SRPO: Self-Referential Policy Optimization for Vision-Language-Action Models

Supplementary Material

A. Progress Reward Benchmark

Based on the carefully curated multi-task dataset of 700 human-annotated successful trajectories and 300 failure trajectories across diverse task domains, we propose a practical evaluation framework to assess the quality of progress reward functions. All successful trajectories are manually selected to exhibit approximately linear growth characteristics, meaning they demonstrate consistent forward progression without significant regressions or backtracking (e.g., no instances of dropping objects and having to retrieve them). To better test the model's generalization performance, the selected trajectories contain various perturbations including camera viewpoint changes, lighting variations, compounding objects, sensor noise and background distractions. The benchmark focuses on five core metrics that capture the essential properties of effective progress signals across multiple tasks and perturbation scenarios.

A.1. Core Evaluation Metrics

- **Temporal Correlation**: Measures the correlation between progress values and frame numbers across all tasks using Spearman's rank correlation coefficient:

$$\rho = \frac{1}{N}\sum_{k=1}^{N} \frac{\sum_{i=1}^{T_k}(x_i^{(k)} - \bar{x}^{(k)})(y_i^{(k)} - \bar{y}^{(k)})}{\sqrt{\sum_{i=1}^{T_k}(x_i^{(k)} - \bar{x}^{(k)})^2 \sum_{i=1}^{T_k}(y_i^{(k)} - \bar{y}^{(k)})^2}}, \tag{12}$$

  where N is the number of tasks, T_k is the trajectory length for task k, $x_i^{(k)}$ represents frame numbers and $y_i^{(k)}$ represents progress values. Higher absolute values indicate stronger monotonic relationships.
- **Temporal Monotonicity**: Quantifies the average percentage of steps where progress increases across all tasks:

$$M_{\text{mono}} = \frac{1}{N}\sum_{k=1}^{N} \frac{1}{T_k - 1}\sum_{t=1}^{T_k-1} \mathbb{I}(r_{t+1}^{(k)} > r_t^{(k)}), \tag{13}$$

  where $\mathbb{I}$ is the indicator function and $r_t^{(k)}$ is the progress at step t for task k. Values closer to 100 indicate stronger monotonic progression.
- **Distribution Separation (MMD)**: Evaluates the average separation between success and failure trajectories across tasks using Maximum Mean Discrepancy:

$$\text{MMD} = \frac{1}{N}\sum_{k=1}^{N} \left\| \frac{1}{n_k}\sum_{i=1}^{n_k} \phi(R_{s,k}^{(i)}) - \frac{1}{m_k}\sum_{j=1}^{m_k} \phi(R_{f,k}^{(j)}) \right\|_{\mathcal{H}}^2, \tag{14}$$

  where $R_{s,k}$ and $R_{f,k}$ represent final progress values for success and failure trajectories in task k, n_k and m_k are the respective sample sizes, and ϕ is the feature map in reproducing kernel Hilbert space $\mathcal{H}$.
- **Jensen-Shannon Divergence**: Measures the average distributional divergence between success and failure trajectories across tasks:

$$\begin{aligned}\text{JSD} = \frac{1}{N}\sum_{k=1}^{N}\Bigg[&\frac{1}{2} D_{\text{KL}}\left(P_{\text{success}}^{(k)} \| M^{(k)}\right) \\ &+ \frac{1}{2} D_{\text{KL}}\left(P_{\text{failure}}^{(k)} \| M^{(k)}\right)\Bigg],\end{aligned} \tag{15}$$

  where $M^{(k)} = \frac{1}{2}(P_{\text{success}}^{(k)} + P_{\text{failure}}^{(k)})$ is the mixture distribution for task k, and D_{KL} denotes the Kullback-Leibler divergence.
- **Standardized Mean Difference**: Measures the average separation effect size between success and failure trajectories across tasks:

$$\text{SMD} = \frac{1}{N}\sum_{k=1}^{N} \frac{\mu_{\text{success}}^{(k)} - \mu_{\text{failure}}^{(k)}}{\sigma_{\text{pooled}}^{(k)}}, \tag{16}$$

  where $\mu_{\text{success}}^{(k)}$ and $\mu_{\text{failure}}^{(k)}$ are the means for task k, and $\sigma_{\text{pooled}}^{(k)} = \sqrt{\frac{(n_k-1)(\sigma_{\text{success}}^{(k)})^2 + (m_k-1)(\sigma_{\text{failure}}^{(k)})^2}{n_k+m_k-2}}$ is the pooled standard deviation for task k.

A.2. Evaluating Progress Reward Quality

A high-quality progress reward function should exhibit several key characteristics that can be assessed through the following complementary metrics:

- **Temporal Consistency**: The progress values should demonstrate strong *temporal correlation* (ρ) with monotonically increasing patterns (*temporal monotonicity* M_{mono}). Effective progress rewards should show consistent improvement over time, with correlation values approaching 1.0 and monotonicity scores near 100% indicating smooth, predictable progression toward task completion.
- **Distribution Discriminability**: Successful and failed trajectories should be well-separated in the progress reward space. This is quantified through multiple complementary measures:
  - *Maximum Mean Discrepancy (MMD)*: Captures distributional differences in reproducing kernel Hilbert spaces, with larger values indicating better separation between success and failure trajectories.
  - *Jensen-Shannon Divergence (JSD)*: Measures the information-theoretic divergence between success and

failure distributions, where values closer to $\ln 2$ suggest maximal discriminability.

– *Standardized Mean Difference (SMD)*: Provides an effect size measure for the separation between success and failure means, with larger absolute values indicating stronger differentiation.

