PRM-AS-A-JUDGE: A DENSE EVALUATION PARADIGM FOR FINE-GRAINED ROBOTIC AUDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

Robotic policies are commonly evaluated by binary success rates, which compress an entire execution into a terminal outcome and provide limited information about the execution process. We present ***PRM-as-a-Judge***, a process-level evaluation paradigm that uses Process Reward Models (PRMs) to estimate dense task progress from robot rollout observations and represent each execution as a progress curve. Based on this curve, we introduce the *OPD (Outcome–Process–Diagnosis)* metric system, which organizes trajectory evaluation around three sequential questions: (1) how far the execution reaches (*Outcome*), (2) how efficiently it advances (*Process*), and (3) how progress deteriorates through regression or stagnation (*Diagnosis*). To evaluate progress judges before policy auditing, we introduce *RoboPulse*, an interval-level benchmark that measures their ability to identify whether task progress increases or decreases over temporal intervals. Experiments across diverse tasks and data sources show a clear advantage of specialized PRM judges over general-purpose vision-language model (VLM) judges in task-progress assessment. Finally, we apply ***PRM-as-a-Judge*** to large-scale policy auditing in both simulation and real-world rollouts, revealing: (1) stage-wise bottlenecks, (2) differences in successful execution quality, and (3) distinct regression and stagnation patterns that are not captured by terminal success rates. We will release RoboPulse and the complete PRM-as-a-Judge evaluation toolkit.

1 INTRODUCTION

Robotic manipulation is increasingly moving toward long-horizon and contact-rich tasks. Binary success rate remains the dominant evaluation metric in widely used benchmarks (Li et al., 2024b; James et al., 2020; Mees et al., 2022), but it reduces each rollout to a terminal outcome. As a result, rollouts with the same outcome cannot be distinguished by how far they progress, how efficiently they progress, or whether task progress exhibits regression or stagnation. As policies become more capable and diverse, these process-level distinctions become increasingly important for evaluation.

Recent work has introduced richer robotic evaluation using predefined subgoals, task-specific rules, or privileged robot and environment states (Wang et al., 2025; ElMallah et al., 2025). We instead study an observation-based setting that uses the task instruction and recorded rollout observations, allowing the evaluation pipeline to be applied across tasks, policies, simulation, and real-world execution.

Under this setting, we propose ***PRM-as-a-Judge***, a process-level evaluation paradigm based on Process Reward Models (PRMs). A progress judge converts the task instruction and rollout observations into a *task progress curve*. Based on this curve, we define the **OPD (Outcome–Process–Diagnosis)** metric system, which organizes trajectory evaluation around three sequential questions: *how far* the execution reaches (*Outcome*), *how efficiently* it progresses (*Process*), and *how* task progress deteriorates through regression or stagnation (*Diagnosis*). Fig. 1 summarizes the framework.

Reliable policy auditing requires accurate estimation of task-progress changes. We therefore introduce ***RoboPulse***, an interval-level benchmark for evaluating progress judges. RoboPulse measures whether a judge can identify task-progress increase and decrease over temporal intervals. Across diverse tasks and data sources, specialized PRMs show a clear advantage over general-purpose vision-language models (VLMs) in task-progress assessment.

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

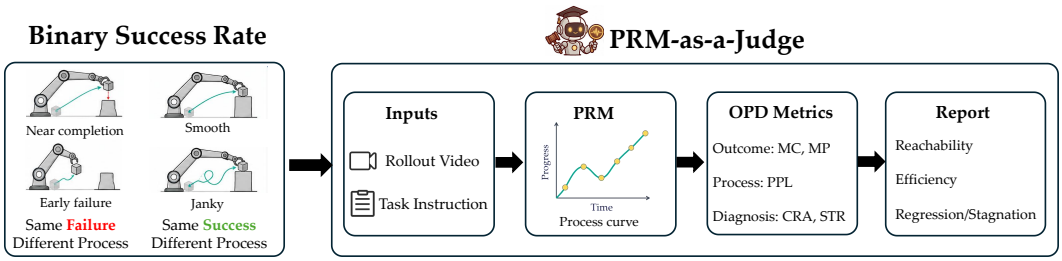


Figure 1: **Overview of PRM-as-a-Judge. Left:** Binary success rate cannot distinguish rollouts with the same terminal outcome but different execution processes. **Right:** A progress judge converts the task instruction and rollout observations into a task progress curve, from which OPD metrics are computed and summarized into a report of reachability, efficiency, regression, and stagnation.

We further apply PRM-as-a-Judge to large-scale policy auditing in simulation and real-world rollouts. The resulting task progress curves and OPD metrics reveal stage-wise progress bottlenecks, differences in progress efficiency among successful rollouts, and regression and stagnation patterns across policy families. These results provide process-level information beyond terminal success rate.

In summary, our main contributions are:

- We propose ***PRM-as-a-Judge***, which constructs task progress curves from robot rollouts and evaluates them with the **OPD (Outcome–Process–Diagnosis)** metric system.
- We introduce ***RoboPulse***, an interval-level benchmark for task-progress increase and decrease, and show a clear advantage of specialized PRMs over general-purpose VLM judges in task-progress assessment.
- We conduct large-scale policy auditing in simulation and real-world rollouts, revealing stage-wise bottlenecks, differences in progress efficiency, and regression and stagnation patterns.

We will release RoboPulse and the PRM-as-a-Judge evaluation toolkit.

2 PRM-AS-A-JUDGE

PRM-as-a-Judge maps a task instruction and recorded rollout observations to a *task progress curve* using a progress judge. OPD then summarizes the curve in terms of reached progress, progress efficiency, regression, and stagnation.

2.1 PROGRESS JUDGE AND TASK PROGRESS CURVE

Given a task instruction c and rollout observations $o_{0:T} = (o_0, o_1, \ldots, o_T)$, a progress judge estimates normalized task progress over the rollout. We anchor the initial progress at $p_0 = 0$ and represent the resulting task progress curve as

$$p_{0:T} = (p_0, p_1, \ldots, p_T), \qquad p_t \in [0, 1], \tag{1}$$

where larger p_t denotes greater progress toward task completion. Progress judges may produce different native outputs. For OPD computation, their outputs are converted to this common normalized representation. Implementation details are provided in App. B.

We use Process Reward Models (PRMs) as progress judges. PRMs provide task-conditioned progress estimates from robot observations and can score recorded policy rollouts post hoc. Progress-judging reliability is evaluated separately on RoboPulse before downstream policy auditing.

2.2 THE OPD METRIC SYSTEM

Given a task progress curve, OPD (Outcome–Process–Diagnosis) organizes trajectory evaluation around three sequential questions: (1) how far does the execution reach (*Outcome*), (2) how efficiently does it progress (*Process*), and (3) how does progress deteriorate through regression or stagnation (*Diagnosis*). Tab. 1 summarizes the metrics.

Table 1: **Overview of the OPD metric system.** Pref. denotes the preference direction (↑: higher is better; ↓: lower is better).

Metric	Pref.	Definition
Outcome		
Max Progress (MP)	↑	Maximum task progress reached during the rollout.
Milestone Coverage (MC@q)	↑	Whether task progress reaches milestone q.
Process		
Path-weighted Progress Length (PPL)	↑	Progress efficiency over the full rollout, weighted by the highest progress reached.
Diagnosis		
Cumulative Regret Area (CRA)	↓	Normalized cumulative gap below the best-so-far progress.
Stagnation Ratio (STR)	↓	Fraction of consecutive steps with negligible task-progress change.

Outcome. Outcome measures how far the execution progresses toward task completion.

- **Max Progress (MP).** MP is the highest progress reached during the rollout:

$$\mathrm{MP} = \max_{0 \le t \le T} p_t. \tag{2}$$

- **Milestone Coverage (MC@q).** MC@q indicates whether the rollout reaches progress milestone q. For $q \in \{25, 50, 75, 100\}$,

$$\mathrm{MC@}q = \mathbb{I}\left[\mathrm{MP} \ge \frac{q}{100}\right]. \tag{3}$$

  When averaged over rollouts, MC@q is the proportion of rollouts that reach milestone q. MC@100 is derived from the task progress curve and is reported separately from the benchmark-provided success rate.

Process. Process measures how efficiently task progress is accumulated over the rollout.

- **Path-weighted Progress Length (PPL).** PPL is defined as

$$\mathrm{PPL} = \mathrm{MP} \cdot \frac{\mathrm{MP}}{\sum_{t=1}^{T} |p_t - p_{t-1}|}. \tag{4}$$

  PPL is defined as 0 when the denominator is 0. The ratio is maximized when progress increases monotonically from 0 to MP, while the leading MP weights the score by maximum progress.

Diagnosis. Diagnosis characterizes regression and stagnation during execution.

- **Cumulative Regret Area (CRA).** CRA measures the cumulative gap between current progress and the best progress reached so far:

$$\mathrm{CRA} = \frac{1}{(T+1)\,\mathrm{MP}} \sum_{t=0}^{T} \left(\max_{0 \le i \le t} p_i - p_t \right). \tag{5}$$

  CRA is defined as 0 when $\mathrm{MP} = 0$. A higher CRA indicates a larger or more persistent gap below the best-so-far progress.

- **Stagnation Ratio (STR).** STR measures the fraction of consecutive steps with negligible task-progress change:

$$\mathrm{STR} = \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}\left[|p_t - p_{t-1}| < \epsilon\right], \tag{6}$$

  where $\epsilon > 0$ is the threshold for negligible task-progress change. We use $\epsilon = 0.5\%$ throughout; sensitivity analysis is provided in App. B. A higher STR indicates that a larger fraction of consecutive steps exhibit negligible task-progress change.



