Robot Critics that Sweat the Small Stuff

Sruthi Sudhakar[1] **Junbang Liang**[1] **Sreehari Rammohan**[1]
Pavel Tokmakov[2] **Richard Zemel**[1] **Carl Vondrick**[1]

[1]Columbia University [2]Toyota Research Institute

robocritic.cs.columbia.edu

Abstract: Large vision-language models contain several priors about the world and object interactions, making them useful critics during inference to steer robot policies towards success. However, closed-loop robot manipulation requires judging small visual differences between success and failure, which remains a challenge for current VLMs. We introduce a method to fine-tune critics by constructing pairwise progress supervision using success and failure rollouts obtained from a policy. Our fine-tuned critic excels at fine-grained progress reasoning and subtle failure detection, outperforming prior progress reasoning baselines. Additionally, we use an action-conditioned video model to predict the visual effect of several candidate actions sampled from a policy, and show that our critic can correctly identify successful candidates to execute, improving the average policy success rate by 11% across real-world tasks and 5.9% across simulation tasks.

Keywords: Vision Language Models, Imitation Learning, Video Generation

1 Introduction

Humans rely on strong visual reasoning to decide which actions will complete a task and which will lead to failure. Recent advances in vision-language foundation models (VLMs) [1, 2, 3] show that these models encode extensive knowledge about the physical world, providing hope of using them for robot policies. One promising approach is to use VLMs as critics in the policy execution loop to reason over multiple candidate actions sampled from a policy [4, 5]. The critic can compare the candidate trajectories and select the action that makes the greatest progress towards success, thereby steering away from failures. Such progress/failure-aware critics aid standard behavior cloning policies which cannot reason about failures because they have only been trained on successful demonstrations.

We find that, while off-the-shelf VLMs [6, 7] and progress detection methods [8, 9] can distinguish initial from final task states and identify high-level subtasks, they fail to discriminate between fine-grained progress and failures. A subtle 1mm gripper misalignment is close, but still insufficient, causing the overall policy to fail. On eight robot tasks, Gemini 3 discriminates success versus failure with only 54.7% accuracy, and the progress detection methods ROVER and ProgressLM [8, 10] achieve only 72.1% and 61.0% accuracy respectively – where chance is 50%. ROVER achieves much higher accuracy on coarse progress detection, highlighting that these methods struggle in the fine-grained regime where the time interval gap narrows.

We introduce a framework to fine-tune VLMs for fine-grained progress/failure discrimination, and use them as critics to improve robot policy performance in-the-loop. To fine-tune VLMs, we generate paired success and failure rollouts from matched initial conditions, enabling construction of a fine-grained progress comparison dataset. We emphasize the need to incorporate policy specific failures to train the critic. This procedure only requires binary success/failure labels at the end of the trajectory, and naturally captures the failure modes of the deployed policy. We then integrate the critic into a robot policy for test-time performance improvement via policy steering. We sample

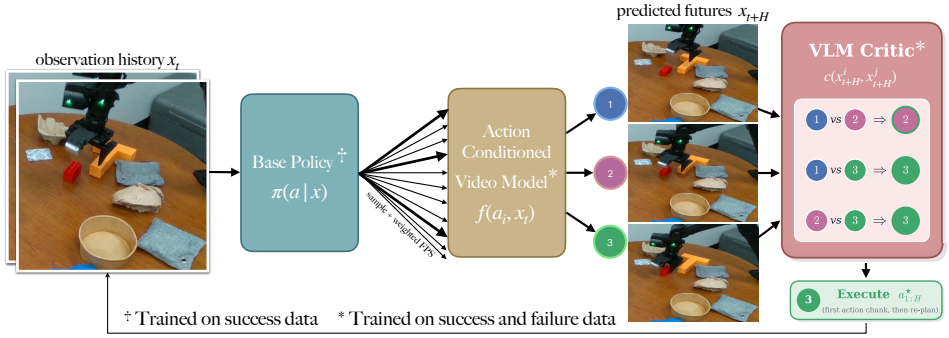


Figure 1: **Critic in-the-loop.** Given an observation, a learned stochastic policy samples K unique candidate action sequences. An action-conditioned generative video model synthesizes visual observations to produce a terminal state per candidate action. The critic performs pairwise progress comparisons to select the best candidate, which is then executed and the re-planning continues.

multiple candidate action sequences from the policy, use a generative video model to synthesize future visual states, and select which one to execute using the critic (Fig. 1).

Critics fine-tuned with this framework achieve over 90% accuracy and substantially outperform baselines in both simulation and real world experiments, and generalize to tasks unseen during training. We systematically evaluate the critic in-the-loop across real world tasks and simulation tasks. We see an average +11% improvement in success rate and 2.3x improvement in mean intersection over union (mIoU) over the base policy on real world tasks, and +5.9% improvement in success rate across all simulation tasks.

2 Related Work

Robot policies and test-time policy steering. Modern robot manipulation policies include both task-specific visuomotor policies and large vision-language-action (VLA) models that can model multimodal action distributions [11, 12, 13, 14, 15]. Recent VLA and generalist robot policies further scale this paradigm with larger backbones and action chunking [16, 17, 18, 19, 20, 21, 22]. Many of these policies share an important test-time property: stochastic policies can produce multiple plausible action candidates from the same observation. This creates an opportunity for policy improvement without retraining. Several methods improve policies at test time by steering [23, 24, 25], reranking or sampling candidate actions [26, 27], or composing policies [28]. Related work also uses predictive world models to strengthen or guide robot policies [29, 30]. These approaches show that predicting future visual states is effective in robot learning.

Vision-language models as critics. Large vision-language models have increasingly been used in robotics as planners, reward functions, value estimators, and critics [3, 31, 32, 33, 34, 35, 36]. For example, SayCan grounds language-model plans in robotic affordances to select feasible skills [37]. VLMs are also used as feedback or scoring functions for either training policies with RL [38] or optimizing actions in a model predictive control loop [39]. In VLMPC, a VLM-derived hierarchical cost function is used within model-predictive control to evaluate candidate action sequences [5]. Recent VLM-in-the-loop steering methods use pretrained VLM representations or latent alignment objectives to guide action selection [4]. These methods show that VLMs can provide useful semantic priors for robot decision-making. While these works rely on relatively coarse judgments, closed-loop policy steering towards success requires differentiating between small changes in gripper pose, object contact, alignment, or early signs of failure. Our work therefore studies VLMs in a fine-grained pairwise comparison between policy-action candidates, rather than as generic semantic goal verifiers/reward functions.

Progress reasoning and failure detection. Several methods aim to estimate task progress from visual observations [40, 41, 42, 43, 44, 45]. Generative Value Learning (GVL) treats VLMs as in-context value learners to obtain the progress value of a new observation [9]. ROVER decomposes

embodied videos into subtasks and uses recursive VLM reasoning to estimate task progress [8]. ProgressLM fine-tunes a VLM to predict absolute progress scores for frames given task demonstrations or key frames [10]. RoboMeter [46] is a reward model to learn general failure cases using failed/suboptimal trajectories, whereas we focus on policy-specific failures and fine-grained differences. Recent work has studied how to classify and use failures at inference time for recovery [47, 48, 49, 50, 51]. These methods are closely related because they also ask whether VLMs can reason about temporal progress in embodied tasks, but they typically estimate more coarse progress on individual frames or videos. Our critic is related to preference-learning and reward-modeling approaches, where a model is trained from comparisons rather than absolute labels [52, 53]. We focus on learning fine-grained pairwise progress critics from successful and failure policy rollouts. This lets the critic recognize both task progress and policy-specific failure modes, which can then be avoided during test-time selection.

3 Method

We use VLMs as critics to guide stochastic robot policies towards success at test-time. Given an image x, the policy predicts absolute joint-angle actions $a^* \in \mathbb{R}^{D \times H}$ to complete the task, where D is the number of degrees of freedom and H is the policy horizon. Our approach will sample multiple action trajectories from a base policy $\pi(x)$ and use a critic to select the final action trajectory:

$$a^* = \arg \min_{a_i \sim \pi(x)} \sum_{j \neq i} c(v_i, v_j) \quad \text{where} \quad v_i = f(a_i, x). \tag{1}$$

f is a generative video model producing the final frame v which acts as a forward dynamics model, and c is a critic that scores the utility of a rollout (Fig 1). We instantiate base policy π as any stochastic robot policy (diffusion policy or GR00T N1.5 in our experiments), f as a video diffusion model, c as a large vision-language model.

We start with our behavior cloned base policy that is already trained on expert demonstrations, and attach a critic to it. We show that we can design this critic to steer the base policy towards successful outcomes, allowing us to integrate the knowledge from VLMs into policies.