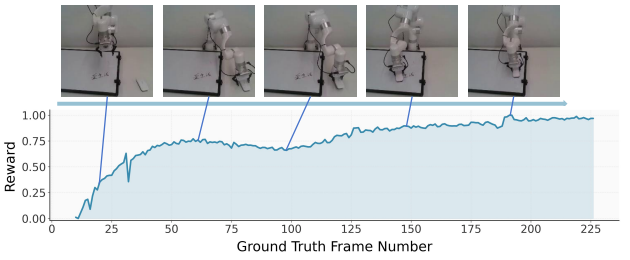
The combination of these metrics provides a comprehensive framework for evaluating progress reward quality, emphasizing both the internal consistency within successful trajectories and the external discriminability between success and failure outcomes.

B. Reward Analysis Details

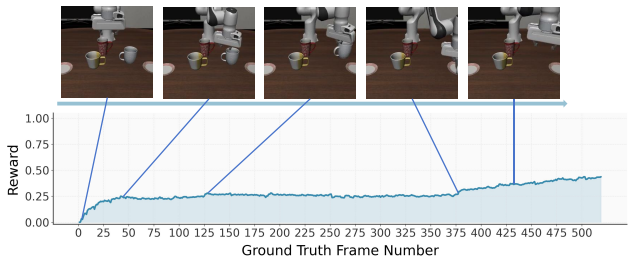
Reward construction for success traj. To ensure a fair and standardized comparison, we adopt the following normalized procedure for progress reward computation: For SRPO and ImageBind, we compute video embeddings using a cumulative sliding window approach: starting from frames 0-10, we progressively extend the window through frames 1-11, 1-12, and so forth, until the final window spanning frames 1 to the penultimate frame, which generates a sequence of embeddings representing cumulative visual context up to each time step. Then, we compute their L2 distances to the embedding of the entire video sequence, and normalize the progress reward by assigning a value of 0 to the frame with the maximum distance and 1 to the frame with the minimum distance. For the pixel-level method (RLVR), we calculate the L1 distance between each frame from frame 10 to the penultimate frame and the final frame, applying the same normalization scheme.

Reward construction for failure traj. For failure trajectories, we use the cluster centers of successful trajectories as reference points. We compute the minimum L2 distance from each failure trajectory segment to these success cluster centers, then normalize the progress reward by assigning a value of 0 to the segment with the maximum distance and 1 to the segment with zero distance (closest to success patterns).

Reward Curve Analysis. We randomly selected some successful and failure trajectories, plotting frame number-progress reward curves to compare our approach with two baseline methods. As shown in Figure 15 16 17, our analysis reveals that pixel-level rewards fail to properly evaluate long-horizon tasks with multiple sub-tasks and tend to exhibit sharp progress increases only in the final few frames. While general visual encoders like ImageBind can capture trajectory-level features, these features lack physical intuition, resulting in oscillatory progress rewards that are non-smooth and frequently display incorrect sudden spikes or drops—characteristics particularly unfriendly for reinforcement learning. In contrast, our reward method demonstrates more reasonable and stable progress estimation. We also provide detailed reward curves for two trajectories as visualizations in Figure 8.



(a) Success Trajectory. Task: Wipe the board.



(b) Faliure Trajectory. Task: Put the white mug on the left plate and put the yellow and white mug on the right plate.

Figure 8. Visualization of reward signals between successful and failure trajectories. (a) A success trajectory (task: wipe the board) exhibits a reward curve that rises smoothly; a slight dip in reward corresponds to a pause during grasping. (b) A failure trajectory (task: put mugs on plates) shows a stagnant reward signal, failing to reach a high value, as the model fails to locate the second mug.

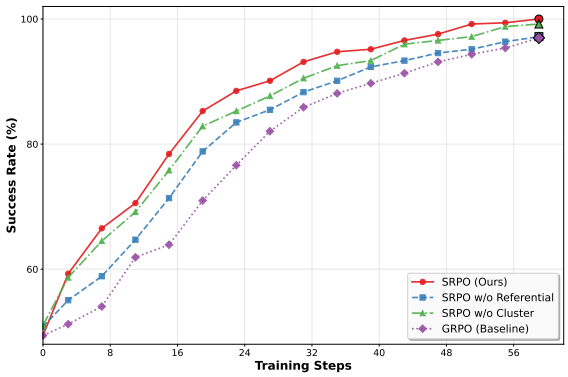


Figure 9. Ablation study on LIBERO Object Task. We compare our full SRPO method against its ablated variants and the GRPO baseline. Removing the referential component (w/o Referential) leads to significant performance drop, while removing the clustering component (w/o Cluster) slows down convergence. Our complete model achieves the best performance and training efficiency.

C. Ablations

C.1. Ablation on Self-Referential Mechanism

A core design of our method is its self-referential nature, which assesses progress by comparing within the current pol-

icy's own rollouts. We hypothesize that relying on a fixed set of external expert trajectories could constrain the policy's exploration and potentially lead to convergence to suboptimal local minima. To validate this, we ablate the self-referential mechanism by replacing the within-batch successful trajectories with a fixed set of 50 pre-selected expert trajectories per task for progress computation. As illustrated in Figure 9, the ablated variant initially trains slightly slower than our full SRPO method, yet still outperforms GRPO significantly. However, its performance plateaus in later stages, requiring nearly 1.4 times the training steps yet still yielding suboptimal results. This suggests two key insights: (1) The *progress-wise* reward, even when computed from external trajectories, provides a more effective learning signal than sparse binary rewards, explaining the initial efficiency gain over GRPO, while the introduction of external information eventually allows GRPO, which requires no such information, to catch up it in performance. (2) The fixed external references ultimately limit the policy's capacity for open-ended exploration. As the policy evolves, these static trajectories fail to provide nuanced, step-by-step progress assessments for the diverse rollouts generated by the policy, leading to a performance ceiling lower than that achieved by our self-referential approach.