Figure 2: **Task vocabulary in RoboPulse.**

Table 2: **Data statistics of RoboPulse across different sources.**

Data source	Episodes	Tasks	Frames	Intervals
Real-World Data				
RoboChallenge-Table30 (Yakefu et al., 2025)	79	25	3,448	882
RoboChallenge-Table30-V2 (Ma et al., 2026)	79	26	2,068	169
ViFailback (Zeng et al., 2026)	265	103	4,760	544
RoboFAC-Real (Ye et al., 2025)	16	6	261	33
Simulation Data				
LIBERO (Liu et al., 2023)	111	39	2,462	239
RoboCasa (Nasiriany et al., 2024)	34	24	1,935	79
SimplerEnv (Li et al., 2024b)	24	9	566	104
RoboTwin 2.0 (Chen et al., 2025b)	22	7	822	58
RoboFAC-Sim (Ye et al., 2025)	70	36	730	136
Total	**700**	**275**	**17,052**	**2,244**

3 RoboPulse: Benchmarking Progress Judges

Reliable policy auditing requires a progress judge that can identify task-progress changes from robot rollouts. We therefore introduce *RoboPulse*, an interval-level benchmark for evaluating task-progress increase and decrease. Each annotated interval is labeled as either *Rising* or *Falling*, corresponding to an increase or decrease in task progress.

3.1 Benchmark Task

Given a task instruction c and a temporal interval of rollout observations

$$O_{t_s:t_e} = (o_{t_s}, \ldots, o_{t_e}), \tag{7}$$

the interval is assigned a task-progress direction

$$y_{[t_s,t_e]} \in \{-1, +1\}, \tag{8}$$

where $+1$ denotes *Rising* progress and -1 denotes *Falling* progress. A progress judge produces task-progress predictions over the rollout, which are converted to directional predictions and evaluated against the annotated direction of the corresponding interval.

3.2 Dataset and Annotation

RoboPulse contains 700 manipulation trajectories spanning 275 task entries, 17,052 sampled frames, and 2,244 human-annotated intervals. The dataset includes 439 real-world trajectories and 261 simulation trajectories, covering diverse robot embodiments, scenes, camera viewpoints, and execution conditions. The tasks span atomic manipulation skills, compositional tasks, and long-horizon execution, including grasping, placing, pushing, pulling, articulated-object manipulation, sorting, stacking, tool use, and multi-stage manipulation. Fig. 2 illustrates the task vocabulary, and Tab. 2 summarizes the data sources and statistics.

For each trajectory, annotators first review the complete execution together with its natural-language task instruction and then identify temporally contiguous intervals with a consistent direction of task-progress change. Each annotated interval is assigned one of two labels:

- ***Rising*** $(+1)$**:** task progress increases toward the goal, such as approaching a target object, moving an object toward its target state, or completing a required manipulation step.
- ***Falling*** (-1)**:** task progress decreases, such as moving a task-relevant state away from the goal or reversing previously achieved progress.

Annotators use the complete trajectory as temporal context when determining interval boundaries and labels. This context distinguishes local fluctuations from task-progress increase or decrease.

3.3 Evaluation Protocol

We evaluate two groups of progress judges: *specialized PRMs* and *general-purpose VLMs*. All judges receive the task instruction and sampled rollout observations. All rollout videos are uniformly sampled at 1 FPS.

Progress judges produce different native outputs, including absolute progress scores, progress changes, and categorical predictions. For an absolute progress prediction p_t, we compute

$$d_t = p_t - p_{t-1}. \tag{9}$$

Native progress-change outputs are used directly, while categorical outputs are mapped to the common *Rising*/*Falling* label space.

Directional predictions within each annotated interval are evaluated against its human-annotated progress direction. We report class-wise Precision, Recall, and F1, together with overall Macro-F1 and Accuracy. Evaluation implementation, including output processing and boundary filtering, is provided in App. C.

4 EXPERIMENTS

We conduct two complementary sets of experiments. First, we evaluate progress-judging reliability through interval-level assessment on RoboPulse and consistency with benchmark-provided task outcomes. Second, we apply PRM-as-a-Judge to policy auditing in both simulation and real-world robot rollouts.

Our experiments address the following questions:

- **RQ1: Reliability of Progress Judging.** Can progress judges reliably identify task-progress increases and decreases, and do the resulting progress curves remain consistent with benchmark-provided task outcomes?
- **RQ2: Stage-wise Reachability.** Can OPD reveal how far policies progress and where execution tends to stop when terminal success provides limited resolution?
- **RQ3: Execution Quality among Successful Rollouts.** Can process and diagnosis metrics distinguish successful executions with different levels of efficiency, regression, and stagnation?
- **RQ4: Regression and Stagnation Patterns.** Can OPD distinguish different forms of execution deterioration among failed rollouts?
- **RQ5: Real-World Validation.** Do the process-level distinctions revealed by OPD remain visible in real-world robot rollouts?

4.1 EXPERIMENTAL SETUP

Our experimental setup consists of progress-judge evaluation and downstream policy auditing. We first evaluate candidate progress judges on RoboPulse, then use the selected judge for policy auditing on RoboTwin 2.0 and RoboDojo-RealWorld.

Progress-judge evaluation. We evaluate progress judges on RoboPulse following Sec. 3. We compare specialized PRMs (Tan et al., 2025; Liang et al., 2026; Wu et al., 2026; Chen et al., 2026a; Zhai et al., 2025; Ma et al., 2024; Liu et al., 2026) with general-purpose VLMs (OpenAI, 2026; Deepmind, 2026; Qwen Team, 2026; Anthropic, 2026) under the common task of identifying *Rising* and *Falling* task progress. We report class-wise Precision, Recall, and F1, together with overall Macro-F1 and Accuracy.

Policy auditing. For simulation, we use RoboTwin 2.0 (Chen et al., 2025b) and evaluate ACT (Zhao et al., 2023), DP (Chi et al., 2025), RDT (Liu et al., 2024), π_0 (Black et al., 2024), and OpenVLA-OFT (Kim et al., 2025) on seven long-horizon manipulation tasks. All policies follow the official training configuration with the same data budget and are evaluated with 50 rollouts per policy per task. For real-world validation, we use released rollouts from RoboDojo-RealWorld (Chen et al., 2026b), covering 18 physical manipulation tasks and nine embodied policies from prior work (Physical Intelligence et al., 2025; Cai et al., 2026a;b; Galaxea Team, 2025; Zheng et al., 2025; Black et al., 2024; Ye et al., 2026; NVIDIA, 2026; Spirit AI Team, 2026). The same observation-based progress-judging and OPD pipeline is applied directly to the recorded rollout videos. Three representative RoboTwin tasks are reported in the main paper, with complete results in App. D.

Table 3: **Progress-judge evaluation on RoboPulse.** We report class-wise Precision, Recall, and F1, together with overall Macro-F1 and Accuracy.

Judge Model	*Rising*			*Falling*			*Overall*	
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Macro-F1**	**Accuracy**
Specialized PRMs								
Robo-Dopamine (Forward)	0.92	**0.92**	**0.92**	**0.79**	0.50	0.61	**0.77**	**0.84**
Robo-Dopamine (Backward)	0.92	0.88	0.90	0.72	0.55	**0.63**	0.76	0.82
Robo-Dopamine (Incremental)	**0.93**	0.84	0.88	0.62	**0.57**	0.59	0.74	0.79
RoboMeter	0.89	0.86	0.87	0.47	0.54	0.50	0.69	0.80
LRM (Temporal)	0.84	0.71	0.77	0.29	0.42	0.34	0.56	0.66
LRM (Absolute)	0.90	0.31	0.46	0.35	0.21	0.26	0.36	0.29
TOPReward	0.85	0.75	0.80	0.28	0.38	0.32	0.56	0.68
VLAC	0.89	0.38	0.54	0.26	0.22	0.24	0.39	0.35
GVL	0.90	0.78	0.84	0.51	0.47	0.49	0.66	0.72
PRIMO	0.86	0.75	0.80	0.20	0.05	0.08	0.44	0.62
General-Purpose VLMs								
GPT-5.4	**0.94**	0.37	0.53	0.47	0.15	0.23	0.38	0.33
Gemini 3.1 Pro	0.91	**0.71**	**0.80**	0.50	**0.29**	**0.37**	**0.58**	**0.63**
Qwen 3.6 Plus	0.91	0.68	0.78	**0.51**	0.21	0.29	0.54	0.59
Claude Sonnet 4.6	0.93	0.38	0.54	0.34	0.21	0.26	0.40	0.35

Default judge and reporting. Unless otherwise specified, we use Robo-Dopamine (Forward) as the default progress judge for policy auditing, as it achieves the strongest overall performance on RoboPulse. The same judge and metric computation are used across downstream evaluations. Benchmark-provided outcome labels are used for success- and failure-conditioned analyses. OPD metrics are reported on a percentage scale for readability.

4.2 Reliability of Progress Judging (RQ1)

We first evaluate progress-judging reliability on RoboPulse and then examine whether the task progress curves produced by the selected judge remain consistent with benchmark-provided success rates in the downstream auditing settings.

Tab. 3 compares specialized PRMs and general-purpose VLMs on RoboPulse.

1) Specialized PRMs provide more reliable task-progress judgments than general-purpose VLMs. Robo-Dopamine (Forward) achieves a Macro-F1 of 0.77 and Accuracy of 0.84, compared with 0.58 and 0.63 for the strongest general-purpose VLM. Specialized PRMs therefore provide stronger overall task-progress judgments.

2) Progress judging remains challenging, particularly for Falling progress. Even for Robo-Dopamine (Forward), Falling F1 is 0.61 compared with 0.92 for Rising. The remaining gap shows that task-progress assessment is not fully solved even by the strongest specialized PRM.

Table 4: **Consistency between PRM-derived and benchmark success rates.** MAE and Mean Diff. are reported in percentage points.