3.1 Training VLMs as Critics

We train a VLM to predict which frame reflects greater progress toward the task goal. Note that lack of progress can also mean failure has occurred. Given two frames (v_i, v_j) from a robot manipulation trajectory and a natural language description of the task $\mathcal{T}$, the VLM makes a discrete decision $c(v_i, v_j) \in \{+1, -1\}$ by outputting a text token (preserving the VLM's natural output space). A $+1$ indicates v_j shows more progress than v_i, and -1 indicates the opposite. We provide v_i and v_j to the VLM as two separate images alongside the task description in context. We do supervised finetuning on our dataset D to minimize the cross entropy loss:

$$\mathcal{L}_{\text{critic}}(\theta) = -\frac{1}{|D|} \sum_{(v_i, v_j, y) \in D} \log c_\theta(y \mid v_i, v_j) \quad \text{for} \quad y \in \{+1, -1\}. \tag{2}$$

To construct a supervised fine-tuning dataset D tailored to learning fine-grained visual task progress/failure, we roll out the policy on the same set of initial conditions multiple times to obtain paired success S^s and failure trajectories S^f (top half of Fig 2). In cases where recording paired failures from the policy is infeasible, we note that human demonstrations of failure trajectories can be quickly and easily collected. We construct pairs of frames from S^s and S^f:

Success-Success Progress Pairs. For each successful trajectory S^s of length L^s, and for each temporal interval $l \in \{4, 8, 12, 16\}$, a pair is defined as consecutive observations from an earlier and later timestep in the trajectory, $(S^s_t, S^s_{(t+l)})$ for $t \in \{0, \ldots, L^s - l\}$. Under successful executions, later timesteps typically reflect greater completion toward the goal; therefore, $S^s_{(t+l)}$ reflects greater completion than S^s_t and the pair is assigned label $+1$ (as visualized in Figure 2, bottom left). These success-success pairs teach the model to recognize gradual forward progress.

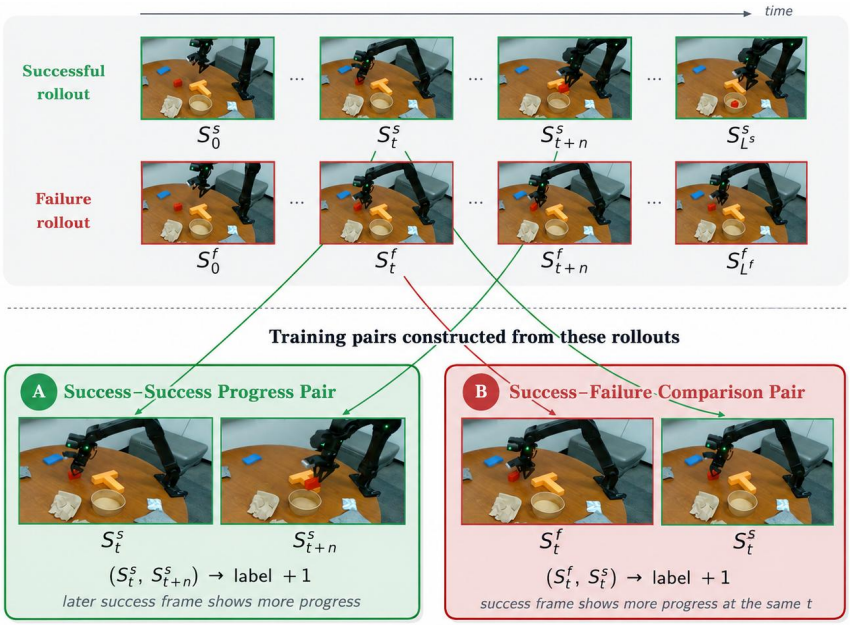


Figure 2: **Training Robot Critics.** VLMs are finetuned with successful and failed rollouts to enable fine-grained task progress/failure detection. To construct the training dataset, consecutive frames are used from successful trajectories to obtain task progress data. Additionally, success and failure frame pairs from the same initial condition are used to obtain policy- and task-specific failure data.

Success–Failure Comparison Pairs. We observe that fine-tuning the VLMs on the successful demonstrations is insufficient as they cannot identify failures without ever having seen failure data. To address this limitation, we include failure trajectory rollouts, S^f, that originate from the same initial conditions as a successful rollout, S^s. We construct pairs (S_t^f, S_t^s) for $t \in \{0, \ldots, L^f\}$, and label them $+1$ as visualized in bottom right box in Fig 2 (i.e., the success frame shows more task progress than the failure frame at the same point in time). This teaches the critic what "bad actions" look like, tailored to this specific policy, task, and environment combination. Creating success-failure pairs with the same initial conditions is important as it helps the VLM learn what kinds of actions cause subtle failures versus what are simply variances in object/robot pose.

3.2 Integrating Critics into the Policy

We use the fine-tuned critic in the loop for test-time policy improvements. At each re-planning step, given the observation x_t, we draw N i.i.d. candidate action chunks from the base policy: $a^{(i)} \sim \pi(x_t)$ for $i \in \{1, \ldots, N\}$. We do greedy farthest-point sampling (FPS) over a weighted joint-space L2 distance to prune to a diverse subset of K actions. Each candidate $a^{(i)}$ specifies a sequence of low-level actions; to derive images that the critic can compare between, we execute all K action chunks with an action-conditioned video model to obtain the terminal states $x_{t+H}^{(i)} = f(a_i, x_t)$.

Action-conditioned video prediction models allow a robot to learn how its actions affect objects in its environment [54, 55]. We use the HunyuanVideo 1.5 pretrained image-to-video Diffusion Transformer [56], a state-of-the-art video generative model. To condition on a sequence of actions, we insert cross-attention blocks that attend to the input action sequence a [57]. We encode the sequence using a transformer encoder to produce a single token per frame, and fine-tune the model using a flow-matching objective to generate videos conditioned on the encoded action tokens. Each action input corresponds to one frame of the predicted video. More details provided in Section C.3.

We then form all $\binom{K}{2}$ pairwise comparisons of these terminal states and query our learned critic to determine which state reflects greater task progress. Each comparison awards a vote to the preferred candidate. We select the candidate whose terminal state accumulates the most votes. The winning

action a^* is then executed in the real environment and this sample-and-select procedure is applied at every re-planning step until task completion or failure.

4 Critic Experiments

We aim to understand the performance of policy critics by answering the following key questions: **Q1.** Does our finetuning approach enable stronger critics in different tasks and environments? **Q2.** Is failure data important? **Q3.** How does our method compare on increasingly fine-grained detection?

4.1 Robot Dataset Setup

RoboCasa. We use RoboCasa [58] to evaluate task-progress assessment. To generate both success and failure data, we train a policy π on each task using the provided 50 human demonstrations. We then roll out the policy π multiple times and collect successful and failure trajectories per initial condition for each task. Across 30 initial conditions, we collect paired success and failure trajectories, and we construct the training dataset as described in Section 3.1 which generates approximately 21k pairs per task. We hold out 7 success and 7 failure trajectories for evaluation (about 5k pairs). We report accuracy on **ID** (in-distribution objects) and **OOD** (out-of-distribution objects).

LBM. We also evaluate our critic finetuning method on the LBM dataset [59]. We use a publicly released portion of the [59] dataset which has 8 tasks including: BimanualBikeRotorInstall, BimanualClearKitchenCounter, BimanualSetUpBreakfastTable, CleanLitterBox, CutAppleIntoSlices, PushCoasterToMug, PutKiwiInCenterOfTable, and TurnMugRightsideUp. For each dataset, there are 50 initial conditions that 2 or 3 policies are rolled out on and videos are generated from: either a Single Task policy, the Finetuned LBM policy, and for some tasks there additionally a rollout from the pre-trained LBM policy (policy trainings are described in [59]). These rollouts lead to either successes or failures which are labeled by human annotators. We create two splits: 40 initial conditions per task for fine-tuning and 10 held-out initial conditions for testing.

4.2 Baselines

We compare against 3 types of baselines: out-of-the-box foundation models, prompted progress detection methods, and fine-tuned progress detection methods. See Section C.4 for prompts.

Out-of-the-box foundation models. We evaluate pretrained foundation models including **Qwen2.5-VL** [6] and **Gemini 3** [7] without finetuning. Similar to our method, we prompt these VLMs with the image pair along with the robot task/environment descriptions and the discrimination task description. These methods output a binary label $y \in \{+1, -1\}$.

Prompted progress detection. We evaluate two prompted progress-reasoning methods. **GVL** [9] predicts per-frame progress given in-context progress annotations along a demonstration trajectory. **ROVER** [8] decomposes videos into subtasks using recursive prompting, predicting subtask id and progress scores. For both methods, we compute the signed difference between each frame's progress scores to obtain binary labels $y \in \{+1, -1\}$.

