PALM: Progress-Aware Policy Learning via Affordance Reasoning for Long-Horizon Robotic Manipulation

Supplementary Material

A. Implementation Details

A.1. PALM Model Details

Vision. Visual encoding employs a MAE-pretrained ViT-B [36] backbone, which serves as the primary feature extractor. For each timestep, the model processes images from two viewpoints: a static eye-on-base camera for global context and an eye-on-hand camera for local, gripper-centric views. The ViT transforms each image into 196 patch embeddings plus a [CLS] token. To maintain computational tractability over long temporal sequences, we employ a Perceiver Resampler [46]. This module uses a set of learnable latent vectors and cross-attention to distill the initial 197 high-dimensional tokens into a compact, fixed-size set of task-relevant visual embeddings, which are then fed to the main backbone.

Text. To ground the policy in natural language, we encode task instructions using a pretrained CLIP text encoder [102]. This module converts the input instruction into a fixed-length embedding that captures its semantic intent. A subsequent linear projection maps this embedding into the model's shared latent space, allowing effective integration with visual and state information.

Robot State. The model receives proprioceptive feedback describing the robot's physical configuration. This state is represented by a six-dimensional vector for the end-effector's 6-DoF Cartesian pose (position and Euler angles) and a binary value for the gripper's open/closed status. For processing, the binary gripper value is first transformed into a two-dimensional one-hot vector. Both the 6-D pose and the one-hot gripper vector are projected through separate linear layers, then concatenated and passed through a final MLP to generate a single, unified state token.

Learnable Queries. We introduce two sets of learnable query tokens that are appended to the multimodal token sequence at each step t and updated inside the transformer via cross-attention under causal masking. Each query set extracts and integrates information from multimodal inputs to enable joint prediction.

- **Fine-grained affordance queries** are specialized tokens that extract a structured view of future interaction. Organized into four sub-queries (`<Global>`, `<Local>`, `<Spatial>`, `<Dynamic>`), they attend to the multimodal context to produce a disentangled, affordance-centric latent $\hat{\mathbf{F}}_{t+n}$ that guides downstream control.
- **Action–progress queries** generate the final control sequence. They pool control-relevant information from current observations and explicitly condition on $\hat{\mathbf{F}}_{t+n}$, enabling a progress-aware inverse-dynamics formulation so actions remain temporally consistent and aligned with the predicted future state.

Backbone. The core of our model is a GPT-2 style transformer [101], which functions as the central multi-modal reasoning engine. It takes as input a concatenated sequence containing the encoded vision, text, and state tokens, alongside the learnable affordance and action-progress queries. By applying causal and cross-modal attention, the backbone fuses these diverse inputs into a coherent latent representation of the world state.

Decoders. We preserve spatial correspondence by adding fixed 2D sine–cosine positional encodings to the image patch tokens and propagating these coordinates through a stack of Transformer encoder layers. After this shared encoding, modality-specific heads decode the four affordance latents. `<Global>` and `<Spatial>` use lightweight 2-layer MLP heads, producing a future object mask and a set of candidate placement points. `<Local>` and `<Dynamic>` use 2-layer Transformer blocks followed by linear projections to produce a future contact heatmap and a dynamic region. These targets are defined at $t + n$ to supervise the affordance latents and are used only during training. The primary model output is produced by the action–progress decoder, which jointly predicts actions and scalar progress. Configuration details for each module are given in Table 6.

Prediction with Diffusion Transformer. We formulate action–progress generation as a conditional denoising task and employ a Diffusion Transformer (DiT) decoder [96]. The decoder conditions on the latent embedding from the action–progress queries, which already integrates the predicted affordance information. By iteratively reversing a Gaussian noise process over a sequence of latent vectors, the DiT models complex, multimodal action distributions and yields a temporally coherent trajectory. It produces a joint sequence of 7-DoF actions and the corresponding scalar progress values associated with subtask completion. Configuration details of the DiT are provided in Table 7.