Setting	# Units	MAE	Mean Diff.	Spearman ρ
RoboTwin 2.0	35	0.84	+0.18	0.98
RoboDojo-RealWorld	9	1.32	-0.35	0.96

Based on its overall RoboPulse performance, we use Robo-Dopamine (Forward) as the default progress judge for subsequent policy auditing. We further examine whether its task progress curves remain consistent with benchmark-provided success rates on RoboTwin 2.0 and RoboDojo-RealWorld. For each evaluation unit, we derive a PRM-based success rate from the fraction of rollouts whose predicted maximum progress reaches task completion, and compare it with the corresponding benchmark success rate using MAE, signed mean difference, and Spearman rank correlation. Tab. 4 summarizes the comparison.

3) The selected progress judge remains consistent with benchmark-provided success rates. The PRM-based success rates closely align with the benchmark values on both RoboTwin 2.0 and RoboDojo-RealWorld. This consistency supports using the resulting task progress curves for the subsequent process-level analyses in RQ2–RQ4.

Table 5: **Fine-grained policy evaluation on RoboTwin 2.0.** Five policy families are evaluated with 50 rollouts per task. SR denotes the official benchmark's binary success rate; the remaining columns report OPD metrics. All values are reported on a percentage scale. ↑ / ↓ indicate that higher / lower values are better. Full results are in App. D.

Task	Policy	Binary	Outcome					Process	Diagnosis	
		SR ↑	MC ↑				MP ↑	PPL ↑	CRA ↓	STR ↓
			@25	@50	@75	@100				
Blocks Ranking RGB	ACT	2.0	84.0	44.0	22.0	2.0	49.9	11.7	9.0	59.7
	DP	0.0	94.0	40.0	18.0	0.0	51.7	4.1	16.3	43.8
	RDT	0.0	100.0	62.0	30.0	0.0	61.2	6.2	16.3	39.0
	π_0	6.0	96.0	66.0	40.0	8.0	63.4	15.9	11.5	48.4
	OpenVLA-OFT	0.0	98.0	42.0	6.0	0.0	48.3	2.4	17.8	38.6
Handover Mic	ACT	72.0	100.0	100.0	94.0	74.0	96.8	72.3	4.1	44.1
	DP	44.0	100.0	94.0	88.0	44.0	93.8	66.0	5.5	57.2
	RDT	100.0	100.0	100.0	100.0	100.0	100.0	84.2	1.5	39.8
	π_0	98.0	100.0	100.0	100.0	98.0	99.4	88.1	1.0	42.7
	OpenVLA-OFT	78.0	100.0	100.0	100.0	76.0	94.2	66.2	5.7	45.1
Place Bread Basket	ACT	4.0	100.0	74.0	46.0	4.0	73.1	17.6	15.5	65.4
	DP	14.0	100.0	94.0	74.0	16.0	87.6	21.3	16.9	48.0
	RDT	10.0	100.0	100.0	78.0	8.0	90.4	16.6	22.7	37.1
	π_0	14.0	100.0	94.0	62.0	16.0	83.7	21.2	18.9	47.6
	OpenVLA-OFT	2.0	100.0	100.0	84.0	2.0	92.6	8.9	26.3	31.9

4.3 Stage-wise Reachability (RQ2)

We first examine the *Outcome* question: how far does the execution reach? Tab. 5 reports the benchmark success rate together with OPD metrics for three representative RoboTwin tasks. The full seven-task results are provided in App. D.

1) Milestone coverage localizes where progress stops. On *Blocks Ranking RGB*, MC@25 ranges from 84 to 100, whereas MC@75 ranges from only 6 to 40. Most policies therefore make early task progress but differ substantially in reaching later task stages.

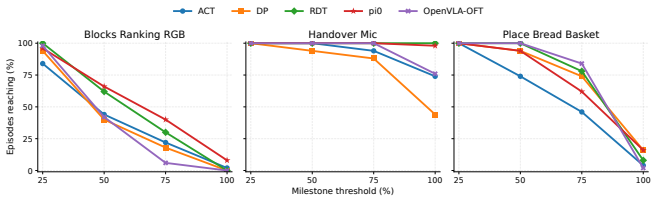


Figure 3: **Stage-wise reachability on RoboTwin 2.0.** We plot the fraction of rollouts reaching each progress milestone.

2) Similar success rates can conceal different stage-wise reachability. On *Blocks Ranking RGB*, DP, RDT, and OpenVLA-OFT all have a success rate of 0, while their MC@75 values are 18, 30, and 6, respectively. Milestone coverage therefore distinguishes how far executions progress despite the same success result.

3) Milestone profiles expose late-stage bottlenecks. On *Handover Mic*, DP reaches MC@75 of 88 but has a success rate of 44. The gap indicates that many executions reach later task stages without completing the task.

4.4 Execution Quality among Successful Rollouts (RQ3)

We next examine whether successful rollouts are equivalent in their execution process. Using the benchmark-provided success labels, we restrict the analysis to successful episodes and compare PPL, CRA, and STR.

1) Successful rollouts remain separable by process metrics. As shown in Fig. 4, policies occupy different regions in PPL–CRA and PPL–STR space even after conditioning on the same successful outcome. Thus, successful executions can differ in how directly they progress, how much they regress, and how often they stagnate.

2) Conditional execution quality and success frequency capture different behavior. A policy can produce relatively direct and stable trajectories among its successful rollouts while still achieving a lower overall benchmark success rate. Conversely, a policy with higher terminal success can exhibit more correction or stagnation during successful execution. Reporting the two together separates *how often* the task succeeds from *how* successful execution unfolds.

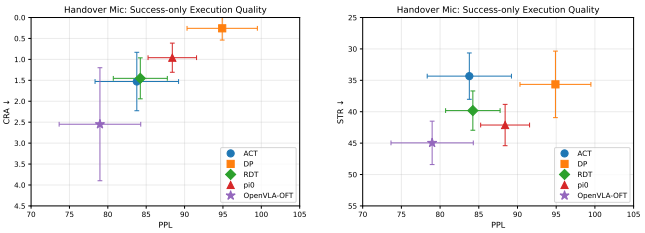


Figure 4: **Execution quality among successful rollouts on Handover Mic.** We compare success-conditioned PPL with CRA and STR. Error bars denote variation across successful episodes.

3) PPL, CRA, and STR provide complementary views of successful execution. PPL measures progress efficiency, CRA captures regression from the best-so-far state, and STR measures the fraction of execution with negligible progress. Their differences remain visible after terminal outcome is fixed.

4.5 Regression and Stagnation Patterns (RQ4)

We finally examine failed rollouts. Using the benchmark-provided failure labels, we aggregate MP, PPL, CRA, and STR over failed episodes. For visualization, each metric is normalized within a task across policy families; CRA and STR are sign-flipped before normalization so that higher values consistently correspond to more desirable behavior.

1) Failed rollouts exhibit different progress patterns. Some failures reach relatively high progress before losing previously achieved progress, producing high MP together with poor CRA. Others stop earlier and exhibit greater stagnation, producing lower MP together with high STR. Both receive the same failure label.

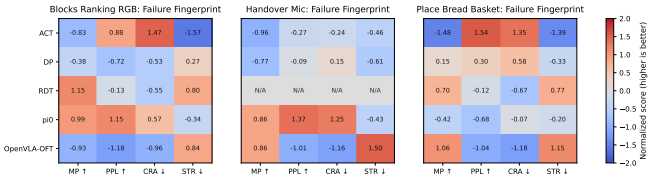


Figure 5: **Process patterns among failed rollouts.** We normalize failed-episode OPD metrics within each task to highlight differences across policy families.

2) Regression and stagnation capture different progress patterns. CRA increases when an execution remains below its previously reached progress, whereas STR increases when task progress changes little over time. The two metrics therefore distinguish regression from stagnation.

3) These patterns vary across policies and tasks. Fig. 5 shows different combinations of reached progress, progress efficiency, regression, and stagnation across policies and tasks. The resulting profiles provide a process-level description of failure without assigning a causal mechanism to the observed behavior.

4.6 Real-World Validation (RQ5)

We further evaluate PRM-as-a-Judge on RoboDojo-RealWorld (Chen et al., 2026b), which covers 18 physical manipulation tasks and nine embodied policies. We apply the same progress judge and OPD pipeline directly to the released rollout videos. Tab. 6 reports benchmark success rate together with model-level OPD results.

1) Failed real-world rollouts can still contain substantial task progress. Spirit v1.5 has a success rate of only 0.6%, yet 66.7% of its rollouts reach MC@25 and 29.2% reach MC@50. Thus, the failure result alone does not indicate how far a rollout progresses before execution ends.

Table 6: **Fine-grained auditing on RoboDojo-RealWorld.** SR is the benchmark-reported success rate; the remaining columns are computed from the PRM task progress curves. All values are reported in percentage scale. Best results are **bolded** and second-best results are underlined.

Model	Binary	Outcome					Process	Diagnosis	
	SR↑	MC@25↑	MC@50↑	MC@75↑	MC@100↑	MP↑	PPL↑	CRA↓	STR↓
$\pi_{0.5}$	**12.8**	**85.3**	**60.6**	**38.8**	**17.1**	**59.9**	**28.9**	**5.2**	**21.8**
InternVLA-A1	7.2	61.8	33.5	18.8	4.7	30.4	14.2	9.2	38.2
Xiaomi-Robotics-0	3.9	54.7	29.4	11.8	3.5	27.1	10.5	13.0	35.5
GalaxeaVLA (G0)	4.4	60.0	27.7	10.0	2.4	29.4	11.2	11.9	35.5
X-VLA	3.3	69.4	32.4	17.1	2.4	37.2	11.4	15.8	22.4
π_0	1.7	54.1	21.2	7.7	1.8	27.0	10.7	10.2	38.8
StarVLA-α	1.7	37.7	15.3	7.1	1.8	17.9	8.3	9.9	47.7
GR00T-N1.7	1.7	63.8	29.4	13.1	0.6	31.6	15.6	10.2	39.4
Spirit v1.5	0.6	66.7	29.2	7.5	0.0	29.7	11.7	15.2	26.9

2) Similar success rates can correspond to different process behavior. X-VLA and GalaxeaVLA have similar success rates and the same MC@100, but X-VLA reaches higher intermediate milestones. Their CRA and STR further distinguish regression and stagnation despite the similar success rates.