Fine-tuned progress detection. We also compare to **ProgressLM** [10], which fine-tunes Qwen2.5-VL to predict an absolute progress score from demonstration key frames with known progress percentages. We compute the signed difference between the absolute score of the two images in the pair to obtain binary labels $y \in \{+1, -1\}$. We fine-tune ProgressLM-7B on the same RoboCasa dataset, using 10 demonstration key frames per trajectory.

Ours. We fine-tune Qwen2.5-VL-7B with Eq. (2) on successful and failure rollouts. We also run an ablation where we fine-tune only on successful rollouts.

Method	Countr Cab	Cab Countr	Cab Micro	Micro Cab	Countr Sink	Sink Countr	Countr Stove	Stove Countr	Avg.
Overall Accuracy									
Qwen2.5-32B	0.495	0.508	0.502	0.489	0.502	0.491	0.512	0.490	0.499
Gemini 3	0.706	0.578	0.428	0.498	0.659	0.681	0.398	0.424	0.547
GVL	0.528	0.462	0.584	0.527	0.576	0.543	0.538	0.486	0.531
ROVER	0.707	0.677	0.798	0.758	0.727	0.657	0.710	0.737	0.721
ProgressLM	0.206	0.498	0.709	0.533	0.850	0.597	0.774	0.715	0.610
Ours (Success-Only)	0.927	0.876	0.874	0.892	0.892	0.858	0.886	0.885	0.886
Ours (Full Method)	**0.966**	**0.931**	**0.908**	**0.942**	**0.965**	**0.918**	**0.916**	**0.943**	**0.936**
OOD Only Accuracy									
ProgressLM	0.200	0.495	0.704	0.598	0.843	0.502	0.770	0.680	0.599
Ours (Success-Only)	0.940	0.851	0.829	0.872	0.894	0.792	0.897	0.896	0.871
Ours (Full Method)	**0.970**	**0.909**	**0.863**	**0.916**	**0.958**	**0.870**	**0.903**	**0.924**	**0.914**

Table 1: **Critic accuracy** on various tasks from the RoboCasa dataset [58]. In-distribution (ID) refers to objects the VLM has seen during training. Out-of-distribution (OOD) are unseen objects on the same tasks. We report average in-distribution and out-of-distribution accuracy. For fine-tuned methods, we also report OOD accuracy separately. Furthermore, we show that finetuning with success and failure data (Ours) leads to stronger fine-grained progress/failure understanding than just success (Ours Success-only).

4.3 Results

Q1. Observing subtle failures creates a strong critic. Out-of-the-box models have never seen the nuances of how a specific policy can fail, something that is typically only understood after observing several rollouts of success and failures. Consistent with this observation, Tables 1 and 2 show that prompted, off-the-shelf VLMs are unreliable for fine-grained task progress. Our method shows strong ID and OOD accuracy on real and simulation datasets. Furthermore, our method generalizes to new tasks that it was never fine-tuned on, suggesting a promising pretraining strategy for future task learning (full results in Section A.2). Our fine-tuned critic learns to both de-prioritize actions that move away from the goal (or failures) and prefer states that are closer to task completion. This fine-grained capability is crucial for using critics as practical guidance signals in closed-loop policy selection as we investigate in Section 5.

Q2. Failure data is necessary for useful critics. Our results show that finetuning on only successful rollouts does not teach the critic to recognize failure modes. Critics finetuned only on success leads to a 26% drop in accuracy on success-failure pairs, whereas a critic fine-tuned on both success and failure rollouts yields strong performance on both success-success and success-failure pairs. One explanation for this might be that despite foundation VLMs' internet knowledge, many robot datasets that exist on the internet are often released for fine-tuning purposes. Therefore, these models may have a strong bias towards videos of robots succeeding at a task. This means common failure modes are under-represented in large pretrained foundation models. Furthermore, generic failure modes are hard to quantify, and policy-specific failures can look very different.

Task	Gem.	ROVER	ProgLM	Ours
BimanBike	.473	.717	.443	**.980**
BimanClear	.492	.545	.709	**.988**
BimanSetUp	.545	.758	.613	**.889**
CleanBox	.510	.615	.615	**1.00**
CutApple	.697	.646	.661	**.919**
PushCoaster	.608	.712	.667	**.885**
PutOnTable	.510	.490	.615	**.778**
TurnUpright	.596	.547	.600	**.763**
Avg.	.554	.629	.615	**.900**

Table 2: **Critic accuracy** on the real world LBM dataset.

Q3. Fine-grained Capabilities Comparison. To isolate the effect of visual difference magnitude, we evaluate ROVER, ProgressLM, and our method under two sampling regimes. In the *fine-grained* regime, we sample query frames with small temporal intervals ($l = 16, 32, 48$ steps $\approx 1-5s$ apart), where differences are subtle. In the *coarse* regime, we sample frames at larger intervals at more than seven seconds apart, where visual differences are more pronounced. As the frame interval length decreases, performance for prior methods drops from an average of 73% to 56% across the eight RoboCasa tasks, indicating that these models are more reliable when differences are large (see plot in A.1). Practical policy guidance, however, requires performance in the fine-grained regime as well, a regime where our critic excels with only a 2% drop.

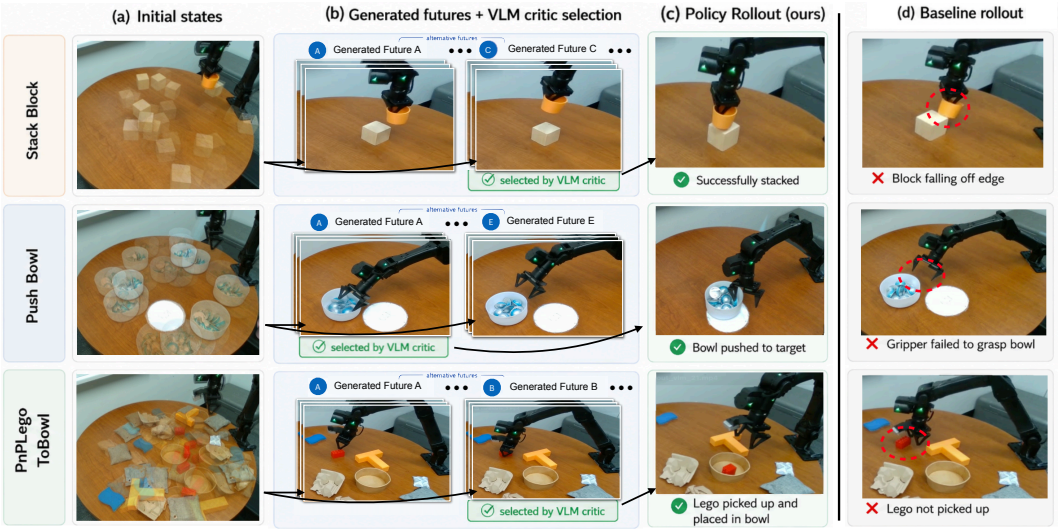


Figure 3: **Real-world Evaluation Results**. (a) Initial states for all eval episodes overlaid - all methods start with the same set of initial states, matched manually with reference images. (b) Critical decision points where the VLM chooses the correct action candidate to lead to success (c) Our method succeeds at the task at the end of the rollout (d) The baseline policy fails at those critical points and therefore fails to complete the task.

5 Critic In-The-Loop Experiments

Here we test whether these critics translate into closed-loop robot policy improvements in both real-world (Sec 5.1) and simulation (Sec 5.2) settings.

To understand the upper bound on our approach, we consider an idealized case, where we had an oracle critic that always picked the best rollout sampled. This improved the policy performance from 31% to 48% (see Section B for acc-vs-K plot), showing that the base policy does assign significant probability density to successful rollouts, but does not always sample them. This result suggests that, if we can learn a strong critic, we can potentially improve the policy performance substantially.

5.1 Real World Experiments

Setup. We experiment on tabletop manipulation tasks using 7-DoF YAM Pro Arms from I2RT. The workspace is observed by two Intel RealSense D435i cameras placed approximately 660 mm apart and angled toward the table at roughly 45°. We use the Yam Pro Leader arm to teleoperate the robot to collect data for all real world tasks (Appendix Table 5 details the number of multi modal expert demonstrations we collect per task). Random demonstrations show the expert randomly moving in the environment and interacting with objects. Only successful demos are used for training the policy, successful and failure demos are used for finetuning the VLM critic, and finally successful, failure, and random demos are used for finetuning the world model. The breakdown of the finetuning dataset for the VLM critic is shown in Appendix Table 6.

