real-robot rollouts under the two-bowl setting. Success is measured by the degree of proper nesting in the final bowl configuration. For each rollout, we assign a nesting score $s \in [0, 1]$ according to the following rubric:

- $s = 0.0$ **(failure)**: the stacking action is not completed within 30 seconds, such as when the bowl is not grasped or the robot does not attempt insertion.
- $s = 0.5$ **(partial success)**: the robot attempts to insert the bowl, but the placed bowl slips out of the target bowl or is placed immediately next to the target bowl.
- $s = 1.0$ **(full success)**: the robot successfully places the bowl inside the target bowl.

The final score of each method is computed as the average nesting score over 12 rollouts. We report this value as the soft success rate, where partial placements receive half credit.

Rollout visualization. Figure 7 shows representative real-robot rollout records under the two-bowl evaluation setting. These examples include different execution outcomes and illustrate how the final nesting score is assigned based on the resulting bowl configuration.

Table 10: Real-robot experimental setup on the JAKA stack-bowls task.

Item	Description
Robot platform	JAKA robot, single arm
Scene	Tabletop stack-bowls task
Training data	50 human teleoperated demonstrations
Demonstration settings	Two-bowl, three-bowl, and four-bowl scenes
Evaluation setting	Two-bowl scene only
Rollouts per method	12 rollouts
Metric	Soft success rate

G Additional Analyses

G.1 Annotation Quality Audit

We manually audit the automatically generated subtask annotations for 92,465 trajectories under a strict trajectory-level criterion. A trajectory qualifies only if all subtask descriptions are correct and all temporal boundaries align with the video within a tolerance of one second. Any incorrect subtask description or boundary deviation exceeding one second causes the entire trajectory to be rejected. The resulting pass rate therefore measures the proportion of fully qualified trajectories, rather than the accuracy of individual frames or annotations.

Table 11: Manual audit of automatically generated subtask annotations. A trajectory qualifies only if every subtask description and temporal boundary satisfies the audit criteria.

Dataset	Reviewed	Qualified	Pass rate
DROID	55,864	18,682	33.44%
OXE	31,560	24,172	76.59%
Table30	2,962	1,286	43.42%
Others	2,079	1,576	75.81%
Total	**92,465**	**45,716**	**49.44%**

As shown in Table 11, 45,716 trajectories pass the audit, yielding an overall pass rate of 49.44%. Manual inspection identifies missing or underspecified task instructions in DROID and accumulated segmentation errors in longer Table30 trajectories as common sources of failure. Because a single annotation error can cause an entire trajectory to fail this stringent audit, rejected trajectories may still contain substantial useful supervision. Their diverse visual scenes and task–subtask structures provide valuable pretraining signals, broadening the range of tasks and environments encountered by the model. We therefore use the broader automatically annotated corpus for pretraining and select approximately 15K qualified trajectories for refinement, combining broad task coverage with precise subtask supervision.

G.2 Ablation of the Value Head

We evaluate the contribution of the continuous value head on the ID and OOD splits of ProcVQA. Using the same jointly trained checkpoint, we compare progress predictions from the value head with progress values parsed from the generated text. The *w/o Value Head* condition bypasses the value head only at inference; both conditions retain the same training history. This comparison therefore evaluates the choice of progress readout at inference. We report VOC for temporal ordering and EPR@50 for progress-score resolution.

Table 12: Ablation of the value head on ProcVQA. Both conditions use the same trained checkpoint; the condition without the value head obtains progress scores from generated text. Higher values are better for both metrics.

	ID		OOD	
Variant	VOC ↑	EPR@50 ↑	VOC ↑	EPR@50 ↑
w/ Value Head	**0.8058**	**5.7489**	**0.7282**	**6.1152**
w/o Value Head	0.8031	5.4400	0.6843	5.5457

Table 12 shows that the value head improves both temporal ordering and prediction resolution. On ID data, bypassing the value head has little effect on VOC but reduces EPR@50 by 5.4%. The effect is larger on OOD data, where VOC and EPR@50 decrease by 6.0% and 9.3%, respectively, relative to predictions from the value head. These results support using the continuous value head for progress estimation, particularly on unseen tasks.

G.3 Reward Modeling Cases

Zero-shot cases. We first evaluate ProcVLM in a fully zero-shot setting on multiple tasks from RoboMIND (Wu et al., 2024) and RoboTwin. As shown in Figure 8, ProcVLM can identify the current action stage, reason about remaining steps, and produce task-conditioned progress feedback without task-specific demonstrations or reward-model adaptation. These cases illustrate the transferable progress perception learned from procedure-aware pretraining.

Zero-shot reward editing. We finally present a zero-shot reward editing case on a RoboMIND pick-and-place task. As shown in Figure 9, ProcVLM is applied to the same video under two task instructions: "put the apple into the basket" and "put the apple into the basket and move the basket to the upper corner." The resulting reward curves change with the edited task description, showing that ProcVLM grounds reward prediction in task-conditioned subtask structure rather than only superficial visual progress.