A.2. Training Details

Datasets. Our training process consists of a pre-training and a fine-tuning stage. For pre-training, we utilize a mixed dataset from two domains. One part is drawn from the DROID [54] dataset and BridgeData V2 [113], which together provide large-scale, in-the-wild robotic arm demonstrations to build a foundational understanding of diverse real-world tasks. Another part is from EPIC-KITCHENS [20] and RoboCerebra [33], which provides fine-grained sub-steps and time-segment annotations to su-

Table 6. **Key parameters of each module in PALM.**

Type	Hidden Size	Number of Layers	Number of Heads
Image Encoder	768	12	12
Perceiver Resampler	768	3	8
GPT-2 (LLM Backbone)	384	24	12
Global Decoder	384	2	16
Local Decoder	384	2	16
Spatial Decoder	384	2	16
Dynamic Decoder	384	2	16

Table 7. **Configuration of the Diffusion Transformer** used for action-progress prediction.

Parameter	Value
Hidden Size	384
Number of Layers	12
Number of Heads	12
Sampling Steps	10
Noise Schedule	Cosine
Action Prediction Steps	3
Loss Function	MSE (L_2 loss)
Precision	fp32

pervise the learning of semantic progress estimation in long-horizon, contact-rich scenarios. To keep storage and computation requirements manageable during pre-training, the model predicts entire frames rather than fine-grained affordances. For the fine-tuning stage, we select 942 trajectories from robot data and annotate them with affordance data and continuous progress labels using a semi-automated method. We then fine-tuned the model on these annotated trajectories to learn conditional affordance foresight for inverse dynamics prediction, ultimately outputting an action-progress pair. For the post-training stage, we adopt dataset-specific schedules. For LIBERO [77], we pre-train on LIBERO-90, which contains fully annotated demonstrations for 90 short-horizon tasks, and then fine-tune and evaluate on LIBERO-LONG, which features long-horizon tasks. For CALVIN ABC→D [88], we first pre-train on the official robot play data without language instructions; the remaining language-conditioned data with full annotations is used for fine-tuning.

Progress Labels. We manually annotate functional keyframes per trajectory (*e.g.*, *Grasp-Contact*, *Release*) and assign fixed semantic progress anchors; intermediate frames are linearly interpolated. In video pre-training, progress is assigned from *observable interaction milestones* using the same anchors as in robot trajectories. Since these milestones are shared across human video and robot rollouts, they define a unified scale that ensures progress labels remain reliable during transfer, allowing PALM to disambiguate task stages regardless of domain or horizon length.

Hyperparameters. We perform training on 8 NVIDIA A100 GPUs, and set an initial learning rate of 1e-3, a weight decay of 1e-4, and a batch size of 64. Throughout the entire training process, the pre-trained visual and text encoders are kept frozen. All models are trained for a total of 30 epochs. Our model has 68 million trainable parameters. This lightweight architecture offers a compact and flexible base for fine-tuning, allowing it to be performed on smaller GPUs such as the RTX 4090. Details are illustrated in Table 8.

Table 8. **Training hyperparameters.**

Hyperparameters	Pre-training	Fine-tuning
Number of GPUs	8	8
Batch Size	80 / GPU	64 / GPU
Learning Rate	1e-4	1e-3
Weight Decay	1e-4	1e-4
Optimizer	AdamW	AdamW
Learning Rate Schedule	Cosine decay	Cosine decay
Training Epochs	30	40
Historical Sequence Length	7	7
Action Prediction Length	3	3

A.3. Policy Roll-out Details

To ensure efficiency during inference, we set the number of DiT diffusion steps to 10, the number of observation steps to 7, and the number of future prediction steps to 3. This configuration results in a rapid sampling time of approximately 40 ms. Our approach uses fine-grained affordances for conditional visual foresight, which avoids the need for explicit image decoding. This enables a closed-loop frequency of 10-15 Hz, with each decision cycle taking less than 80 ms.