We evaluate on multiple tasks: (i) Stacking a cup onto a block, (ii) Pushing a bowl onto a white place mat, (iii) Pick and Place a Lego block into a brown bowl in the presence of eight distractor objects, and (iv) Pickup Lego. For each task, we collect expert teleoperated demonstrations at 30 Hz. Details on number of demonstrations per task and model included in Table 5. All real-world results are evaluated over 50 trials per task with varying unseen initial conditions as seen in Fig 3a.

Our base policy π is a CNN-based Diffusion Policy [60]. We subsample expert trajectories at 10 Hz for training. The policy receives the current robot pose and two RealSense camera images at 640×360, and encodes them using two pretrained ResNet-18 image encoders. It predicts a horizon of 32 absolute joint-angle actions into the future. During execution, we only execute the first 16 actions

before re-planning. This follows the standard closed-loop policy execution setup, where executing a shorter prefix helps reduce compounding errors from stale predictions. The video generation model (Hunyuan1.5) [56] is finetuned on the successful and failure demonstrations used to train the critic/policy, along with random trajectories. We finetune the model per-task across 8 A100 GPUs.

At each re-planning step, we sample 100 actions from the base policy. We perform greedy farthest point sampling in weighted joint space where we down weight the gripper contribution by 0.5 (as this often dominates the variance). Each of the $K = 5$ remaining candidates is rolled forward with the action-conditioned video model, and we use the final frames of the generated videos as the input to the critic. The robot observes the new state and repeats the sample, generate, compare, and execute loop until task completion or a fixed max number of steps.

Results. Table 3 reports real-world policy performance. On all evaluated tasks, our critic improves success rate over the base diffusion policy. We improve Stacking by +18%, and Push Bowl experiments show a 2.3x improvement in mIoU. In the cluttered and difficult settings of Pickup Lego and Pick and Place Lego to bowl, our method improves success rate by +10%/+4%. These results suggest that strong critics can find the needle in the action haystack to improve performance of stochastic policies. For example, in the Lego task, different diffusion samples can lead to visibly different gripper-object alignments as seen in Figure 3 row 3, column b. The critic is able to select the image that shows a stable grasp between the robot gripper and the Lego.

We also compare our critic against a critic trained without failure rollouts (Success-only), and find that similar to the baseline, the success-only critic achieves 12/50 on the Pickup Lego task and 6/50 on PickPlace Lego to Bowl. This supports our central hypothesis: for critic-in-the-loop policy improvement, it is not sufficient to recognize progress along successful demonstrations. The critic must also learn what failures look like to allow the policy to improve upon its mistakes.

Finally, we compare the average mean squared error (MSE) between the video generated by the candidate action that the critic chooses to execute and the ground truth observation video that actually rolls out in the real world upon executing $a*$. On cases where the policy successfully completes the task, the video model scores an average MSE of 0.00467 across all generated videos along the execution. When the policy fails to complete the task, the video model has a higher MSE of 0.00523.

5.2 Simulation Experimental Setup and Results

Setup. Alongside real world experiments, we conduct experiments on the 18 atomic tasks from the official RoboCasa365 simulation framework [61]. Our base policy π is a multi-task language-conditioned policy. We use the publicly released Robocasa365 checkpoint which starts from pre-trained checkpoints released by NVIDIA Isaac GR00T N1.5 [62]. This multi-task policy is trained on data across 300 tasks, comprising 65 atomic tasks and 235 composite tasks. For each task, 100 task demonstrations per task are used, resulting in 482 hours of total data.

We follow the Robocasa365 setup by evaluating on the pretrain scenes. To gather failure demonstrations, we rollout GR00T N1.5 on 50 different initial conditions across 5 different seeds from the pretraining set, following the protocol in Robocasa365 benchmarking. Of these 250 rollouts, the policy exhibits variance, creating a dataset of failures to train the VLM critic. The breakdown of total (train and validation) number of demonstrations and number of success/failure pairs before balancing are provided in Appendix Table 6. We follow the same evaluation parameters with an action horizon of 16 and standard closed-loop policy execution setup. We train a multi-task critic with the same procedure as in real. Candidate videos are rendered by the simulator for these tasks. We use the same hyperparameters as real for critic-in-the-loop with N=100, K=5, and greedy farthest point sampling.

Results. As seen in Table 3, across all tasks, our critic in-the-loop achieves 49.5% success rate compared to baseline GR00T N1.5 of 43.6%, an average gain of +5.9%. Our method shows significant improvements in tasks that require pushing/pulling buttons, doors, and levers.

Task	Diffusion Policy	
	Base	**Ours**
Stacking	23 / 50	**32 / 50**
Push-Bowl	0.140 mIoU	**0.321 mIoU**
Pickup Lego	11 / 50	**16 / 50**
PickPlace Lego To Bowl	6 / 50	**8 / 50**

(a) Real World

Task	GR00T N1.5		Task	GR00T N1.5	
	Base	**Ours**		**Base**	**Ours**
NavigateKitchen	0.040 ± 0.009	0.040 ± 0.009	PickPlaceCounterToStove	$\mathbf{0.696 \pm 0.015}$	0.568 ± 0.020
CloseBlenderLid	$\mathbf{0.072 \pm 0.016}$	0.013 ± 0.007	PickPlaceDrawerToCounter	0.116 ± 0.021	$\mathbf{0.180 \pm 0.021}$
CloseFridge	0.772 ± 0.031	$\mathbf{0.863 \pm 0.014}$	PickPlaceSinkToCounter	$\mathbf{0.888 \pm 0.022}$	0.757 ± 0.022
CloseToasterOvenDoor	0.356 ± 0.017	$\mathbf{0.623 \pm 0.032}$	PickPlaceToasterToCounter	$\mathbf{0.780 \pm 0.019}$	0.729 ± 0.022
CoffeeSetupMug	0.160 ± 0.028	0.184 ± 0.031	SlideDishwasherRack	0.300 ± 0.021	$\mathbf{0.530 \pm 0.018}$
OpenCabinet	0.500 ± 0.025	$\mathbf{0.552 \pm 0.019}$	TurnOffStove	0.104 ± 0.021	$\mathbf{0.203 \pm 0.017}$
OpenDrawer	0.544 ± 0.010	$\mathbf{0.648 \pm 0.019}$	TurnOnElectricKettle	0.348 ± 0.016	$\mathbf{0.620 \pm 0.032}$
OpenStandMixerHead	0.588 ± 0.023	$\mathbf{0.713 \pm 0.015}$	TurnOnMicrowave	0.384 ± 0.025	$\mathbf{0.427 \pm 0.029}$
PickPlaceCounterToCabinet	0.760 ± 0.017	0.752 ± 0.030	TurnOnSinkFaucet	0.440 ± 0.017	$\mathbf{0.500 \pm 0.013}$
			Average	Base: 0.436 ± 0.020	Ours: $\mathbf{0.495 \pm 0.021}$

(b) RoboCasa365

Table 3: **Policy results.** We compare baseline Diffusion Policy and GR00T N1.5 against our critic-guided policies on four real-world and eighteen RoboCasa365 simulation atomic tasks. For RoboCasa365, we report mean success rate $\pm$ standard error across 5 runs. The critic selects candidate action proposals from the base policy that help the policy avoid failures, leading to overall improvement in task success rate.

5.3 Limitations

Weak action-conditioned video model predictions can cause failures. For example, we sometimes see the model hallucinate grasping the Lego in the predicted video when the real-world rollout fails to grasp, or the bowl pushed to a slightly different end location than the ground truth. We expect these errors to reduce as video models improve. The video model takes $\sim$25 seconds to synthesize five videos in parallel (across 8 A100 GPUs and 10 inference steps), and the critic takes $\sim$1 second to generate all pair-wise rankings in parallel (across 8 A100 GPUs). We expect the video model's latency to reduce as faster implementations are developed.

6 Conclusion

We studied a method to finetune VLM critics for pairwise progress/failure detection by using success and failure rollouts. The resulting critic substantially improves offline accuracy in simulation and on a real-world rollout dataset, particularly on success–failure discrimination. We put these critics into closed-loop policies to select between several candidate future states generated from an action-conditioned video model, improving average policy success rate by 11% across real-world tasks and 5.9% across simulation tasks.

Acknowledgments

We thank Matei Ciocarlie and Huy Ha for key discussions. This research is based on work partially supported by the Toyota Research Institute and NSF awards #2046910 and #2132519. This work is also partially supported by the funds provided by the National Science Foundation and by DoD OUSD (R&E) under Cooperative Agreement PHY-2229929 (The NSF AI Institute for Artificial and Natural Intelligence).

References

[1] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

[2] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[3] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[4] Y. Wu, R. Tian, G. Swamy, and A. Bajcsy. From foresight to forethought: Vlm-in-the-loop policy steering via latent alignment. *ArXiv*, abs/2502.01828, 2025. URL https://api.semanticscholar.org/CorpusID:276107486.