3) Outcome and process metrics provide complementary information. GR00T-N1.7 achieves the second-highest PPL despite a low success rate, whereas X-VLA has the second-highest MP but the highest CRA. These differences show that reached progress, progress efficiency, and regression are not determined by success rate alone.

Together, these results show that the process-level distinctions observed on RoboTwin 2.0 remain visible in physical robot rollouts, supporting PRM-as-a-Judge for real-world policy auditing.

5 Conclusion

We introduced **PRM-as-a-Judge**, a process-level evaluation paradigm that converts robot rollouts into task progress curves for policy auditing. RoboPulse shows that specialized PRM judges outperform general-purpose VLMs in task-progress assessment, while *Falling* progress remains more challenging. Using the strongest judge, we further show that PRM-based success rates remain consistent with benchmark-provided success rates on RoboTwin 2.0 and RoboDojo-RealWorld. Based on the resulting progress curves, the **OPD** metric system reveals stage-wise reachability, successful execution quality, and regression and stagnation patterns beyond binary success rate. Experiments in simulation and real-world rollouts demonstrate the applicability of PRM-as-a-Judge for process-level policy auditing.

Limitations and Societal Impact. PRM-as-a-Judge depends on the reliability of the underlying progress judge, and current judges remain imperfect, particularly for task-progress decrease. OPD characterizes regression and stagnation from the task progress curve but does not explain their semantic causes. Such interpretation could be further supported by semantic reasoning models such as VLMs. More detailed process-level evaluation can make robotic benchmarking more transparent by exposing execution differences hidden by binary success rate.

AI use statement

We used generative AI tools to assist with language polishing during manuscript preparation. This included correcting grammatical errors, revising awkward sentences, reducing repetitive wording, and improving transitions between paragraphs. We also used these tools to make technical descriptions clearer and terminology more consistent across the manuscript. The authors take full responsibility for the accuracy and content of the final manuscript.

REFERENCES

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

Anthropic. Introducing claude sonnet 4.6. `https://www.anthropic.com/news/claude-sonnet-4-6`, February 2026. Accessed: 2026-08-10.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Junhao Cai, Zetao Cai, Jiafei Cao, Yilun Chen, Zeyu He, Lei Jiang, Hang Li, Hengjie Li, Yang Li, Yufei Liu, et al. Internvla-a1: Unifying understanding, generation and action for robotic manipulation. *arXiv preprint arXiv:2601.02456*, 2026a.

Rui Cai, Jun Guo, Xinze He, Piaopiao Jin, Jie Li, Bingxuan Lin, Futeng Liu, Wei Liu, Fei Ma, Kun Ma, et al. Xiaomi-robotics-0: An open-sourced vision-language-action model with real-time execution. *arXiv preprint arXiv:2602.12684*, 2026b.

Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Yide Shentu, and Philipp Wu. Sarm: Stage-aware reward modeling for long horizon robot manipulation. *arXiv preprint arXiv:2509.25358*, 2025a.

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. *arXiv preprint arXiv:2602.19313*, 2026a.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025b.

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

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Google Deepmind. Gemini 3.1 pro best for complex tasks and bringing creative concepts to life, February 2026. URL `https://deepmind.google/models/gemini/pro/`.

Ramy ElMallah, Krish Chhajer, and Chi-Guhn Lee. Score the steps, not just the goal: Vlm-based subgoal evaluation for robotic manipulation. *arXiv preprint arXiv:2509.19524*, 2025.

Galaxea Team. Galaxea g0: Open-world dataset and dual-system vla model. *arXiv preprint arXiv:2509.00576*, 2025.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. Maniskill2: A unified benchmark for generalizable manipulation skills. *arXiv preprint arXiv:2302.04659*, 2023.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

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

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, Kai Shu, Lu Cheng, and Huan Liu. From generation to judgment: Opportunities and challenges of llm-as-a-judge. *arXiv preprint arXiv: 2411.16594*, 2024a.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024b.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`, 5 2023.

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. *ArXiv*, 2026.

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

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

Yibin Liu et al. From passive observer to active critic: Reinforcement learning elicits process reasoning for robotic manipulation. *arXiv preprint arXiv:2603.15600*, 2026.

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

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

Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, et al. Vision language models are in-context value learners. In *The Thirteenth International Conference on Learning Representations*, 2024.

Yunchao Ma, Ze Chen, Erjin Zhou, Ziming Liu, Haowei Zhang, Kai Liu, and Haoqiang Fan. Table30 v2: Evaluating generalized models by real robots at scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 4461–4467, June 2026.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

NVIDIA. About nvidia isaac gr00t n1.7 - a foundation model for generalist robots, March 2026. URL `https://developer.nvidia.com/isaac/gr00t`.

OpenAI. Introducing gpt-5.4, March 2026. URL `https://openai.com/zh-Hans-CN/index/introducing-gpt-5-4/`.

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. *ArXiv*, 2025.

Qwen Team. Qwen3.6-Plus: Towards real world agents, April 2026. URL `https://qwen.ai/blog?id=qwen3.6`.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 1134–1141. IEEE, 2018.

Spirit AI Team. Spirit-v1.5: Clean data is the enemy of great robot foundation models, January 2026. URL `https://www.spirit-ai.com/en/blog/spirit-v1-5`.

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

Yi Ru Wang, Carter Ung, Grant Tannert, Jiafei Duan, Josephine Li, Amy Le, Rishabh Oswal, Markus Grotz, Wilbert Pumacay, Yuquan Deng, et al. Roboeval: Where robotic manipulation meets structured and scalable evaluation. *arXiv preprint arXiv:2507.00435*, 2025.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. 2024.

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. *arXiv preprint arXiv:2603.16065*, 2026.

Adina Yakefu, Bin Xie, Chongyang Xu, Enwen Zhang, Erjin Zhou, Fan Jia, Haitao Yang, Haoqiang Fan, Haowei Zhang, Hongyang Peng, et al. Robochallenge: Large-scale real-robot evaluation of embodied policies. *arXiv preprint arXiv:2510.17950*, 2025.

Jinhui Ye, Ning Gao, Senqiao Yang, Jinliang Zheng, Zixuan Wang, Yuxin Chen, Pengguang Chen, Yilun Chen, Shu Liu, and Jiaya Jia. StarVLA-α: Reducing complexity in vision-language-action systems. *arXiv preprint arXiv:2604.11757*, 2026.

Zewei Ye, Weifeng Lu, Minghao Ye, Tao Lin, Shuo Yang, Junchi Yan, and Bo Zhao. Robofac: A comprehensive framework for robotic failure analysis and correction. *arXiv preprint arXiv:2505.12224*, 2025.

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

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

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

APPENDIX OVERVIEW

This appendix provides additional details and results for **PRM-as-a-Judge**. The sections are organized as follows:

- **App. A:** Reviews prior work on robotic evaluation, process reward models, and automated judges, and clarifies the positioning of PRM-as-a-Judge relative to state-based and task-specific evaluation frameworks.
- **App. B:** Provides implementation details for task progress curve construction and the OPD metrics, together with sensitivity analysis of the stagnation threshold.
- **App. C:** Describes the RoboPulse data composition, interval annotation protocol, progress-judge evaluation procedure, and additional results.
- **App. D:** Provides complete policy-auditing results on all seven RoboTwin 2.0 tasks, including stage-wise reachability, success-conditioned execution quality, and failure patterns.
- **App. E:** Reports the computational cost of representative progress judges.
- **App. F:** Provides qualitative examples that align rollout observations with task progress curves and OPD measurements.

A RELATED WORK

A.1 EVALUATION METRICS AND PARADIGMS IN ROBOTICS

Robotic manipulation is commonly evaluated using binary task success, as adopted by many widely used benchmarks and simulation suites (Li et al., 2024b; James et al., 2020; Mees et al., 2022). Beyond terminal success, prior work has introduced richer forms of evaluation. Rule-based methods use predefined stages, subgoals, or manually specified criteria to assess intermediate execution (ElMallah et al., 2025). Simulator- or state-based methods further measure properties such as task progress, path efficiency, coordination, stability, contact, or mechanical effort from robot and environment states (Anderson et al., 2018; Zhu et al., 2020; Gu et al., 2023; Wang et al., 2025). These approaches provide substantially richer information than binary success when the required states and task-specific evaluation logic are available.

How PRM-as-a-Judge differs. PRM-as-a-Judge performs process-level evaluation directly from task instructions and recorded rollout observations. As shown in Tab. 7, it does not require privileged environment states or manually specified stages and process rules. The same evaluation pipeline can therefore be reused across tasks, policies, simulation, and real-world rollouts with substantially less task-specific setup. Its reliability naturally depends on the underlying progress judge; we therefore introduce RoboPulse to evaluate this capability independently before policy auditing.

Table 7: **Comparison of robotic evaluation paradigms.** ✓: supported; △: limited or task-dependent; –: not generally supported.

Paradigm	Process-level	Task Setup	Observation-only	Cross-task Reuse
Success Rate	–	Success criterion	△	△
Rule-/Subgoal-based	△	Rules / stages	△	△
State-based Process	✓	States / rules	–	△
PRM-as-a-Judge	✓	Task instruction	✓	✓

A.2 PROCESS REWARD MODELS IN ROBOTICS

Process Reward Models (PRMs) provide task-conditioned intermediate signals along robotic trajectories and have primarily been studied as reward or value functions for robot learning (Sermanet et al., 2018; Ma et al., 2022; 2023; Chen et al., 2025a). Recent progress models further learn task progress or reward signals from robotic observations and trajectories, including both local comparison and

longer-context formulations (Zhai et al., 2025; Tan et al., 2025; Ma et al., 2024). These models provide dense signals that reflect how task-relevant states evolve throughout an execution.