B. Long-Horizon Real-World Task Details

B.1. Task Setup and Success Criteria

We design a long-horizon manipulation task to evaluate the model's ability to follow complex, sequential instructions. The experimental setup consists of a UFACTORY xArm 6 robot arm equipped with a Gripper G2. Visual perception is provided by two RealSense D455 cameras: one mounted on the robot's wrist for an eye-in-hand perspective, and another positioned statically for a third-person, eye-on-base view. The task is driven by a single, high-level language

instruction: "Put the pineapple on the white plate, the grape on the white bowl, and the orange on the blue bowl." The scene contains three toy fruits (pineapple, grape, orange), two white containers (a plate and a bowl), and one blue bowl. For each trial, the initial positions of all objects are randomized within a predefined workspace to test policy robustness. The full task is considered a success only when all three fruits are correctly placed in their designated containers as specified by the instructions. The success criteria for each of the six subtasks are detailed below.

◇ **Subtask 1. Pick up pineapple:** The robot must successfully grasp the toy pineapple from its initial randomized location. Success for this subtask is defined as the robot securely gripping the pineapple and lifting it clear of the table surface without dropping it. This initial step challenges the policy's ability to generalize its grasping strategy to objects with irregular shapes and textures, evaluating its local geometric reasoning for identifying stable contact points.

◇ **Subtask 2. Place on white plate:** The robot must transport the grasped pineapple and place it onto the white plate. Success is achieved if the pineapple is fully supported by the plate and remains stable after the gripper retracts. This step tests both precise spatial targeting and the model's capacity for language-based disambiguation, as it must correctly identify the "plate" from two similar white containers.

◇ **Subtask 3. Pick up grape:** The robot must successfully grasp the toy grape from its initial position. The success criteria are identical to Subtask 1. This action requires fine-grained motor control, evaluating the policy's precision when interacting with small objects where the margin for positional error is minimal.

◇ **Subtask 4. Place on white bowl:** The robot must transport the grasped grape and place it inside the white bowl. Success is achieved if the grape is fully contained within the bowl after the gripper retracts. This subtask again evaluates language disambiguation ("bowl" vs. "plate") and tests the model's understanding of different placement affordances, specifically placing an object *into* a concave container versus *onto* a flat surface.

◇ **Subtask 5. Pick up orange:** The robot must successfully grasp the toy orange from its initial position. The success criteria are identical to Subtask 1. This action serves as a baseline evaluation of the model's core grasping capability on a simple, regular object shape.

◇ **Subtask 6. Place on blue bowl:** The robot must transport the grasped orange and place it inside the blue bowl. Success is achieved if the orange is fully contained within the bowl after the gripper retracts. This explicitly tests the model's language grounding for object attributes, as it must correctly associate the color "blue" with the appropriate target container.

B.2. Robustness of Progress Estimates

To evaluate the reliability and physical grounding of the learned self-progress indicator p_t, we subject a representative subtask to three categories of dynamic perturbations injected twice per episode at random timesteps. Specifically, we test: ① random object relocations, *i.e.*, changing the target's pose or position mid-execution to require geometric reactivity; ② unseen lighting, *i.e.*, switching illumination to out-of-distribution conditions (*e.g.*, sudden dimming or color-temperature shifts) to require perceptual invariance; and ③ visual distractions, *i.e.*, introducing additional distractor objects to induce workspace clutter and occlusions. This analysis verifies that the progress estimate tracks the underlying task state rather than spurious visual correlations, supporting robust closed-loop behavior in unstructured environments. The details and results are as follows.

◇ **Random Relocation Disturbances.** As illustrated in Figure 5, during the subtask of "pick up grape", *we relocate the grape to a new pose twice mid-execution* (vertical dashed lines). The predicted progress increases smoothly over time and, at each relocation event, exhibits a transient deviation followed by a rapid recovery that quickly returns to the previous upward trend rather than resetting or collapsing. This behavior indicates that the progress signal is robust to substantial changes in the target object's pose and continues to track subtask completion rather than the instantaneous geometric configuration of the scene.