[5] W. Zhao, J. Chen, Z. Meng, D. Mao, R. Song, and W. Zhang. Vlmpc: Vision-language model predictive control for robotic manipulation. *ArXiv*, abs/2407.09829, 2024. URL https://api.semanticscholar.org/CorpusID:271212525.

[6] S. Bai, K. qin Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang, H. Zhong, Y. Zhu, M. Yang, Z. Li, J. Wan, P. Wang, W. Ding, Z. Fu, Y. Xu, J. Ye, X. Zhang, T. Xie, Z. Cheng, H. Zhang, Z. Yang, H. Xu, and J. Lin. Qwen2.5-vl technical report. *ArXiv*, abs/2502.13923, 2025. URL https://api.semanticscholar.org/CorpusID:276449796.

[7] G. C. et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *ArXiv*, abs/2507.06261, 2025. URL https://api.semanticscholar.org/CorpusID:280151524.

[8] P. Schroeder, O. Biza, T. Weng, H. Luo, and J. R. Glass. Rover: Recursive reasoning over videos with vision-language models for embodied tasks. *ArXiv*, abs/2508.01943, 2025. URL https://api.semanticscholar.org/CorpusID:280422750.

[9] Y. J. Ma, J. Hejna, A. Wahid, C. Fu, D. Shah, J. Liang, Z. Xu, S. Kirmani, P. Xu, D. Driess, T. Xiao, J. Tompson, O. Bastani, D. Jayaraman, W. Yu, T. Zhang, D. Sadigh, and F. Xia. Vision language models are in-context value learners. *ArXiv*, abs/2411.04549, 2024. URL https://api.semanticscholar.org/CorpusID:273877849.

[10] J. Zhang, C. Qian, H. Sun, H. Lu, D. Wang, L. Xue, and H. Liu. Progresslm: Towards progress reasoning in vision-language models. *ArXiv*, abs/2601.15224, 2026. URL https://api.semanticscholar.org/CorpusID:284917784.

[11] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *ArXiv*, abs/2402.10329, 2024. URL https://api.semanticscholar.org/CorpusID:267740127.

[12] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Y. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J.-P. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. B. Simpson, Q. U. Vương, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Zhao, C. Agia, R. Baijal, M. G. Castro, D. L. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, M. Z. Irshad, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. M. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Mart'in-Mart'in, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset. *ArXiv*, abs/2403.12945, 2024. URL https://api.semanticscholar.org/CorpusID:268531351.

[13] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, A. Raffin, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Ichter, C. Lu, C. Xu, C. Finn, C. Xu, C. Chi, C. Huang, C. Chan, C. Pan, C. Fu, C. Devin, D. Driess, D. Pathak, D. Shah, D. Büchler, D. Kalashnikov, D. Sadigh, E. Johns, F. Ceola, F. Xia, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Schiavi, H. Su, H. Fang, H. Shi, H. B. Amor, H. I. Christensen, H. Furuta, H. R. Walke, H. Fang, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Kim, J. Schneider, J. Hsu, J. Bohg, J. Bingham, J. Wu, J. Wu, J. Luo, J. Gu, J. Tan, J. Oh, J. Malik, J. Tompson, J. Yang, J. J. Lim, J. Silvério, J. Han, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Zhang, K. Majd, K. Rana, K. P. Srinivasan, L. Y. Chen, L. Pinto, L. Tan, L. Ott, L. Lee, M. Tomizuka, M. Du, M. Ahn, M. Zhang, M. Ding, M. K. Srirama, M. Sharma, M. J. Kim, M. Z. Irshad, N. Kanazawa, N. Hansen, N. M. O. Heess, N. J. Joshi, N. Suenderhauf, N. D. Palo, N. M. Shafiullah, O. Mees, O. Kroemer, P. R. Sanketi, P. Wohlhart, P. Xu, P. Sermanet, P. Sundaresan, Q. H. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Mendonca, R. Shah, R. Hoque, R. C. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Moore, S. Bahl, S. Dass, S. Song, S. Xu, S. Haldar, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Dasari, S. Belkhale, T. Osa, T. Harada, T. Matsushima, T. Xiao, T. Yu, T. Ding, T. Davchev, T. Zhao, T. Armstrong, T. Darrell, V. Jain, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Wang, X. Zhu, X. Li, Y. Lu, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Xu, Y. Wang, Y. Bisk, Y. Cho, Y. Lee, Y. Cui, Y. hua Wu, Y. Tang, Y. Zhu, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Xu, and Z. J. Cui. Open x-embodiment: Robotic learning datasets and rt-x models : Open x-embodiment collaboration0. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903, 2023. URL https://api.semanticscholar.org/CorpusID:263626099.

[14] W. Pumacay, I. Singh, J. Duan, R. Krishna, J. Thomason, and D. Fox. The colosseum: A benchmark for evaluating generalization for robotic manipulation. *ArXiv*, abs/2402.08191, 2024. URL https://api.semanticscholar.org/CorpusID:267636930.

[15] H. Kress-Gazit, K. Hashimoto, N. Kuppuswamy, P. Shah, P. Horgan, G. Richardson, S. Feng, and B. Burchfiel. Robot learning as an empirical science: Best practices for policy evaluation. *ArXiv*, abs/2409.09491, 2024. URL https://api.semanticscholar.org/CorpusID:272689744.

[16] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. R. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. π0.5: a vision-language-action model with open-world generalization. *ArXiv*, abs/2504.16054, 2025. URL https://api.semanticscholar.org/CorpusID:277993634.

[17] K. Black, M. Y. Galliker, and S. Levine. Real-time execution of action chunking flow policies. *arXiv preprint arXiv:2506.07339*, 2025.

[18] W. Wu, F. Lu, Y. Wang, S. Yang, S. Liu, F. Wang, Q. Zhu, H. Sun, Y. Wang, S. Ma, et al. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*, 2026.

[19] H. Xie, B. Wen, J. Zheng, Z. Chen, F. Hong, H. Diao, and Z. Liu. Dynamicvla: A vision-language-action model for dynamic object manipulation. *arXiv preprint arXiv:2601.22153*, 2026.

[20] L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdulmohsin, M. Tschannen, E. Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[21] J. Liang, R. Liu, E. Ozguroglu, S. Sudhakar, A. Dave, P. Tokmakov, S. Song, and C. Vondrick. Dreamitate: Real-world visuomotor policy learning via video generation, 2024.

[22] J. Liang, P. Tokmakov, R. Liu, S. Sudhakar, P. Shah, R. Ambrus, and C. Vondrick. Video generators are robot policies, 2025. URL https://arxiv.org/abs/2508.00795.

[23] Y. Wang, L. Wang, Y. Du, B. Sundaralingam, X. Yang, Y.-W. Chao, C. Pérez-D'Arpino, D. Fox, and J. A. Shah. Inference-time policy steering through human interactions. *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15626–15633, 2024. URL https://api.semanticscholar.org/CorpusID:274280942.

[24] M. Du and S. Song. Dynaguide: Steering diffusion polices with active dynamic guidance. *ArXiv*, abs/2506.13922, 2025. URL https://api.semanticscholar.org/CorpusID:279410894.

[25] M. Nakamoto, O. Mees, A. Kumar, and S. Levine. Steering your generalists: Improving robotic foundation models via value guidance, 2025. URL https://arxiv.org/abs/2410.13816.

[26] S. Jang, D. Kim, C. Kim, Y. Kim, and J. Shin. Verifier-free test-time sampling for vision language action models. *arXiv preprint arXiv:2510.05681*, 2025.

[27] W. Guo, G. Lu, H. Deng, Z. Wu, Y. Tang, and Z. Wang. Vla-reasoner: Empowering vision-language-action models with reasoning via online monte carlo tree search. *arXiv preprint arXiv:2509.22643*, 2025.

[28] J. Cao, Y. Huang, H. Guo, R. Zhang, M. Nan, W. Mai, J. Wang, H. Cheng, J. Sun, G. Han, et al. Compose your policies! improving diffusion-based or flow-based robot policies via test-time distribution-level composition. *arXiv preprint arXiv:2510.01068*, 2025.

[29] H. Qi, H. Yin, Y. Du, and H. Yang. Strengthening generative robot policies through predictive world modeling. *ArXiv*, abs/2502.00622, 2025. URL https://api.semanticscholar.org/CorpusID:276095203.

[30] S. Gao, W. Liang, K. Zheng, A. Malik, S. Ye, S. Yu, W.-C. Tseng, Y. Dong, K. Mo, C.-H. Lin, et al. Dreamdojo: A generalist robot world model from large-scale human videos. *arXiv preprint arXiv:2602.06949*, 2026.

[31] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. R. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500, 2022. URL https://api.semanticscholar.org/CorpusID:252355542.