How PRM-as-a-Judge differs. PRM-as-a-Judge repurposes PRMs as progress judges for post-hoc process-level evaluation. The judge converts a rollout into a task progress curve, from which OPD measures stage-wise reachability, progress efficiency, regression, and stagnation. Because downstream auditing depends on the quality of this progress signal, we evaluate candidate progress judges independently on RoboPulse before applying them to policy auditing.

A.3 Foundation Models as Automated Judges

Foundation models have been increasingly used as automated evaluators, particularly through LLM-as-a-Judge frameworks for natural-language generation (Zheng et al., 2023; Li et al., 2023; Wang et al., 2024; Li et al., 2024a). With multimodal models, the same idea can be extended to visual and robotic execution: a general-purpose VLM can inspect task instructions and observations and judge whether an execution is making progress. Such models provide a flexible evaluator without requiring task-specific training.

Relation to PRM-as-a-Judge. General-purpose VLMs can serve as automated judges from task instructions and visual observations. PRM-as-a-Judge uses trajectory-trained PRMs to provide progress judgments grounded in robotic execution. RoboPulse directly compares these two classes of judges on task-progress increase and decrease, providing an independent evaluation of progress-judging reliability before downstream policy auditing.

B OPD Implementation Details

The definitions of MC, MP, PPL, CRA, and STR are provided in Sec. 2.2. Here we specify the implementation and reporting conventions used throughout the experiments.

B.1 Task Progress Curve

For each rollout, the outputs of the progress judge are converted into a normalized task progress curve

$$p_{0:T} = (p_0, p_1, \ldots, p_T), \qquad p_t \in [0, 1], \tag{10}$$

where larger values indicate greater task progress. We set the initial progress to $p_0 = 0$ and apply the same progress-curve construction procedure to all rollouts evaluated by the same judge. All OPD metrics are computed from this common representation.

B.2 Metric Computation and Reporting

OPD metrics are first computed independently for each rollout. MC@q is reported as the percentage of rollouts whose maximum progress reaches milestone $q \in \{25, 50, 75, 100\}$. MP, PPL, CRA, and STR are then aggregated over the rollout set considered by the corresponding analysis.

For success-conditioned and failure-conditioned analyses, rollout subsets are determined using the benchmark-provided binary outcome labels rather than predictions from the progress judge. This separates the official task outcome from the process measurements derived from the task progress curve.

Benchmark SR and MC@100. We report benchmark Success Rate (SR) and MC@100 separately. SR is the task outcome provided by the original benchmark, whereas MC@100 indicates whether the progress curve reaches the 100% milestone. The former serves as the benchmark reference, while the latter is an outcome-level measurement derived from the progress judge. Their consistency is evaluated separately in Sec. 4.2.

Percentage scale. For readability, OPD values in policy-auditing experiments are multiplied by 100 unless otherwise specified. Thus, MP, PPL, CRA, and STR are reported on a 0–100 scale, while MC@q is reported as the percentage of rollouts reaching the corresponding milestone.

B.3 Sensitivity to the Stagnation Threshold

Among the OPD metrics, STR is the only metric that uses a behavior-sensitive threshold. MP and CRA are threshold-free, while the values of q in MC@q specify reported milestones rather than tuned hyperparameters. We use $\epsilon = 0.5\%$ as the default threshold for negligible task-progress change.

To examine sensitivity to this choice, we sweep

$$\epsilon \in \{0.1\%, 0.25\%, 0.5\%, 0.75\%, 1.0\%\}.$$

Across the complete sweep, STR changes by at most 0.4 percentage points. The policy ordering remains unchanged, with Spearman rank correlation $\rho = 1.0$ across the tested thresholds. The comparative STR results are therefore insensitive to the specific value of ϵ within this range.

Table 8: **Sensitivity of STR to the stagnation threshold ϵ.** The reported STR values remain stable over a $10\times$ range of thresholds.

Policy	$\epsilon = 0.1\%$	$\epsilon = 0.25\%$	$\epsilon = 0.5\%$	$\epsilon = 0.75\%$	$\epsilon = 1.0\%$
OpenVLA-OFT	44.6	44.7	44.7	44.9	45.0
RDT	54.2	54.2	54.2	54.3	54.3
π_0	54.4	54.4	54.5	54.6	54.7
ACT	60.6	60.7	60.7	60.8	60.9
DP	62.0	62.0	62.1	62.2	62.3

C RoboPulse Benchmark Details

This section provides additional details of RoboPulse, including its data sources, annotation protocol, evaluation implementation, annotation quality, and domain-wise results. Dataset statistics are reported in Tab. 2 of the main paper.

C.1 Data Sources and Coverage

RoboPulse combines robot rollouts from both real-world and simulation environments. The real-world data are drawn from RoboChallenge-Table30 (Yakefu et al., 2025), RoboChallenge-Table30-V2 (Ma et al., 2026), ViFailback (Zeng et al., 2026), and RoboFAC-Real (Ye et al., 2025). The simulation data are drawn from LIBERO (Liu et al., 2023), RoboCasa (Nasiriany et al., 2024), SimplerEnv (Li et al., 2024b), RoboTwin 2.0 (Chen et al., 2025b), and RoboFAC-Sim (Ye et al., 2025).

These sources cover diverse robot embodiments, scenes, camera viewpoints, and manipulation tasks. RoboPulse uses the task instruction and recorded rollout observations as the common evaluation interface across all data sources.

C.2 Annotation Protocol

Three adult annotators with bachelor's degrees participated in the annotation process. For each trajectory, annotators are provided with the complete rollout together with its natural-language task instruction.

Annotators first review the complete execution and then identify temporally contiguous intervals exhibiting a clear change in task progress. Each labeled interval is assigned one of two labels, following the definitions in Sec. 3.2: *Rising* indicates an increase in task progress toward the goal, while *Falling* indicates a decrease in task progress or the reversal of previously achieved progress. Periods with negligible or ambiguous task-progress change are left unlabeled and excluded from RoboPulse evaluation.

Interval boundaries mark the beginning or end of a clearly directional task-progress segment, or a change in task-progress direction. Annotations are based on task-relevant state evolution rather than robot motion itself. For example, robot motion without a meaningful change in the task state is left unlabeled rather than treated as task-progress increase.

Annotators use the complete trajectory as temporal context when determining both interval boundaries and labels. This allows local observations to be interpreted with respect to the overall execution rather than in isolation.

Table 10: **Progress-judge evaluation on the real-world subset of RoboPulse.** We report class-wise Precision, Recall, and F1, together with overall Macro-F1 and Accuracy.

Judge Model	*Rising*			*Falling*			*Overall*	
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Macro-F1**	**Accuracy**
Specialized PRMs								
Robo-Dopamine (Forward)	**0.93**	**0.91**	**0.92**	**0.81**	0.55	0.65	**0.78**	**0.84**
Robo-Dopamine (Backward)	**0.93**	0.86	0.89	0.72	**0.60**	**0.66**	0.77	0.81
Robo-Dopamine (Incremental)	**0.93**	0.83	0.88	0.64	0.59	0.61	0.75	0.79
RoboMeter	0.89	0.86	0.88	0.49	0.57	0.53	0.70	0.81
LRM (Temporal)	0.84	0.78	0.81	0.32	0.34	0.33	0.57	0.70
LRM (Absolute)	0.92	0.35	0.51	0.48	0.23	0.31	0.41	0.33
TOPReward	0.84	0.75	0.79	0.27	0.36	0.31	0.55	0.68
VLAC	0.89	0.42	0.57	0.33	0.24	0.27	0.42	0.39
GVL	0.91	0.78	0.84	0.57	0.49	0.53	0.69	0.73
PRIMO	0.84	0.81	0.82	0.17	0.04	0.06	0.44	0.66
General-Purpose VLMs								
GPT-5.4	**0.93**	0.38	0.54	0.51	0.21	0.30	0.42	0.34
Gemini 3.1 Pro	0.92	0.63	0.75	0.51	**0.31**	**0.39**	**0.57**	0.57
Qwen 3.6 Plus	0.90	**0.66**	**0.76**	**0.54**	0.28	0.37	**0.57**	**0.59**
Claude Sonnet 4.6	0.92	0.36	0.52	0.36	0.29	0.32	0.42	0.35

C.3 Annotation Quality

We evaluate annotation consistency using trajectories independently annotated by all three annotators. We report raw agreement, Fleiss' κ, and the agreement between the majority-vote annotation and the final benchmark label.

The annotations show strong inter-annotator consistency, with a raw agreement of 89.6% and a Fleiss' κ of 0.78. The majority-vote labels agree with the final benchmark labels in 97.1% of the cases.

Table 9: **Annotation quality of RoboPulse.** We report agreement among the three annotators and consistency between majority-vote annotations and the final benchmark labels.

Metric	**Value**
Raw agreement	89.6%
Fleiss' κ	0.78
Majority vote vs. final label	97.1%

C.4 Evaluation Implementation

All rollout videos are uniformly sampled at 1 FPS. Each progress judge receives the task instruction and the sampled rollout observations.

Different judges expose different native progress representations. For judges that predict absolute task progress p_t, we obtain temporal progress changes using

$$d_t = p_t - p_{t-1}. \tag{11}$$

Judges that directly estimate task-progress changes use their native outputs, while categorical predictions are mapped to the common *Rising*/*Falling* label space.

For frame-wise progress estimates, we apply a 5-frame causal moving average before computing temporal progress changes. To reduce ambiguity around human-annotated transition boundaries, predictions from the first and last two sampled frames of each annotated interval are excluded from evaluation.