C.2. Ablation on Success Clustering

We introduce clustering of successful trajectories to compute progress rewards based on two primary motivations. First, a task can often be accomplished through multiple distinct strategies (e.g., placing object A before B, or vice versa). A failure trajectory should be compared to the nearest *successful strategy* rather than an arbitrary one. Second, using the centroid of a cluster, rather than the single nearest successful trajectory, provides a more robust distance measure. Individual success trajectories might contain sub-optimal or noisy segments (e.g., a gripper moving momentarily away from an object before successfully grasping it). If a failure trajectory shares a similar initial deviation, measuring the distance solely to this specific, noisy success trajectory could yield an inaccurately high progress score. Using a cluster centroid mitigates this by representing a more prototypical and cleaner version of a success strategy.

To substantiate these points, we conduct an ablation where progress is computed using the distance to the single nearest success trajectory instead of the cluster centroid. The results are presented in Figure 9. We observe that this variant's initial learning efficiency is comparable to our full SRPO method. However, its performance gains diminish significantly as training progresses. We attribute this to the following: in early training stages, the repertoire of successful strategies is limited, and the number of success trajectories is small, thus diminishing the advantage of clustering. In later stages, as successful exploration diversifies and the number of success trajectories grows substantially, the ability of clustering to distill prototypical strategies and provide robust progress signals becomes crucial, leading to the increasing performance gap observed in our results.

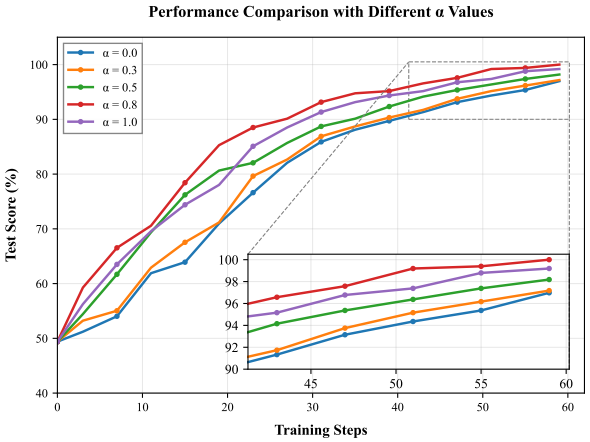


Figure 10. Performance comparison with different α values in the reward function. The results demonstrate that $\alpha = 0.8$ achieves the best performance, followed by $\alpha = 1.0$, $\alpha = 0.5$, $\alpha = 0.3$, and $\alpha = 0$ in descending order. This validates the importance of balancing progress awareness with outcome correctness in our reward design.

D. Hyperparameter Analysis

In Equation (5), we employ an activation function to map the reward trajectory into the range (0, 1). In our implementation, we use a sigmoid function and precede it with a scaling coefficient α to control the trade-off between progress awareness and outcome correctness in our reward function. We now conduct a comprehensive analysis of this hyperparameter α. The reward design provides full credit (1.0) for correct answers and scales the progress reward by α otherwise. We evaluate five different values of α: 0, 0.3, 0.5, 0.8, and 1.0.

The experimental results reveal a clear performance hierarchy: $0 < 0.3 < 0.5 < 1.0 < 0.8$. This pattern provides several important insights:

- $\alpha = 0$ **(no progress reward)**: Performs worst, confirming that purely outcome-based rewards are insufficient for complex tasks requiring sequential reasoning.
- $\alpha = 0.3$ and 0.5: Show gradual improvement, indicating that even small progress rewards enhance learning efficiency.
- $\alpha = 1.0$ **(equal weighting)**: Performs better than lower values but suboptimally, suggesting that over-emphasizing progress may distract from the final objective.
- $\alpha = 0.8$ **(optimal)**: Achieves the best performance, demonstrating that a strong but not exclusive focus on progress rewards (80% of maximum) provides the ideal

balance for guiding the learning process while maintaining focus on task completion.

This analysis validates our reward design principle: properly weighted progress awareness ($\alpha = 0.8$) significantly outperforms both purely outcome-based rewards ($\alpha = 0$) and excessively progress-focused rewards ($\alpha = 1.0$). The optimal α value enables the agent to benefit from intermediate guidance while remaining oriented toward the ultimate task objective.

E. What if we use a pixel-level world model for reward shaping?

Another promising avenue for reward shaping involves leveraging pixel-level world models to generate reference trajectories based on language instructions or action priors. Since these priors are already inputs to the policy or RL algorithm, this approach does not strictly constitute introducing external information. In this section, we use Cosmos-Predict2 [1] as a case study to elucidate why this paradigm, while intuitive, often falls short in practice.

Zero-Shot Generation Yields Unsatisfactory Results. We experimented with the large-scale Cosmos-Predict2-14B model in a zero-shot setting on several tasks from the LIBERO benchmark. Using the language instruction as conditioning signals, the model was tasked with generating a reference video trajectory. As shown in Figure 18, the generated videos suffer from poor scene consistency. Furthermore, although task-specific supervised fine-tuning (SFT) may mitigate these inconsistencies and improve generation quality, this high-cost approach, which relies heavily on extensive expert demonstrations, is clearly inferior to the reward modeling paradigm proposed by SRPO, which is based on a latent world representation and proves to be more cost-effective and generalizable.