[32] J. Gao, B. Sarkar, F. Xia, T. Xiao, J. Wu, B. Ichter, A. Majumdar, and D. Sadigh. Physically grounded vision-language models for robotic manipulation. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12462–12469, 2023. URL https://api.semanticscholar.org/CorpusID:261556939.

[33] Y. J. Ma, W. Liang, V. Som, V. Kumar, A. Zhang, O. Bastani, and D. Jayaraman. Liv: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning*, 2023. URL https://api.semanticscholar.org/CorpusID:258999195.

[34] Y. Du, K. Konyushkova, M. Denil, A. Raju, J. Landon, F. Hill, N. de Freitas, and S. Cabi. Vision-language models as success detectors. *ArXiv*, abs/2303.07280, 2023. URL https://api.semanticscholar.org/CorpusID:257496810.

[35] W. Zhou, M. Tao, C. Zhao, H. Guo, H. Dong, M. Tang, and J. Wang. Physvlm: Enabling visual language models to understand robotic physical reachability. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6940–6949, 2025. URL https://api.semanticscholar.org/CorpusID:276929115.

[36] S. Zhai, Q. Zhang, T. Zhang, F. Huang, H. Zhang, M. Zhou, S. Zhang, L. Liu, S. Lin, and J. Pang. A vision-language-action-critic model for robotic real-world reinforcement learning. *ArXiv*, abs/2509.15937, 2025. URL https://api.semanticscholar.org/CorpusID:281411120.

[37] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. M. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. C. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. M. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, and M. Yan. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, 2022. URL https://api.semanticscholar.org/CorpusID:247939706.

[38] Y. Wang, Z. Sun, J. Zhang, Z. Xian, E. Biyik, D. Held, and Z. Erickson. Rl-vlm-f: Reinforcement learning from vision language foundation model feedback. In *International Conference on Machine Learning*, 2024. URL https://api.semanticscholar.org/CorpusID:267499679.

[39] C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[40] Z. Xue, J. An, X. Yang, and K. Grauman. Progress-aware video frame captioning. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13639–13650, 2024. URL https://api.semanticscholar.org/CorpusID:274446032.

[41] K.-H. Hung, P.-C. Lo, J.-F. Yeh, H.-Y. Hsu, Y.-T. Chen, and W. H. Hsu. Victor: Learning hierarchical vision-instruction correlation rewards for long-horizon manipulation. *ArXiv*, abs/2405.16545, 2024. URL https://api.semanticscholar.org/CorpusID:270064037.

[42] P. Pacaud, R. Garcia, S. Chen, and C. Schmid. Guardian: Detecting robotic planning and execution errors with vision-language models. *ArXiv*, abs/2512.01946, 2025. URL https://api.semanticscholar.org/CorpusID:283448847.

[43] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Parikh, and D. Batra. Vqa: Visual question answering. *International Journal of Computer Vision*, 123:4 – 31, 2015. URL https://api.semanticscholar.org/CorpusID:3180429.

[44] D. A. Hudson and C. D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6693–6702, 2019. URL https://api.semanticscholar.org/CorpusID:152282269.

[45] X. Yue, Y. Ni, K. Zhang, T. Zheng, R. Liu, G. Zhang, S. Stevens, D. Jiang, W. Ren, Y. Sun, C. Wei, B. Yu, R. Yuan, R. Sun, M. Yin, B. Zheng, Z. Yang, Y. Liu, W. Huang, H. Sun, Y. Su, and W. Chen. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9556–9567, 2023. URL https://api.semanticscholar.org/CorpusID:265466525.

[46] A. Liang, Y. Korkmaz, J. Zhang, M. Hwang, A. Anwar, S. Kaushik, A. Shah, A. Huang, L. S. Zettlemoyer, D. Fox, Y. Xiang, A. Li, A. Bobu, A. Gupta, S. Tu, E. Biyik, and J. Zhang. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. 2026. URL https://api.semanticscholar.org/CorpusID:286223261.

[47] C. Xu, T. K. Nguyen, E. Dixon, C. Rodriguez, P. Miller, R. Lee, P. Shah, R. Ambrus, H. Nishimura, and M. Itkina. Can we detect failures without failure data? uncertainty-aware runtime failure detection for imitation learning policies, 2025. URL https://arxiv.org/abs/2503.08558.

[48] J. Duan, W. Pumacay, N. Kumar, Y. R. Wang, S. Tian, W. Yuan, R. Krishna, D. Fox, A. Mandlekar, and Y. Guo. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. *ArXiv*, abs/2410.00371, 2024. URL https://api.semanticscholar.org/CorpusID:273022765.

[49] Z. Lin, J. Duan, H. Fang, D. Fox, R. Krishna, C. Tan, and B. Wen. Failsafe: Reasoning and recovery from failures in vision-language-action models, 2025. URL https://arxiv.org/abs/2510.01642.

[50] J. Park, J. Yoon, B. Jeon, J. Park, J. Shin, N. Cho, K. Lee, S. Yun, and S. Choi. Hierarchical vision language action model using success and failure demonstrations, 2025. URL https://arxiv.org/abs/2512.03913.

[51] H. Li, K. Lei, S. Zang, K. Hu, Y. Liang, B. An, X. Li, and H. Xu. Failure-aware rl: Reliable offline-to-online reinforcement learning with self-recovery for real-world manipulation, 2026. URL https://arxiv.org/abs/2601.07821.

[52] P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences, 2023. URL https://arxiv.org/abs/1706.03741.

[53] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[54] C. Finn, I. J. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. *ArXiv*, abs/1605.07157, 2016. URL https://api.semanticscholar.org/CorpusID:2659157.

[55] S. Yang, Y. Du, K. Ghasemipour, J. Tompson, L. Kaelbling, D. Schuurmans, and P. Abbeel. Learning interactive real-world simulators, 2024. URL https://arxiv.org/abs/2310.06114.

[56] B. Wu, C. Zou, C. Li, D. Huang, F. Yang, H. Tan, J. Peng, J. Wu, J. Xiong, J. Jiang, Linus, Patrol, P. Zhang, P. Chen, P. Zhao, Q. Tian, S. Liu, W. Kong, W. Wang, X. He, X. Li, X. Deng, X. Zhe, Y. Li, Y. Long, Y. Peng, Y. Wu, Y. Liu, Z. Wang, Z. Dai, B. Peng, C. Li, G. Gong, G. Xiao, J. Tian, J. Lin, J. Liu, J. Zhang, J. Lian, K. Pan, L. Wang, L. Niu, M. Chen, M. Chen, M. Zheng, M. Yang, Q. Hu, Q. Yang, Q. Xiao, R. Wu, R. Xu, R. Yuan, S. Sang, S. Huang, S. Gong, S. Huang, W. Guo, X. Yuan, X. Chen, X. Hu, W. Sun, X. Wu, X. Ren, X. Yuan, X. Mi, Y. Zhang, Y. Sun, Y. Lu, Y. Li, Y. Huang, Y. Tang, Y. Li, Y. Deng, Y. Zhou, Z. Hu, Z. Liu, Z. Yang, Z. Yang, Z. Lu, Z. Zhou, and Z. Zhong. Hunyuanvideo 1.5 technical report, 2025. URL https://arxiv.org/abs/2511.18870.

[57] Y. Li and A. Torralba. Multimodal action conditioned video generation, 2025. URL https://arxiv.org/abs/2510.02287.

[58] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *ArXiv*, abs/2406.02523, 2024. URL https://api.semanticscholar.org/CorpusID:270226600.

[59] T. L. Team, J. Barreiros, A. Beaulieu, A. Bhat, R. Cory, E. Cousineau, H. Dai, C. Fang, K. Hashimoto, M. Z. Irshad, M. Itkina, N. Kuppuswamy, K.-H. Lee, K. Liu, D. Mcconachie, I. McMahon, H. Nishimura, C. Phillips-Grafflin, C. Richter, P. Shah, K. P. Srinivasan, B. Wulfe, C. Xu, M. Zhang, A. Alspach, M. Angeles, K. Arora, V. Guizilini, A. M. Castro, D. Chen, T.-S. Chu, S. Creasey, S. Curtis, R. Denitto, E. Dixon, E. Dusel, M. B. R. Ferreira, A. Goncalves, G. E. Gould, D. Guoy, S. Gupta, X. Han, K. Hatch, B. Hathaway, A. Henry, H. Hochsztein, P. Horgan, S. Iwase, D. Jackson, S. Karamcheti, S. S. Keh, J. Masterjohn, J.-P. Mercat, P. T. Miller, P. C. Mitiguy, T. K. Nguyen, J. W. Nimmer, Y. Noguchi, R. Ong, A. O. Onol, O. Pfannenstiehl, R. Poyner, L. P. M. Rocha, G. Richardson, C. Rodriguez, D. Seale, M. Sherman, M. Smith-Jones, D. Tago, P. Tokmakov, M. T. Tran, B. V. Hoorick, I. Vasiljevic, S. Zakharov, M. Zolotas, R. Ambrus, K. Fetzer-Borelli, B. Burchfiel, H. Kress-Gazit, S. Feng, S. Ford, and R. Tedrake. A careful examination of large behavior models for multitask dexterous manipulation. *ArXiv*, abs/2507.05331, 2025. URL https://api.semanticscholar.org/CorpusID:280114687.