The remaining predictions are pooled across intervals. We report class-wise Precision, Recall, and F1 for *Rising* and *Falling*, together with Macro-F1 and Accuracy.

C.5 Domain-wise Results

We further evaluate progress judges separately on the real-world and simulation subsets of RoboPulse. Tabs. 10 and 11 report the complete class-wise results.

Table 11: **Progress-judge evaluation on the simulation subset of RoboPulse.** We report class-wise Precision, Recall, and F1, together with overall Macro-F1 and Accuracy.

Judge Model	*Rising*			*Falling*			*Overall*	
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Macro-F1**	**Accuracy**
Specialized PRMs								
Robo-Dopamine (Forward)	0.90	**0.93**	**0.92**	**0.76**	0.42	0.54	0.73	**0.84**
Robo-Dopamine (Backward)	0.91	0.92	0.91	0.72	0.46	**0.56**	**0.74**	0.83
Robo-Dopamine (Incremental)	**0.93**	0.85	0.89	0.57	0.54	0.55	0.72	0.79
RoboMeter	0.88	0.86	0.87	0.43	0.49	0.46	0.67	0.79
LRM (Temporal)	0.86	0.59	0.70	0.27	**0.56**	0.36	0.53	0.58
LRM (Absolute)	0.85	0.24	0.37	0.21	0.17	0.19	0.28	0.23
TOPReward	0.86	0.75	0.80	0.30	0.42	0.35	0.58	0.69
VLAC	0.90	0.32	0.47	0.18	0.19	0.18	0.32	0.29
GVL	0.88	0.78	0.83	0.41	0.44	0.42	0.63	0.71
PRIMO	0.90	0.64	0.75	0.26	0.07	0.11	0.43	0.54
General-Purpose VLMs								
GPT-5.4	**0.96**	0.36	0.53	0.30	0.04	0.07	0.30	0.30
Gemini 3.1 Pro	0.89	**0.86**	**0.88**	**0.47**	**0.26**	**0.34**	**0.61**	**0.75**
Qwen 3.6 Plus	0.93	0.72	0.81	0.34	0.07	0.12	0.47	0.60
Claude Sonnet 4.6	**0.96**	0.42	0.59	0.22	0.06	0.10	0.34	0.36

Across the real-world subset, Robo-Dopamine (Forward) achieves a Macro-F1 of 0.78 and an Accuracy of 0.84, while the strongest general-purpose VLM reaches 0.57 Macro-F1 and 0.59 Accuracy. On the simulation subset, the strongest specialized PRM reaches 0.74 Macro-F1 and 0.84 Accuracy, compared with 0.61 and 0.75 for the strongest general-purpose VLM. The advantage of specialized PRMs therefore remains visible in both domains.

Across both subsets, *Falling* progress is consistently more difficult to identify than *Rising* progress, particularly for general-purpose VLMs. This pattern is consistent with the overall RoboPulse results in the main paper.

D Additional Policy Auditing Results

This section provides complete policy-auditing results on RoboTwin 2.0.

We report OPD results for five policy families, including *ACT*, *DP*, *RDT*, π_0, and *OpenVLA-OFT*, across all seven RoboTwin 2.0 tasks. Tab. 12 provides the complete per-task results, while Figs. 6–7 summarize stage-wise reachability, success-conditioned execution quality, and failure patterns.

Across tasks, the complete results show the same process-level distinctions discussed in the main paper. Milestone coverage separates early-stage reachability from terminal completion; PPL distinguishes executions with similar reached progress but different progress efficiency; and CRA and STR capture different regression and stagnation patterns. N/A indicates that too few episodes are available for a stable estimate.

Table 12: **Process-level auditing on RoboTwin 2.0.** Five policy families are evaluated with 50 rollouts per task. SR denotes the binary success rate; the remaining columns report OPD metrics. All values are reported on a percentage scale. ↑ / ↓ indicate that higher / lower values are better.

Task	Policy	Binary	Outcome					Process	Diagnosis	
		SR↑	MC↑				MP↑	PPL↑	CRA↓	STR↓
			@25	@50	@75	@100				
Blocks Ranking RGB	ACT	2.0	84.0	44.0	22.0	2.0	49.9	11.7	9.0	59.7
	DP	0.0	94.0	40.0	18.0	0.0	51.7	4.1	16.3	43.8
	RDT	0.0	100.0	62.0	30.0	0.0	61.2	6.2	16.3	39.0
	π_0	6.0	96.0	66.0	40.0	8.0	63.4	15.9	11.5	48.4
	OpenVLA-OFT	0.0	98.0	42.0	6.0	0.0	48.3	2.4	17.8	38.6
Handover Block	ACT	42.0	86.0	60.0	44.0	42.0	66.4	48.0	9.6	65.5
	DP	44.0	92.0	52.0	50.0	44.0	66.9	62.2	1.1	69.6
	RDT	38.0	94.0	82.0	62.0	38.0	78.9	53.1	9.9	65.1
	π_0	40.0	84.0	58.0	50.0	40.0	68.2	43.5	8.6	53.5
	OpenVLA-OFT	2.0	84.0	44.0	36.0	2.0	56.4	4.7	18.6	55.7
Handover Mic	ACT	72.0	100.0	100.0	94.0	74.0	96.8	72.3	4.1	44.1
	DP	44.0	100.0	94.0	88.0	44.0	93.8	66.0	5.5	57.2
	RDT	100.0	100.0	100.0	100.0	100.0	100.0	84.2	1.5	39.8
	π_0	98.0	100.0	100.0	100.0	98.0	99.4	88.1	1.0	42.7
	OpenVLA-OFT	78.0	100.0	100.0	100.0	76.0	94.2	66.2	5.7	45.1
Hanging Mug	ACT	14.0	96.0	84.0	74.0	14.0	84.9	23.6	17.1	51.0
	DP	14.0	100.0	100.0	98.0	14.0	86.4	30.8	18.7	57.0
	RDT	20.0	98.0	96.0	92.0	20.0	88.5	48.4	11.4	65.8
	π_0	16.0	100.0	96.0	92.0	16.0	95.4	50.3	8.5	60.9
	OpenVLA-OFT	8.0	100.0	92.0	88.0	8.0	82.3	18.9	21.2	40.9
Place Bread Basket	ACT	4.0	100.0	74.0	46.0	4.0	73.1	17.6	15.5	65.4
	DP	14.0	100.0	94.0	74.0	16.0	87.6	21.3	16.9	48.0
	RDT	10.0	100.0	100.0	78.0	8.0	90.4	16.6	22.7	37.1
	π_0	14.0	100.0	94.0	62.0	16.0	83.7	21.2	18.9	47.6
	OpenVLA-OFT	2.0	100.0	100.0	84.0	2.0	92.6	8.9	26.3	31.9
Place Bread Skillet	ACT	8.0	100.0	80.0	34.0	8.0	68.7	34.2	8.0	69.1
	DP	4.0	100.0	58.0	30.0	4.0	62.5	31.5	8.4	78.1
	RDT	6.0	100.0	80.0	50.0	6.0	76.4	27.9	12.6	63.7
	π_0	16.0	100.0	80.0	54.0	16.0	77.9	32.4	11.8	61.3
	OpenVLA-OFT	10.0	100.0	60.0	26.0	10.0	63.4	14.2	13.7	30.7
Place Can Basket	ACT	4.0	98.0	88.0	74.0	4.0	83.8	30.6	9.1	69.9
	DP	12.0	88.0	60.0	34.0	12.0	60.9	39.9	5.8	81.0
	RDT	16.0	100.0	100.0	94.0	16.0	95.9	52.4	4.6	69.0
	π_0	28.0	100.0	94.0	74.0	28.0	87.4	45.9	8.0	67.1
	OpenVLA-OFT	8.0	100.0	98.0	78.0	8.0	84.0	34.5	6.9	69.6

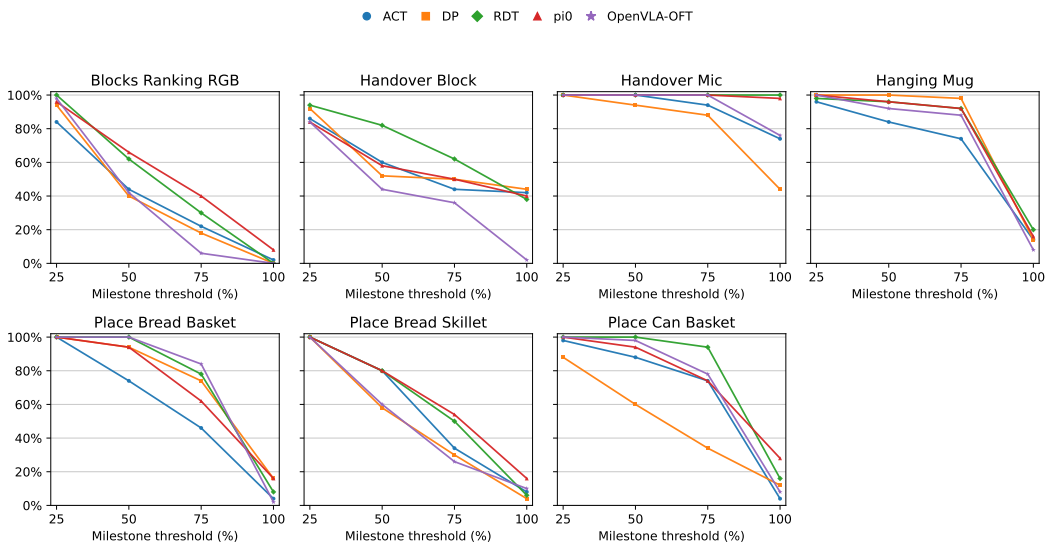


Figure 6: **Reachability profiles over all RoboTwin 2.0 tasks.** For each task, we plot the fraction of rollouts reaching milestone thresholds at 25/50/75/100%.