F. Training Details

F.1. Supervised Fine-Tuning (SFT) Stage

Our one-shot SFT phase builds upon the OpenVLA (with Action Chunking and Parallel Decoding) checkpoint. The training was conducted on $8 \times$ A100 GPUs. Key training configurations are as follows:

- **Optimizer:** AdamW
- **Learning Rate:** 5×10^{-4}
- **Batch Size:** 8
- **Maximum Training Steps:** 150,005
- **Learning Rate Decay:** Applied after 100,000 steps
- **LoRA Rank:** 32
- **Image Augmentation:** Enabled
- **Input Modalities:** Text instruction and 3rd image
- **Action Chunking:** 8 action chunks

F.2. SRPO Post-Training Stage

The SRPO reinforcement learning phase is initialized from the one-shot SFT model. Key hyperparameters include:

- **Algorithm:** SRPO
- **Learning Rate:** 5×10^{-6}
- **Samples Per Group:** 8
- **Batch Size:** 64 ($\times 8$, training), 496 (validation)
- **Mini-batch Size:** 128
- **Progress Reward Weight:** 0.8
- **Number of Trials per Task:** 50
- **Action Configuration:** 7 action tokens, 8 action chunks
- **Trajectory Mini-batch Size:** 16
- **Log Probability Batch Size:** 8

F.3. Model Details

This modified architecture retains the Llama 2 backbone for generating discrete action tokens, in contrast to the continuous action heads employed in OpenVLA-OFT. While this design choice may potentially sacrifice some level of action precision compared to continuous output methods, it provides the crucial advantage of enabling direct access to action log-probabilities essential for policy gradient methods in reinforcement learning.

G. Real-world Experiment Details

G.1. Detailed Setup

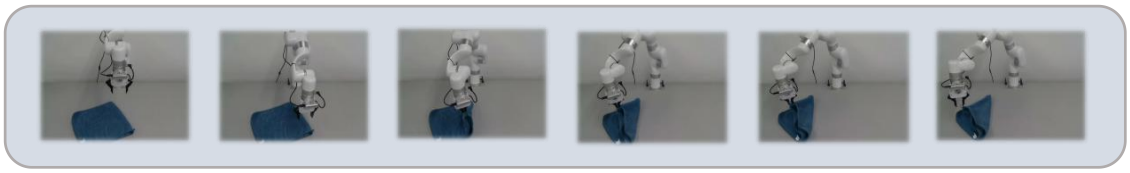
In our physical robot experiments, due to safety concerns associated with online exploration and the significant time cost of manual resets, we adopted an offline reinforcement learning (RL) paradigm. Specifically, our technical approach integrates the Advantage-Weighted Regression (AWR) strategy [33] with SRPO's self-referential progress-wise reward and advantage mechanism. We first collect demonstration data and store it in a trajectory buffer. The expected cumulative reward (or value) $R_{i,t}$ for the i-th trajectory at step t is computed as in (2)–(5). We then define the incremental progress at step t as $D_{i,t} = R_{i,t} - R_{i,t-1}$. Following the SRPO framework, the advantage function is calculated as:

$$A_{i,t} = \frac{D_{i,t} - \mu}{\sigma}, \tag{17}$$

where μ and σ are the mean and standard deviation of $D_{i,t}$ computed across all trajectories. Our real-world task policy is trained from scratch via RL starting from pretrained weights, with the baseline for comparison being the SFT model.

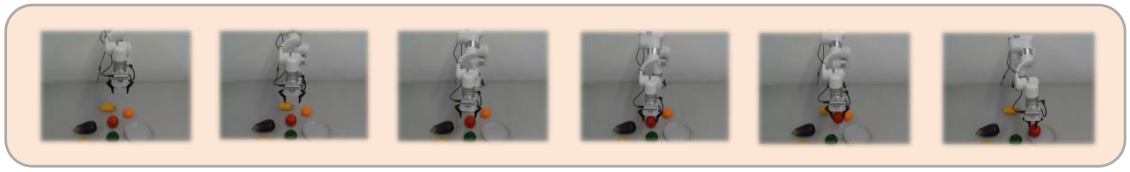
The pick-and-place tasks include *put apple into the plate* and *put pear into the plate*. To test robust object identification, we place multiple fruits on the table initially, requiring the model to distinguish target objects from distractors. During task execution, we randomly swap positions between

(a) Put the pear into the plate.

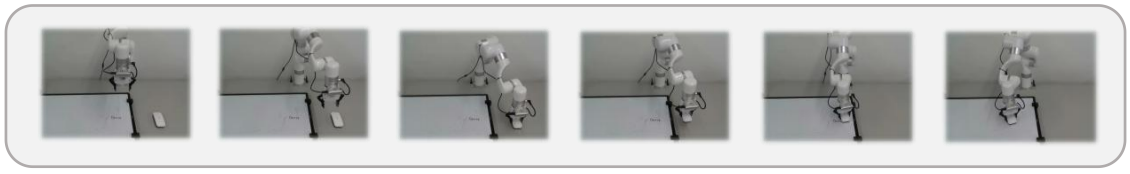


(b) Fold the towel.

(c) Pick up the Joker.



(d) Put the apple into the plate.



(e) Wipe the board.

Figure 11. Success cases of the real-world experiment.

target and distractor objects to assess the model's capacity for rapid adaptation.