◇ **Unseen Lighting Disturbances.** As shown in Figure 6, during the subtask of "pick up grape", we introduce two abrupt switches to unseen illumination conditions (vertical dashed lines), substantially altering global brightness and shadows while leaving the physical scene unchanged. The predicted progress continues to increase over time, with only mild local deviations at the change points before returning to its prior upward trend. This indicates that the progress estimator is robust to severe photometric disturbances, exhibiting stable behavior under illumination changes.

◇ **Multi-Object Visual Distractions.** As illustrated in Figure 7, during the subtask of "pick up grape", we introduce *multiple additional objects into the scene twice* (vertical dashed lines), placing them near the target and within the camera's field of view to create visual clutter. The predicted progress keeps increasing over time, with localized deviations at the distraction events that promptly recover and return to the overall upward trend. This indicates that the progress estimator remains stable under multi-object visual distractions, with limited sensitivity to distractors.

C. More Experiments

C.1. Ablation on Progress Threshold

The threshold ϕ serves as the decision boundary for terminating the current sub-policy and triggering the subsequent

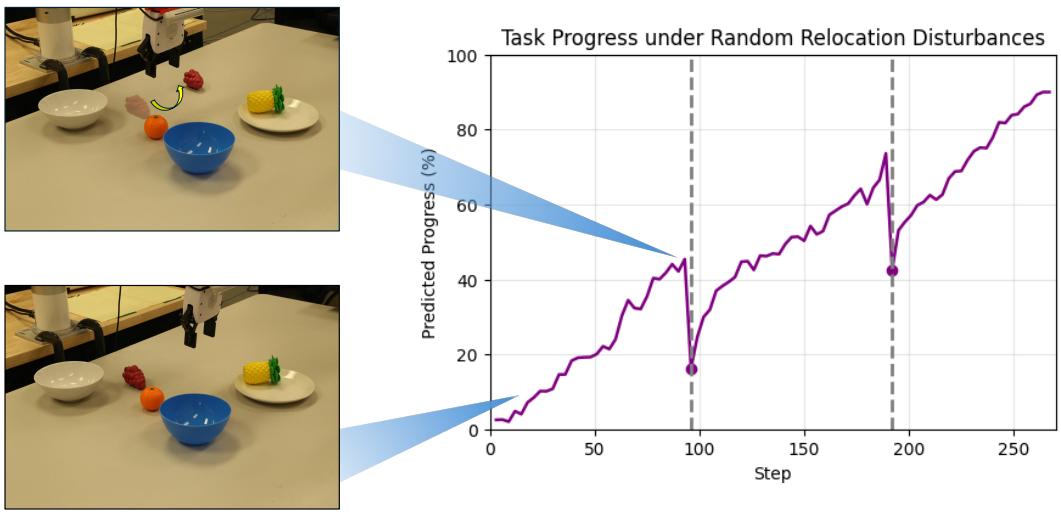

Figure 5. **Random Relocation Disturbances.** Predicted progress in the "`pick up grape`" subtask under two random grape relocations.