[60] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44:1684 – 1704, 2023. URL https://api.semanticscholar.org/CorpusID:257378658.

[61] S. Nasiriany, S. Nasiriany, A. Maddukuri, and Y. Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots, 2026. URL https://arxiv.org/abs/2603.04356.

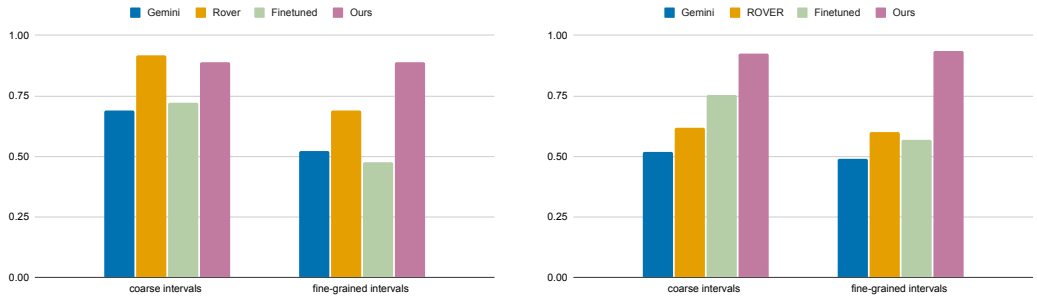
[62] NVIDIA, J. Bjorck, N. C. Fernando Castañeda, X. Da, R. Ding, L. J. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, J. Jang, Z. Jiang, J. Kautz, K. Kundalia, L. Lao, Z. Li, Z. Lin, K. Lin, G. Liu, E. Llontop, L. Magne, A. Mandlekar, A. Narayan, S. Nasiriany, S. Reed, Y. L. Tan, G. Wang, Z. Wang, J. Wang, Q. Wang, J. Xiang, Y. Xie, Y. Xu, Z. Xu, S. Ye, Z. Yu, A. Zhang, H. Zhang, Y. Zhao, R. Zheng, and Y. Zhu. GR00T N1: An open foundation model for generalist humanoid robots. In *ArXiv Preprint*, March 2025.

Appendix

A Additional Critic results

A.1 Fine vs coarse performance

To isolate the effect of visual difference magnitude, we evaluate ROVER, ProgressLM, and our method under two sampling regimes. In the **fine-grained** regime, we sample query frames with small temporal intervals ($l = 16, 32, 48$ steps apart), where differences are subtle. In the **coarse** regime, we sample frames at larger intervals ($l = 74, 100, 150$), where more pronounced changes occur. Figure 4a and Figure 4b show that increasing the frame interval substantially boosts performance for prompted models, indicating that these models are more reliable when differences are large. However, practical policy guidance requires performance in the fine-grained regime as well.



(a) RoboCasa (b) Real World

Figure 4: **Coarse vs fine-grained critic performance**. Current VLM performance degrades when judging fine grained visual differences in task performance. Fine-tuning for fine-grained task progress recognition can boost these scores.

A.2 Generalization to unseen tasks

We evaluate the three strongest methods (ROVER, ProgressLM, and ours) on held-out tasks that are unseen for ProgressLM and our method. ROVER remains prompted and continues to receive one-shot examples, as in other splits. Our method generalizes to new tasks that it was never fine-tuned on (Table 4), suggesting a promising path toward broader generalization as training data and task diversity scale. We believe that this VLM critic finetuning recipe can build strong critic backbones with task finetuning for more learning about policy specific success/failures.

Task	ROVER	ProgressLM	Ours (Success+Failure)
TurnSinkSpout	0.372	0.601	0.700
OpenDrawer	0.559	0.565	0.825
CloseSingleDoor	0.490	0.365	0.750
CloseDoubleDoor	0.571	0.583	0.695
OpenDoubleDoor	0.740	0.783	0.655
OpenSingleDoor	0.473	0.652	0.815
Avg. Multi-task Acc.	**0.533**	**0.591**	**0.740**

Table 4: **Generalization to tasks unseen** during finetuning for ProgressLM and our method (ROVER remains prompted with one-shot examples). Our method generalizes to new tasks never seen during finetuning.

B Additional Policy results

Strong robot policies are typically represented as stochastic distributions, and most behavior cloning evaluations report the mean success rate over a small number of independent runs (different random seeds). Figure 5 shows that rather than averaging, best-of-K sampling yields large gains even for $K \leq 5$, indicating that successful action sequences frequently exist in the policy's support but are not sampled reliably under a single rollout. Furthermore, Figure 6 shows the variance in joint values output by the policy when taking 5 samples at test time. This motivates using a learned critic to select among candidate rollouts at test time.

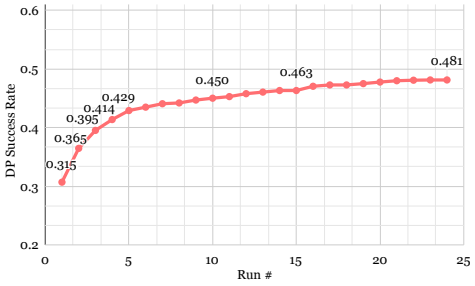


Figure 5: **Best-of-K sampling** reveals substantial headroom for Diffusion Policy, increasing success rate from 31% to 48%.

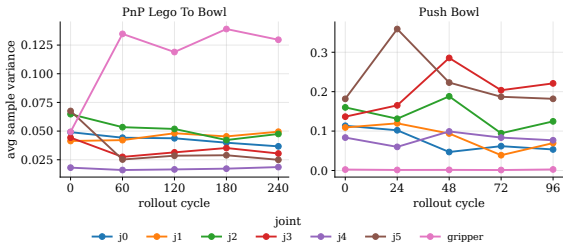


Figure 6: **Per-joint sample variance** across K=5 samples. Variance indicates the multimodality of the policy.

C Experiment Details

C.1 Experimental Setup

We set up two stereo cameras (Intel RealSense D435i cameras) spaced approximately 660 mm apart at a 45 degree angle. The distance between the cameras and the table is about 500 mm. The realworld demonstrations are recorded at a resolution of 1280×720 (each model crops and resizes it to fit the specific model's input requirements). The robot used is I2RT's Yam Pro Arm with 7dof, and we record/store the absolute joint angles of the follower. We use the Yam Pro Leader Arm for teleoperation during data collection. For all tasks, we allow the policy to rollout for a maximum number of steps which is set to 1.5x the average number of steps used during data collection. Task success for the Pick and Place Lego to Bowl and Stacking tasks are binary and defined by a human. For calculating the mIoU in the Push Bowl experiment, the view from one of the stereo cameras is used.

Task	Type	# Demos	Avg. Length
Push-Bowl	Success	30	227.7
	Failure	11	367.0
	Random	10	4,359.9
Stacking	Success	50	184.5
	Failure	39	187.3
	Random	20	825.5
PickPlace Lego to Bowl	Success	200	265.6
	Failure	49	760.7
	Random	10	2,675.4

Table 5: **Real World dataset** statistics by task and trajectory type. Average length is measured in timesteps per demonstration.

C.2 Base Policy

Task	# Succ.	# Fail	# S–S	# S–F
Lego (PnP)	69	39	10,829	1,438
Push Bowl	30	11	14,376	1,446
Stacking	48	38	18,890	3,134

(a) Real-world tasks

Task	# Succ.	# Fail	# S–S	# S–F
CloseBlenderLid	18	232	836	474
CloseFridge	193	57	40,934	3,887
CloseToasterOvenDoor	89	161	7,895	3,616
CoffeeSetupMug	40	210	4,271	2,754
OpenCabinet	125	125	20,235	5,595
OpenDrawer	136	114	13,175	3,712
OpenStandMixerHead	147	103	8,206	1,274
PickPlaceCounterToCabinet	190	60	19,809	1,071
PickPlaceCounterToStove	174	76	19,894	2,277
PickPlaceDrawerToCounter	29	221	2,896	1,832
PickPlaceSinkToCounter	222	28	26,865	880
PickPlaceToasterToCounter	195	55	28,107	3,143
SlideDishwasherRack	75	175	5,088	1,339
TurnOffStove	26	224	2,729	1,938
TurnOnElectricKettle	87	163	6,077	2,231
TurnOnMicrowave	96	154	10,505	5,370
TurnOnSinkFaucet	110	140	12,251	6,889

(b) RoboCasa-365 tasks

Table 6: **VLM critic finetuning datasets**. For real-world tasks, we use human collected data demonstrating success and failures. For Robocasa365, we rollout on 50 initial conditions per task 5 times, which generates multiple success/failure demos (counts shown in columns). From these gathered demos, we generate success–success (S–S) and success–failure (S–F) pairs using our finetuning method. Raw counts are shown before balancing.