To specifically demonstrate the advantages of dense progress-aware rewards in complex manipulation, we incorporate *folding towels* involving deformable object manipulation and *cleaning whiteboard* requiring coordinated surface contact. Furthermore, to evaluate semantic understanding capabilities, we include the *select poker* task where the model must identify the specified card from these five playing cards: Joker, Jack of Spades, King of Club, Jack of Spades, 10 of Spades.

Our real-world experimental results demonstrate significant performance improvements across all five manipulation tasks when applying progress-aware value weighting. As shown in Figure 7, both VLA policy backbones exhibit substantial gains over their SFT counterparts.

G.2. Rewarding Validation

To quantitatively validate the generalization capability of our progress reward model to real-world scenarios, we evaluate our pre-trained progress reward function on five real-robot manipulation tasks, each containing 700 successful trajectories and 300 failure trajectories. The comprehensive Progress Reward Benchmark results across all tasks demonstrate consistent performance in real-world settings:

As shown in Table 4, the consistently high Progress Reward Quality scores across all five real-robot tasks confirm that our latent progress reward modeling effectively captures task progression dynamics in real-world robotic manipulation. With near-perfect Spearman correlation and monotonicity, coupled with strong distribution separation in MMD and SMD, our method demonstrates robust generalization despite domain shifts from simulation to reality.

Table 4. Progress Reward Benchmark results on real-robot datasets. Our method maintains strong performance across all tasks and metrics, demonstrating robust generalization to diverse real-world manipulation tasks.

Task	SC	Mono	MMD	JS	SMD
Put Apple	0.987	0.975	0.589	0.562	165.3
Put Pear	0.991	0.982	0.601	0.578	172.8
Fold Towel	0.984	0.968	0.572	0.549	158.6
Wipe Board	0.993	0.986	0.624	0.591	181.2
Select Poker	0.989	0.979	0.595	0.569	169.5
Average	0.989	0.978	0.596	0.570	169.5

G.3. Case Study

As shown in Figure 11, we evaluate our method on three real-world tasks to assess its robustness and generalization. The results across these diverse scenarios are as follows:

- In *put the pear into the plate*, we test the policy's response to a dynamic environment by moving the target plate during execution. The policy pauses, re-locates the plate, and executes a new trajectory to complete the task, demonstrating its capability for dynamic replanning.
- In *fold the towel*, the policy is applied to a deformable object. The results show that the policy can successfully complete the folding sequence, indicating that the method can handle the continuous state changes associated with deformable object manipulation.
- In *pick up the Joker card*, the policy is required to identify a specific card among multiple similar ones. The successful completion of this task shows that the policy can utilize semantic information to distinguish between visually similar objects.

These results indicate that our method can adapt to a range of challenges, including dynamic environments, deformable objects, and the need for semantic discrimination, confirming its functionality in real-world settings.

H. Policy Exploration and Novel Strategy Acquisition

To analyze the policy's exploratory behavior, we performed rollouts on a set of tasks and visualized a subset of the resulting end-effector trajectories. The trajectory points were obtained by recording the end-effector position (eef pos) from the MuJoCo environment. These spatial coordinates were then projected onto the image plane using the camera's intrinsic and extrinsic parameters to derive the corresponding pixel coordinates for visualization.

As illustrated in Figure 12, even when the policy is initially exposed to only a single successful demonstration trajectory, the subsequent online RL fine-tuning enables it to discover novel strategies beyond the provided example.

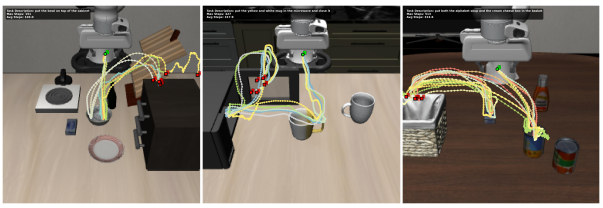


Figure 12. Visualization of end-effector trajectories across three tasks (from left to right): *put the bowl on top of the cabinet*, *put the yellow and white mug in the microwave and close it*, and *put both the alphabet soup and the cream cheese box in the basket*. Taking the first task as an example, the SRPO policy discovers novel successful placement locations for the bowl as well as alternative grasping points.

This exploration is evident not only in the diversity of spatial paths taken to approach the object but also in the variety of grasping positions it discovers. This finding underscores a key strength of our method: the ability to autonomously acquire and leverage novel knowledge, in terms of both motor skills and affordance understanding.

I. More Tasks

We further add dual-arm cloth folding (flexible obj, over 2k steps) and ingredient washing tasks (sequentially transporting ingredients and toggling water on/off), where SRPO outperforms SFT (Fig. 13) by effectively down-weighting suboptimal actions.

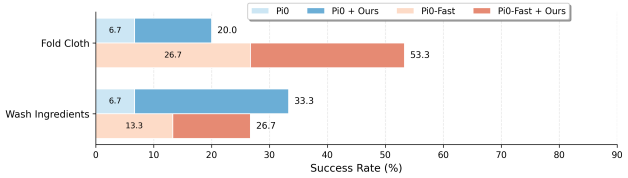


Figure 13. Real-world success rates on complex tasks.

J. Comparison with Other Reward Models

We evaluated SRPO across five datasets, finding it consistently outperforms GVL [31] and zero-shot VLAC [51], especially on human data (Fig. 14), which we attribute to the effective modeling for full trajectory dynamics of SRPO.