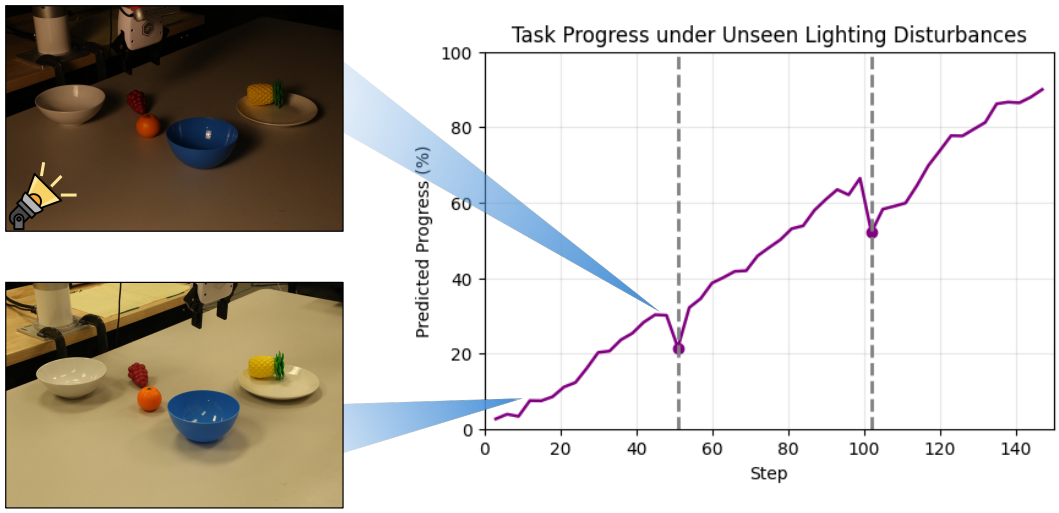

Figure 6. **Unseen Lighting Disturbances.** Predicted progress in the "`pick up grape`" subtask under two unseen lighting changes.

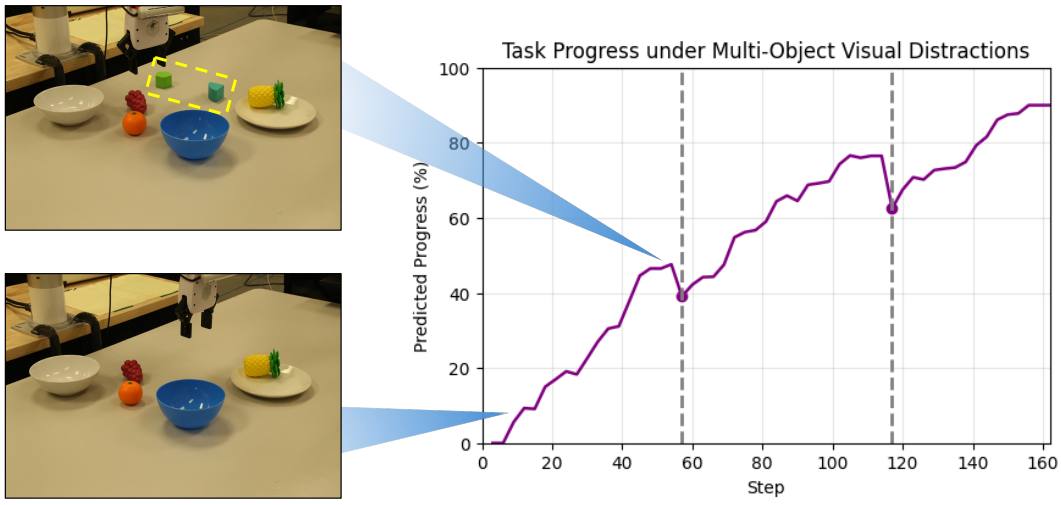

Figure 7. **Multi-Object Visual Distractions.** Predicted progress in "`pick up grape`" under two injected visual distraction events.

Table 9. **Ablation studies on the progress threshold ϕ.** Results on the CALVIN ABC→D benchmark demonstrate the effect of the threshold setting on long-horizon task performance.

Threshold ϕ	Task completed in a row					
	1	2	3	4	5	Avg. Len.
70%	93.4	86.3	78.4	71.9	65.5	3.96
80%	95.3	90.2	84.2	79.7	74.1	4.24
90%	**96.9**	**93.8**	**89.3**	**85.9**	**82.0**	**4.48**
100%	95.2	89.7	84.4	78.3	73.0	4.21

Table 10. **Ablation on prediction vs. reconstruction.** Results on CALVIN ABC→D and inference latency, demonstrating the impact of different prediction and reconstruction objectives on long-horizon task performance.