One-shot adaptation cases. We further evaluate ProcVLM under a one-shot adaptation setting, where only one successful demonstration is used for LoRA tuning in each scenario. As shown in Figures 10, 11, 12, and 13, ProcVLM quickly adapts to RoboMIND and RoboFAC scenarios containing successful, failed, and retry executions, providing meaningful progress feedback beyond the demonstrated success trajectory.

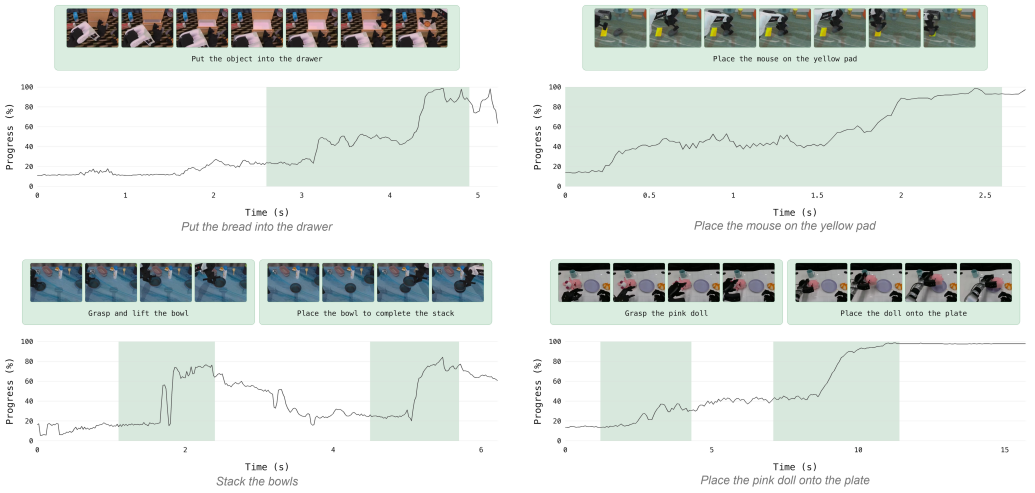


Figure 8: Zero-shot reward modeling cases on RoboMIND and RoboTwin. ProcVLM provides task-conditioned progress feedback across different manipulation tasks without task-specific adaptation.

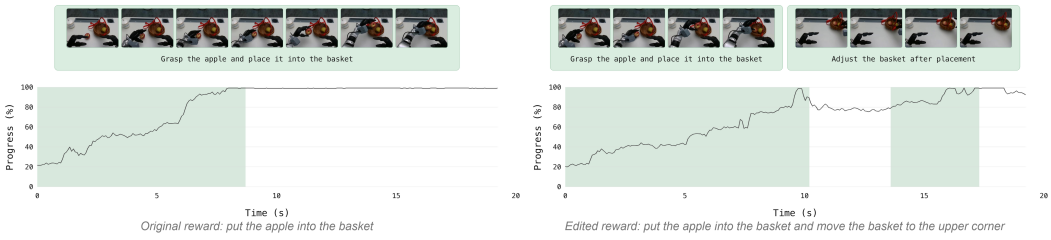


Figure 9: Zero-shot reward editing on the same video sequence. Left: reward for "put the apple into the basket." Right: edited reward for "put the apple into the basket and move the basket to the upper corner."

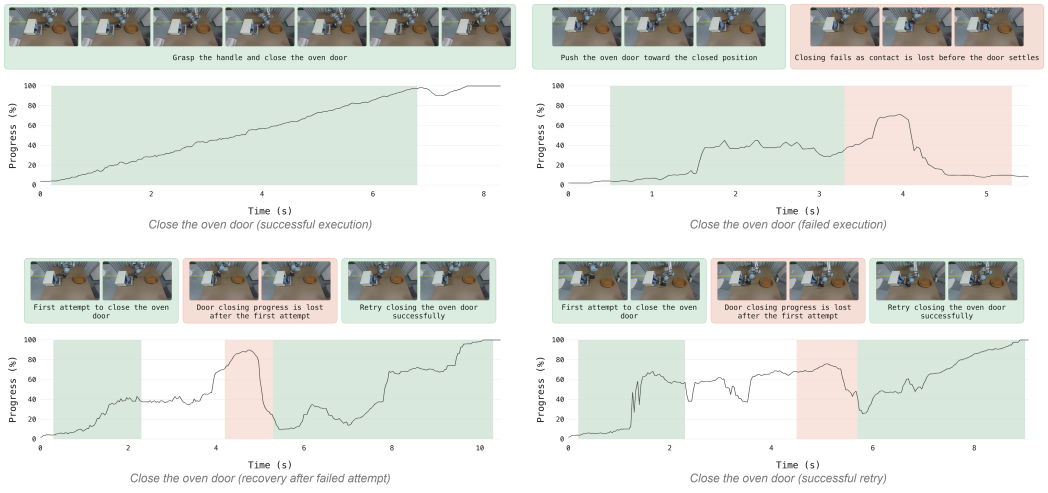


Figure 10: One-shot adaptation case for a close-oven task. ProcVLM adapts from one successful demonstration and evaluates different execution outcomes, including failures and retry-based recoveries.