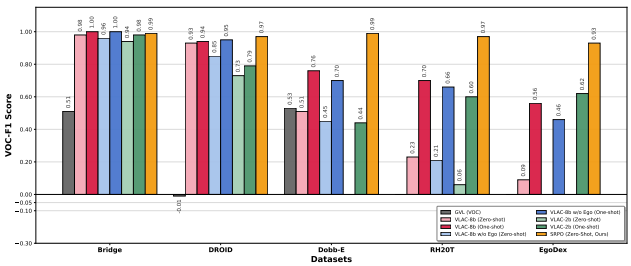
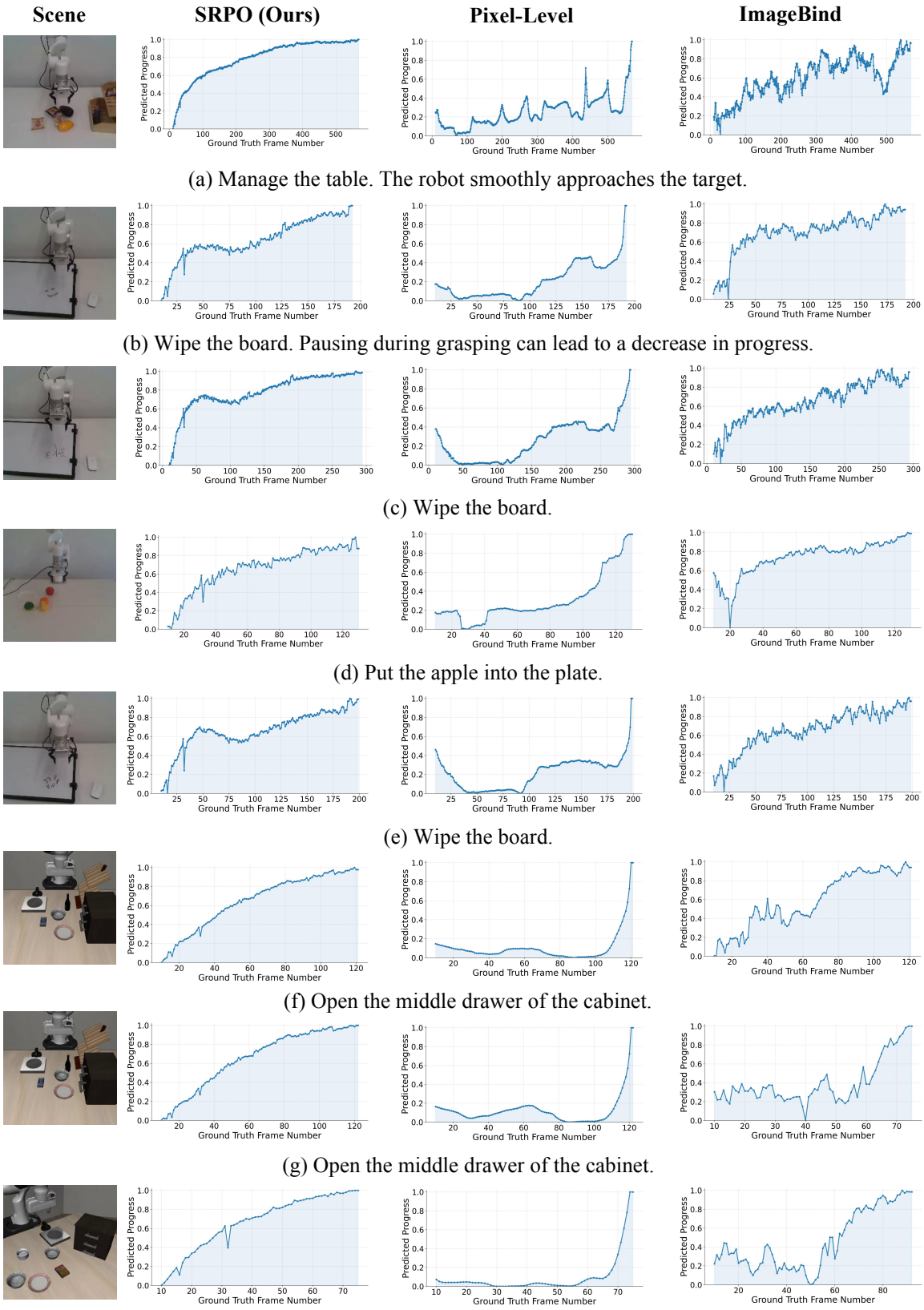


Figure 14. Reward curves for complex tasks.



(a) Manage the table. The robot smoothly approaches the target.

(b) Wipe the board. Pausing during grasping can lead to a decrease in progress.

(c) Wipe the board.

(d) Put the apple into the plate.

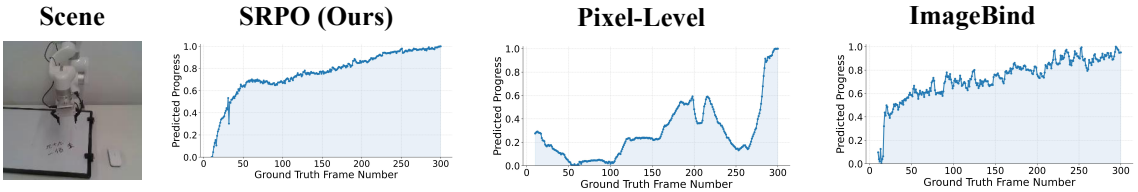
(e) Wipe the board.

(f) Open the middle drawer of the cabinet.