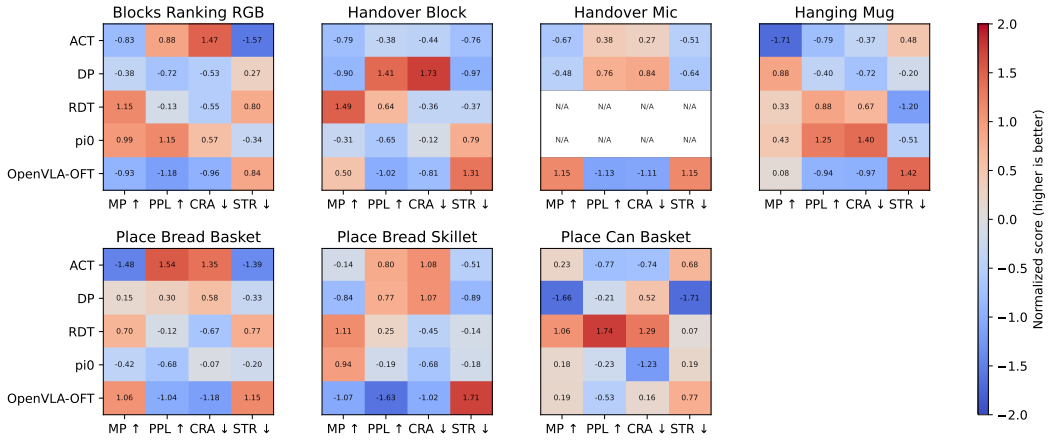


Figure 7: **Process patterns among failed rollouts over all RoboTwin 2.0 tasks.** MP, PPL, CRA, and STR are aggregated over failed episodes and normalized within each task. CRA and STR are sign-flipped before normalization so that higher normalized values consistently correspond to more desirable process characteristics.

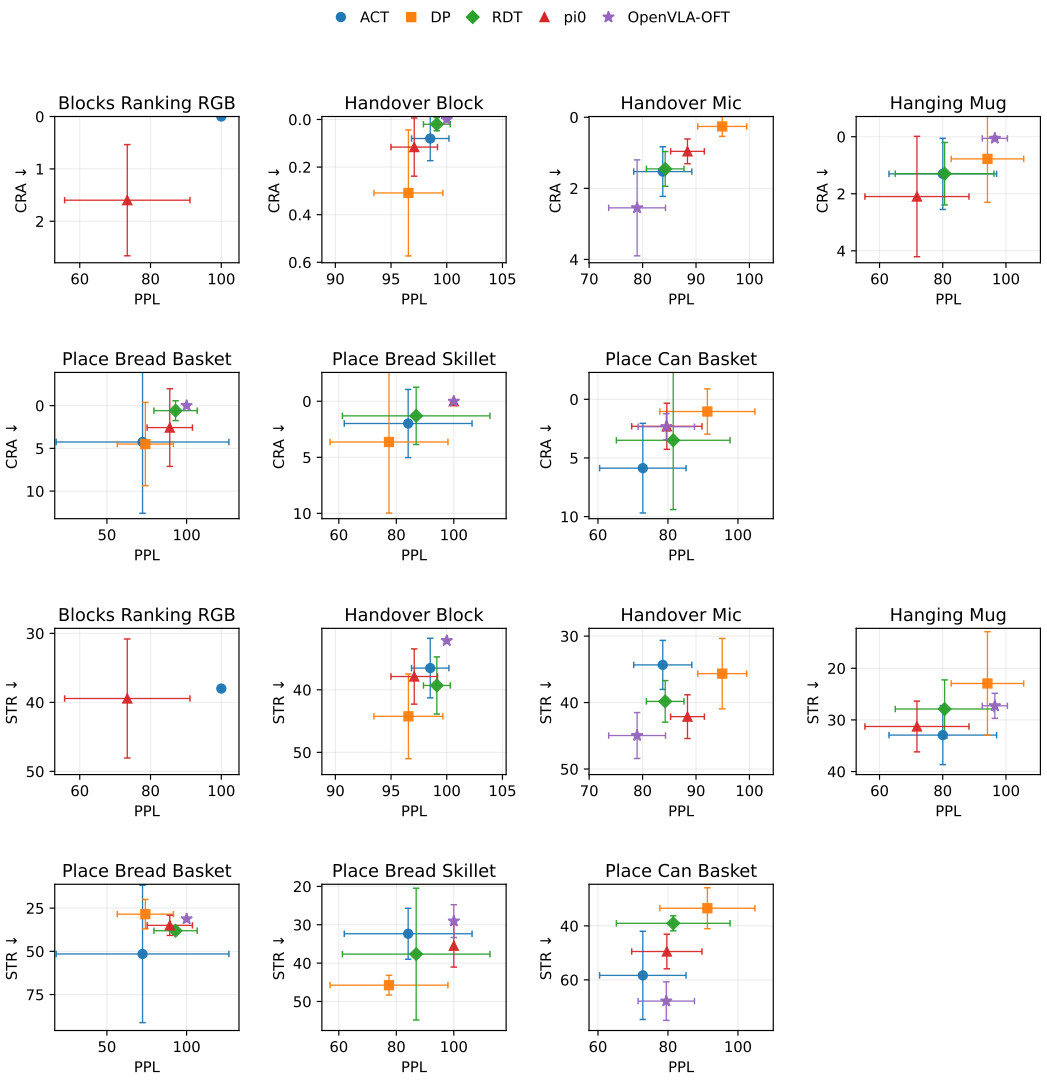


Figure 8: **Success-conditioned execution quality over all RoboTwin 2.0 tasks.** PPL, CRA, and STR are computed over successful episodes only. Error bars denote variation across successful rollouts.

Table 13: **Representative progress-judge inference cost on 100 rollouts.** Local models are evaluated on a single NVIDIA H100 GPU; GVL is accessed through its API. Time is measured over 100 rollouts.

Judge	Batch Size	Peak Memory	Time
VLAC-2B	1	6 GB	50 min 42 s
GVL	1	–	29 min 42 s
Robo-Dopamine-3B	1	12 GB	53 min 48 s
Robo-Dopamine-3B	10	72 GB	6 min 12 s

E Computational Resources

All locally executed experiments were conducted on internal GPU servers equipped with NVIDIA H100 GPUs. For RoboPulse, we used up to 8 H100 GPUs for parallel inference with the evaluated progress judges when local execution was available. All judges were evaluated zero-shot without task-specific fine-tuning.

For RoboTwin 2.0 policy auditing, we used up to 8 H100 GPUs to reproduce and evaluate ACT, DP, RDT, π_0, and OpenVLA-OFT following their official training and evaluation configurations. All policies used the same data budget and were evaluated with 50 rollouts per policy-task, as described in Sec. 4.1. The recorded rollouts were then scored post hoc by Robo-Dopamine to construct task progress curves and compute OPD metrics.

Progress-judge inference cost. We benchmark representative progress judges on 100 rollouts uniformly sampled at 1 FPS, with 30.6 sampled frames per rollout on average in Tab. 13. Local models are evaluated on a single NVIDIA H100 GPU, while GVL is accessed through its API. Wall-clock time includes video preprocessing and progress-judge inference.

Once the task progress curves are available, OPD computation is negligible. Computing all OPD metrics for 100 rollouts takes 14.0 ms; including JSON loading, aggregation, and output writing, the complete post-processing pipeline takes 0.50 s. Thus, the evaluation cost is dominated by progress-judge inference rather than OPD post-processing.

The main computational cost of the overall experiments comes from RoboTwin policy reproduction and rollout generation, followed by post-hoc progress-judge inference. No additional large-scale pretraining was performed in this work.

F Qualitative Visualization

We visualize representative task progress curves produced by Robo-Dopamine across real-world, human, and simulation trajectories. Figs. 9–14 align the predicted progress scores with representative events from each rollout. For each episode, we additionally report the corresponding OPD metrics, including MC@100, MP, PPL, CRA, and STR.

Task-progress regression. The task progress curves capture non-monotonic execution behavior. In *Stack the wooden blocks* (Fig. 9) and *Clean the table* (Fig. 10), progress decreases when previously achieved task progress is partially reversed, such as after an incorrect block interaction or disruption of an already organized scene. These decreases are reflected by CRA, which summarizes how far and how long the rollout remains below its best-so-far progress.

Stagnation. The curves also expose periods with little task-progress change. In *Blocks Ranking RGB* (Fig. 12), repeated empty grasps and incorrect block interactions correspond to an extended low-progress region before later advancement. The resulting STR of 16.22% summarizes the fraction of the rollout with negligible progress change.

Recovery and completion. The visualizations further show that progress can decrease and subsequently recover within the same rollout. All examples eventually reach task completion, with MC@100 and MP reaching 100%. PPL distinguishes how directly this progress is accumulated. For

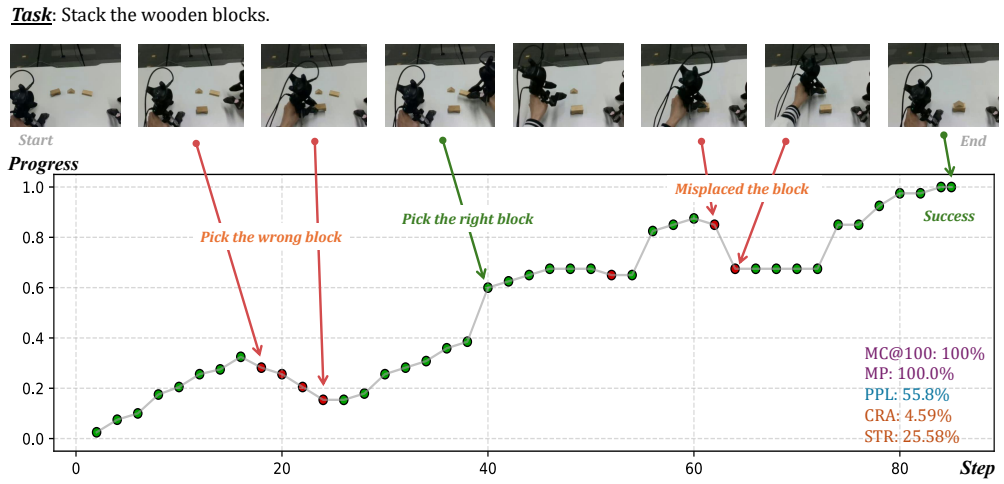


Figure 9: **Real-World Task: Stack the wooden blocks.** The task progress curve decreases after incorrect block interactions and subsequently recovers as the execution advances toward completion. The resulting CRA of 4.59% summarizes regression from the best-so-far progress.