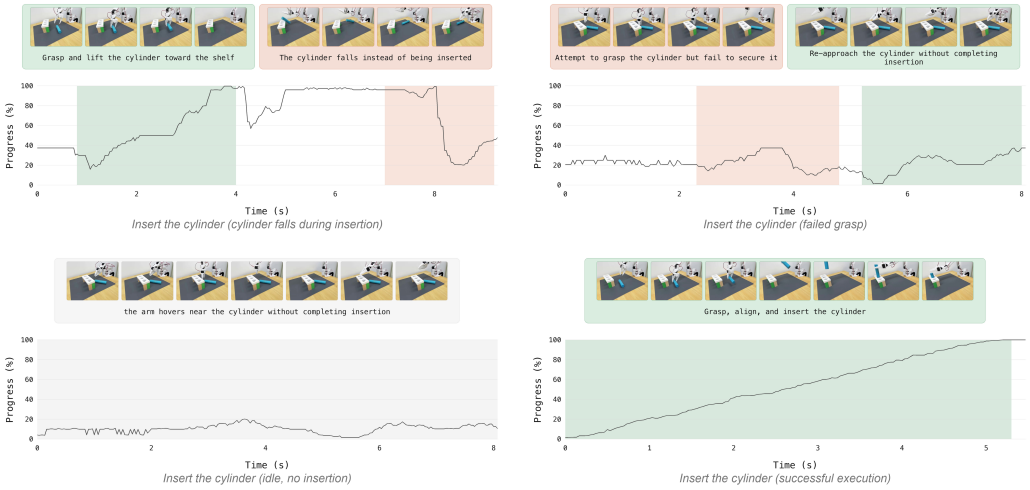


Figure 11: One-shot adaptation case for an insert-cylinder task. ProcVLM transfers the demonstrated task structure to executions with successful, failed, and idle behaviors.

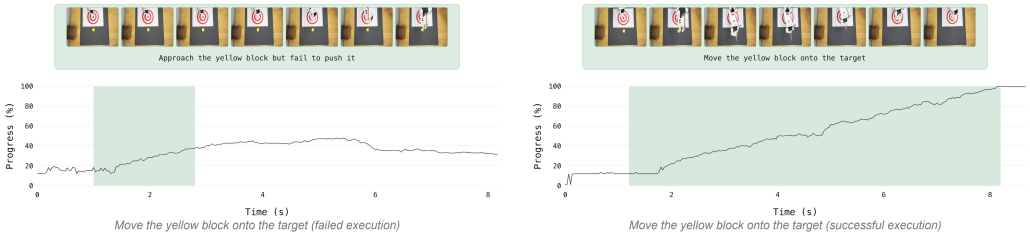


Figure 12: One-shot adaptation case for a move-target task. ProcVLM transfers the demonstrated task structure to successful and failed executions.

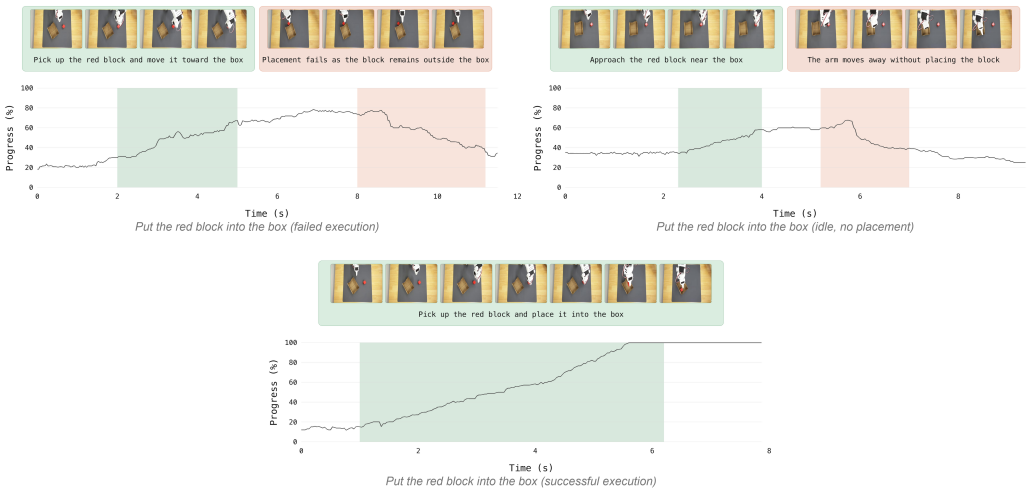


Figure 13: One-shot adaptation case for a put-in-box task. ProcVLM generalizes from one successful demonstration to evaluate successful and failed executions.