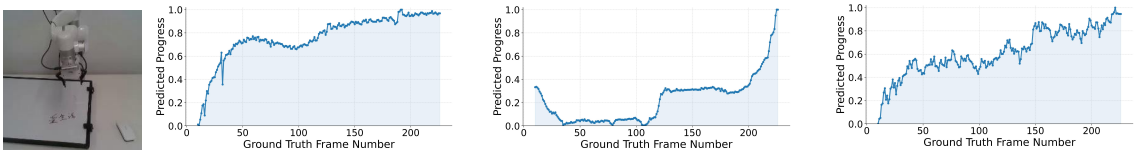
(g) Open the middle drawer of the cabinet.

(h) Pick up the black bowl from table center and place it on top of the cabinet.

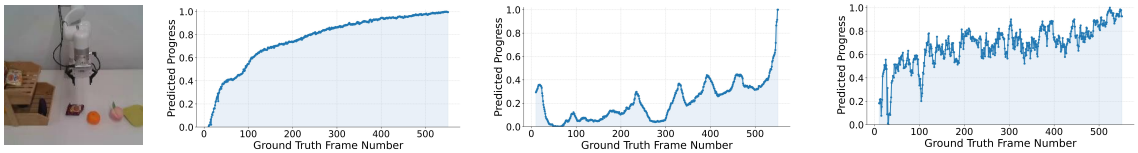
Figure 15. Reward curves of different methods on successful trajectories (Part One).



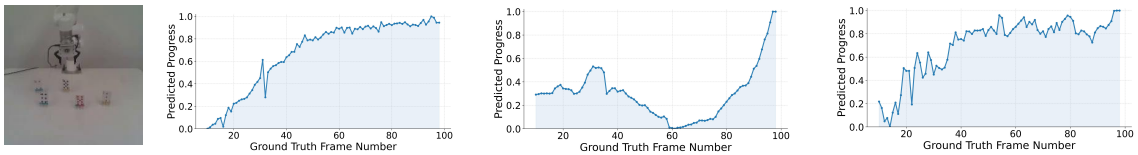
(a) Wipe the board. During moving the eraser, there was a process of dropping and picking it up again.



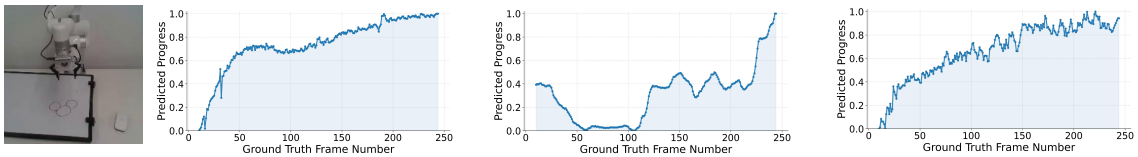
(b) Wipe the board. We found that pausing during grasping can lead to a decrease in progress, as the model might consider this action to be unrelated to the final goal.



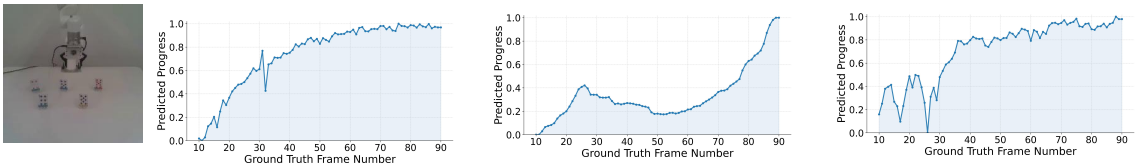
(c) Manage the table. The robot smoothly approaches the target.



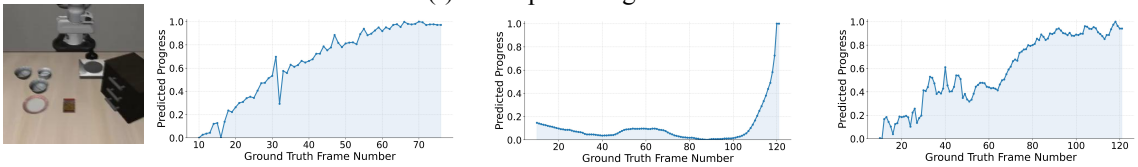
(d) Pick up the joker.



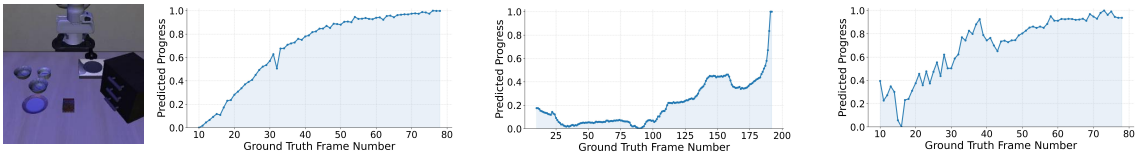
(e) Wipe the board.



(f) Pick up the king of club.

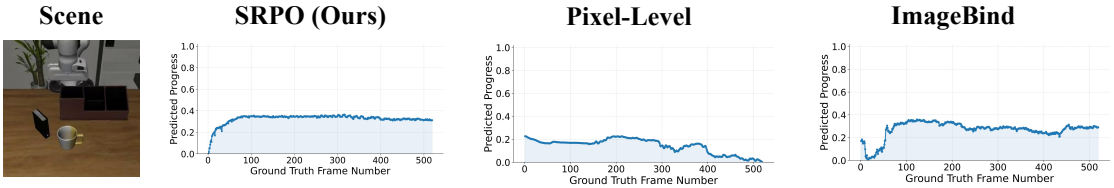


(g) Open the middle drawer of the cabinet + Sensor Noise

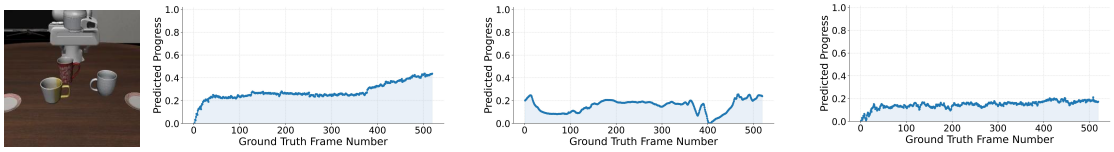


(h) Pick up the black bowl from table center and place it on top of the cabinet + Light Perturbation