We use Diffusion Policy [60] as our robot policy for real world experiments which has the ability to model multimodal distributions and complex robot behaviors. We train a per-task policy on all successful demos for real-world. In simulation, we use the finetune checkpoint GR00T N1.5 [62] which is a single language-conditioned multi-task policy. Training hyperparameter details are listed in Table 7.

Hyperparam.	Value	Hyperparam.	Value
State Norm.	Yes	Action Norm.	Yes
Img. Chunk	2	Img. Size	640×360
State Dim.	7	Action Dim.	7
Exec. Hor.	16	Pred. Hor.	32
Batch Size	64	Epochs	250
LR	10^{-4}	Workers	4
Train Steps	100	Infer. Steps	16

Table 7: **Diffusion Policy training hyperparameters** for real-world experiments.

C.3 Action-Conditioned Video Generation Model

We finetune the pretrained HunyuanVideo-1.5 model [56] by adapting its architecture to incorporate action conditioning. Specifically, we add cross-attention blocks to the video generation Diffusion Transformer (DiT), enabling the model to condition video synthesis on robot action trajectories.

We denote the model input as $(x_0, a_{0:T})$, where $x_0 \in \mathbb{R}^{H\times W\times 3}$ is the initial RGB image observation of the environment and $a_{0:T} \in \mathbb{R}^{T\times 7}$ is the sequence of 7-DoF robot arm states corresponding to the T frames of the generated video. For action conditioning, each robot arm state is projected into a single action token with dimension D_a using an input MLP, producing an action-token sequence in $\mathbb{R}^{T\times D_a}$. A 4-layer transformer encoder then processes these tokens along the temporal dimension while preserving the sequence shape, yielding encoded action tokens in $\mathbb{R}^{T\times D_a}$. The resulting action tokens are used as conditioning inputs to cross-attention blocks inserted before the self-attention block in each layer of the video generation DiT, and the video tokens query the encoded action-token sequence through cross-attention. The outputs of the cross-attention blocks are further passed through a projection layer to produce action-dependent modulation signals. These signals are injected into the DiT through the timestep modulation pathway by adding the action modulation to the original timestep modulation. We initialize the model from the pretrained checkpoint and finetune it on all demonstration data described in Table 5. Training uses the standard flow-matching video generation objective adopted during pretraining of the base video model [56]. The model and training hyperparameters are summarized in Table 8.

For the action-conditioned video generation model, we found that sufficient diversity in the random demonstrations is important for enabling the world model to learn a broad range of environment state transitions.

Hyperparameter	Value
Base Model	Pretrained HunyuanVideo-1.5
LoRA	Rank 64 on Pretrained Blocks
Training Precision	bf16
Video Resolution	848×480
Video Length	33 frames
Action Length	33 frames
Optimizer	Muon
Action Blocks Learning Rate	$1e-4$
LoRA Learning Rate	$1e-5$
Weight Decay	0.01
Effective Batch Size	32
Training Steps	4000

Table 8: **Action-conditioned video generation model hyperparameters.**

C.4 VLM critic

We finetune the VLM critic on the collected successful and failure data (dataset details provided in Table 6). The critic in the real world is trained per-task while for simulation we train a single multi-task critic across all 18 atomic tasks. We provide the details of the hyperparameters used to finetune the VLM critic in Table 9. Furthermore, the details on the prompts used per baseline in Figure 7.

Hyperparameter	Value
Base Model	Qwen2.5-VL-3B-Instruct
Fine-tuning	Full (all parameters)
Attention Implementation	FlashAttention-2
Mixed Precision	bf16
Distributed Backend	DeepSpeed ZeRO-2
Number of GPUs	8
Images per Comparison	2
Max Image Resolution	960×540
Optimizer	AdamW
Learning Rate	$1e-5$
LR Schedule	Linear
Warmup Ratio	0.05
Weight Decay	0.0
Max Gradient Norm	1.0
Per-Device Batch Size	4
Gradient Accumulation Steps	2
Effective Batch Size	64
Number Steps	1400
Completion-Only Loss	Yes
Balance Failure vs. Success	Yes

(a) Real World

Hyperparameter	Value
Base Model	Qwen2.5-VL-3B-Instruct
Fine-tuning	Full (all parameters)
Attention Implementation	FlashAttention-2
Mixed Precision	bf16
Distributed Backend	DeepSpeed ZeRO-2
Number of GPUs	8
Images per Comparison	2
Max Image Resolution	848×480
Optimizer	AdamW
Learning Rate	$1.5e-5$
LR Schedule	Linear
Warmup Steps	200
Weight Decay	0.0
Max Gradient Norm	1.0
Per-Device Batch Size	12
Gradient Accumulation Steps	2
Effective Batch Size	192
Number Steps	4500
Completion-Only Loss	Yes
Balance Failure vs. Success	Yes

(b) RoboCasa365 Simulation

Table 9: **VLM Critic Fine-tuning Hyperparameters.**

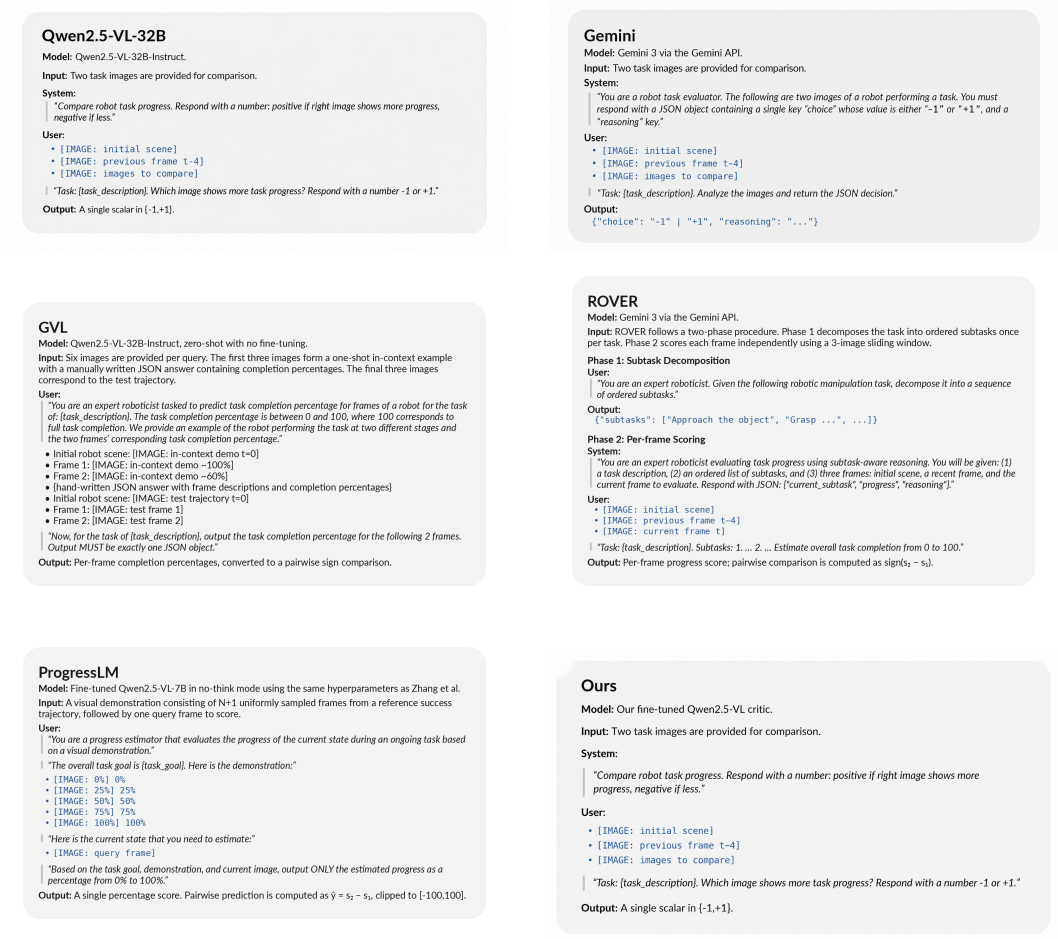


Figure 7: **Prompt templates** used for baselines and our method.