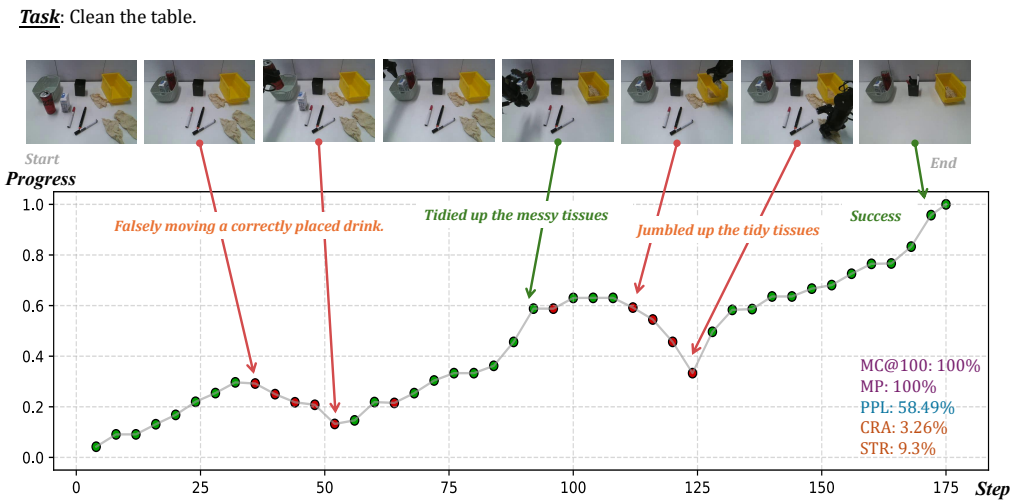


Figure 10: **Real-World Task: Clean the table.** Moving an already placed drink and disrupting the organized tissues correspond to decreases in task progress, followed by recovery and final completion. The episode has PPL 58.49% and CRA 3.26%.

example, *Place bread into the basket* (Fig. 13) achieves a PPL of 75.06%, with only limited variation in the task progress curve before completion.

Overall, these examples illustrate how the task progress curve and OPD provide complementary descriptions of reached progress, progress efficiency, regression, and stagnation at the episode level.

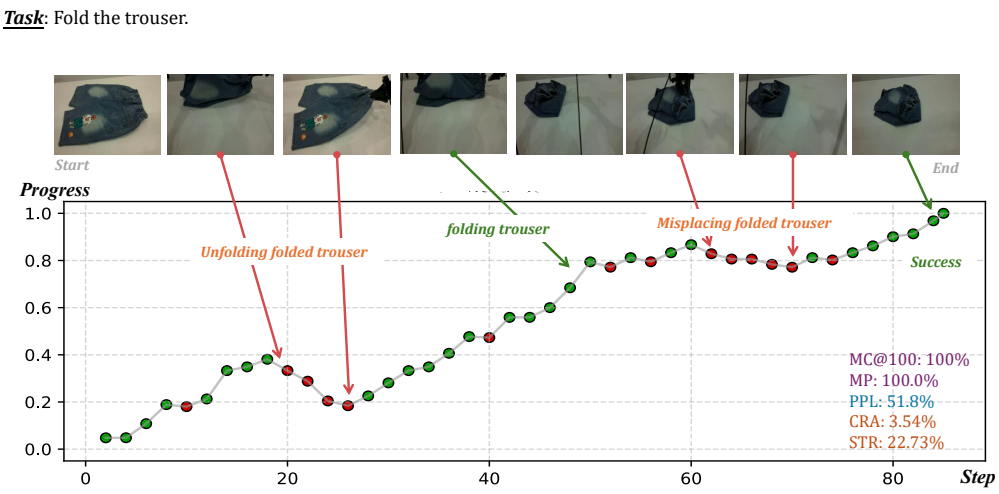


Figure 11: **Human Task: Fold the trouser.** The task progress curve decreases when the trouser is unfolded or the folded trouser is displaced, and increases again as the desired folded state is restored.

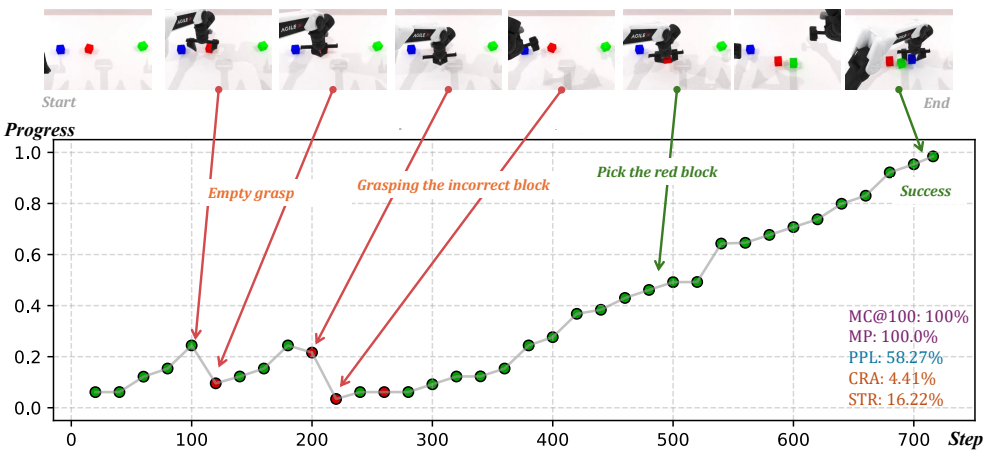


Figure 12: **Simulation Task: Blocks Ranking RGB.** Repeated empty grasps and incorrect block interactions correspond to an extended low-progress region before later task advancement. The episode has an STR of 16.22%.

Task: Place bread into the basket.

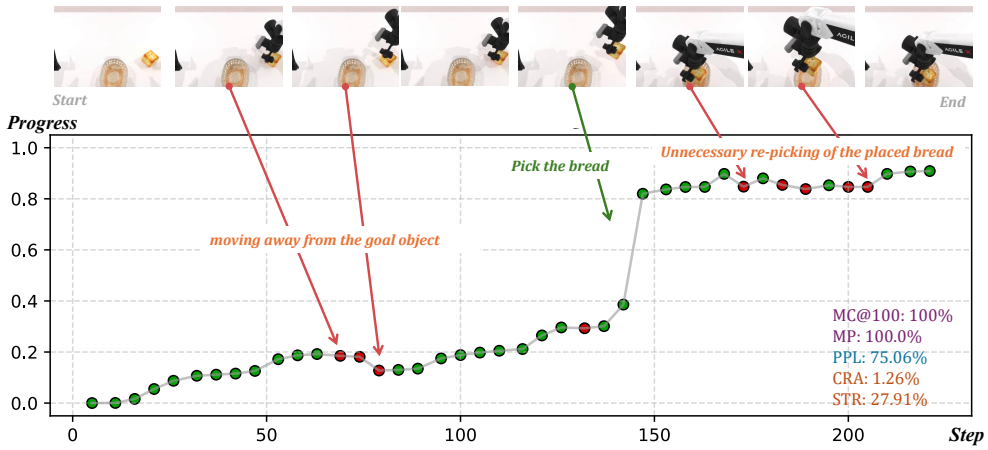


Figure 13: **Simulation Task: Place bread into the basket.** The curve briefly decreases when the bread moves away from the target and then rises as the task proceeds toward completion. The episode achieves a PPL of 75.06%.

Task: Hanging the mug.

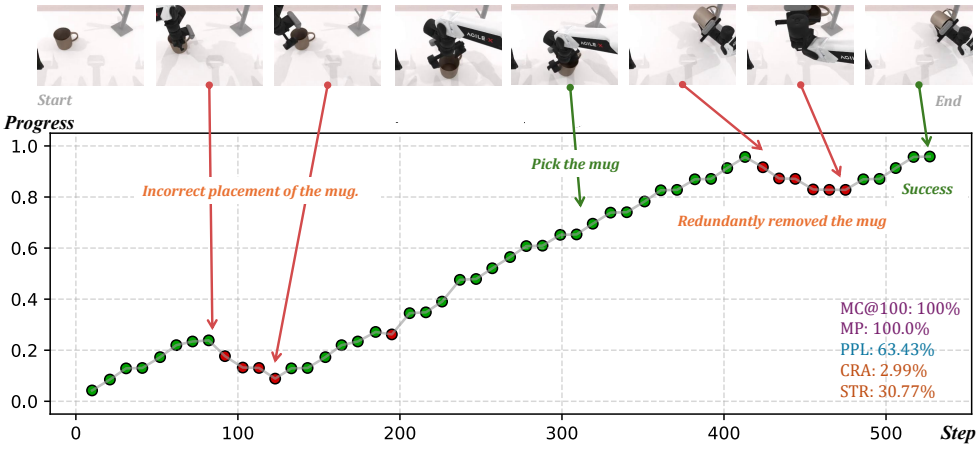


Figure 14: **Simulation Task: Hanging the mug.** The curve captures a decrease in task progress when the mug is removed after being partially placed, followed by renewed progress toward final completion.