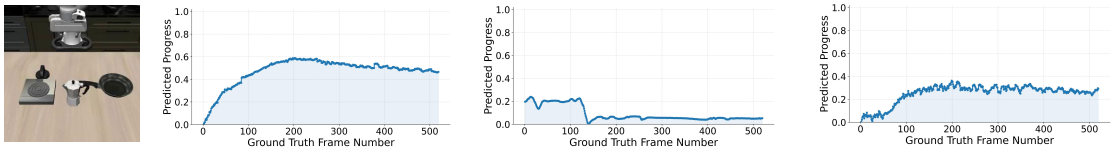
Figure 16. Reward curves of different methods on successful trajectories (Part Two).



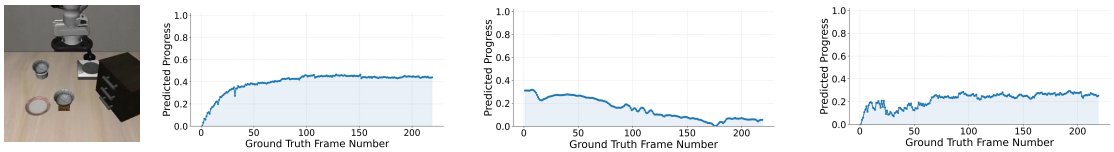
(a) Pick up the book and place it in the back compartm: Failed at step 1 of 2.



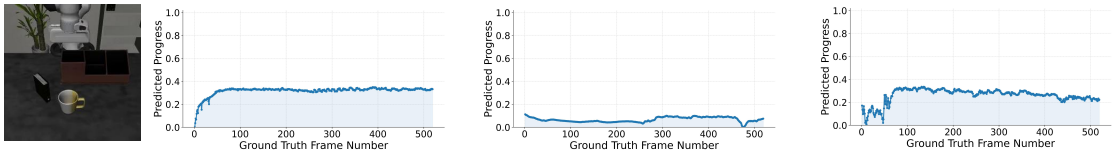
(b) Put the white mug on the left plate and put the yellow mug on the right plate: Finished step 1 of 2.



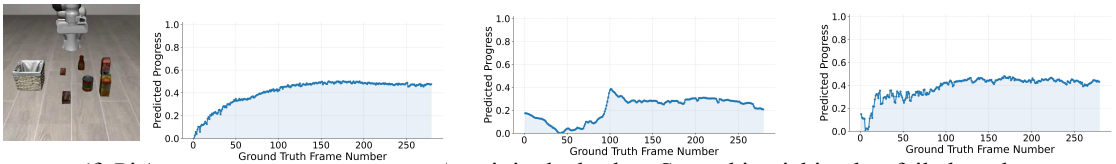
(c) Turn on the stove and put the moka pot on it: Finished step 1 of 2.



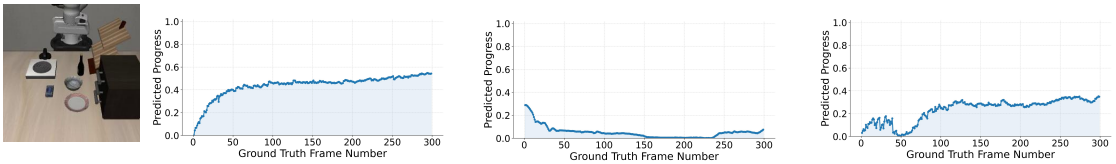
(d) Pick up the black bowl on the ramekin and place it on top of the cabinet: Failed at step 1 of 2 as the bowl slid off the clamp but the end effector still moved above the cabinet.



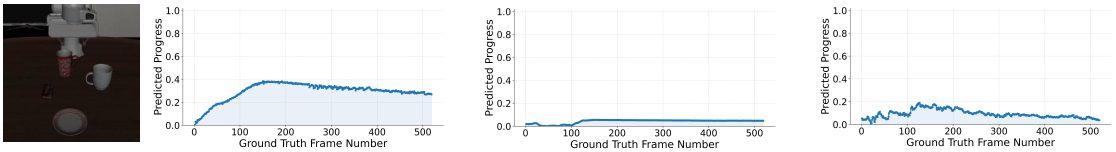
(e) Pick up the book and place it in the back compartm: Succeed in reaching the book but fail to pick up.



(f) Pick up the tomato sauce and place it in the basket: Suceed in picking but failed to place.



(g) Push the plate to the front of the stove: Pushed the plate to a wrong position.



(h) Put the white mug on the plate and put the chocolate pudding to the right of the plate: mug has been put on the proper position but was knocked over when being released.

Figure 17. Reward curves of different methods on failure trajectories.

(a) Pick up the book and place it in the back compartment of the caddy.

(b) Put both moka pots on the stove.

(c) Put the black bowl in the bottom drawer of the cabinet and close it.

(d) Put both the alphabet soup and the cream cheese box in the basket.

(e) Put the yellow and white mug in the microwave and close it.

(f) Manage the table.

(g) Wipe the board.

Figure 18. Trajectories generated by Cosmos-Predict2 [1].