Ablation Type	CALVIN ABC→D	Latency
	Avg. Len. ↑	(ms) ↓
Affordance (PALM)	**4.48**	**~70**
Image / Video	4.17	~90
Auxiliary	3.58	~55

phase based on the predicted progress signal p_t. Since p_t serves as a continuous indicator of the sub-task's temporal progress, the setting of this threshold directly governs the transition timing: a lower threshold risks premature termination of the action, while an excessively high threshold may induce execution stagnation due to signal non-saturation. To investigate the impact of this parameter on final manipulation performance, we conduct an ablation study on the CALVIN ABC→D benchmark across $\phi \in \{70\%, 80\%, 90\%, 100\%\}$, with $\phi = 90\%$ as the default setting. This analysis aims to validate the effectiveness of the switching logic in long-horizon tasks by quantifying this trade-off.

As illustrated in Table 9, the default threshold of 90% yields the highest success rates across all chain lengths, achieving an average sequence length of 4.48. Reducing ϕ to 70% significantly degrades performance due to premature transitions, whereas increasing it to 100% causes a slight decline, confirming that a strict saturation requirement can hinder task completion.

C.2. Ablation on Prediction vs. Reconstruction

We perform an ablation on prediction targets vs. reconstruction to understand which form of foresight best supports long-horizon control. With the backbone and task-conditioned policy fixed, we compare: (1) Affordance Foresight, which predicts future affordance maps to highlight actionable regions and interaction points; (2) Image/Video Foresight, which predicts future RGB observations as a purely pixellevel forecasting objective; and (3) Auxiliary or Reconstruction, which reconstructs the current observation at time t as a representation-learning signal without future prediction. All variants are evaluated on CALVIN ABC→D using the average number of consecutively completed subtasks (Avg. Len.) and inference latency.

As shown in Table 10, affordance foresight attains the highest average length (4.48) with moderate latency (~70 ms), while future RGB prediction yields lower performance (4.17) at higher latency (~90 ms), and auxiliary reconstruction is fastest (~55 ms) but degrades performance the most (3.58). This pattern indicates that structured, action-centric affordance prediction yields substantial long-horizon gains without prohibitive computational cost, and provides the most effective trade-off among the tested prediction types.

Table 11. **Quantifying real-robot failure modes.** Compared with baseline, PALM achieves longer executions and substantially reduces repeat, skip, and premature-stop errors on the real robot.

Method	Avg. Len. ↑	Repeat ↓ Rate	Skip ↓ Rate	Premature Stop ↓
OpenVLA [55]	2.30	28%	16%	22%
Octo [111]	1.85	34%	26%	18%
PALM	**3.90**	**14%**	**8%**	**10%**

C.3. Analysis of Real-Robot Failure Modes

Table 11 analyzes real-robot failures in the same 6-subtask setting with N=50 rollouts per method. We report episode-level **Repeat** (small ineffective motions), **Skip** (attempting later subtasks early), and **Premature Stop** (sustained near-zero motion before completion). Rates are the fraction of episodes with ≥1 event under identical detection rules. PALM reduces such failures compared to baselines.

D. Qualitative Results and Visualization

In this section, we visualize the internal reasoning process of PALM during the execution of a long-horizon manipulation task: "Slide the pick block into the drawer." As illustrated in Figure 8, our model actively predicts structured affordances to guide its decision-making. We display the rollout over five timesteps, visualizing the four distinct affordance outputs that the model generates:

- **Global Affordance:** The model correctly segments the target object (the pink block) and the goal region (the open drawer), demonstrating semantic understanding of the instruction.
- **Local Affordance:** The heatmaps focus precisely on the interaction points, shifting from the block's graspable surface to the handle of the drawer as the task progresses.
- **Spatial Affordance:** The predicted yellow keypoints indicate valid placement candidates, guiding the robot to move the block towards the drawer's opening.
- **Dynamic Affordance:** The arrows visualize the predicted motion of the end-effector and the object, showing a clear trajectory for sliding the block forward.

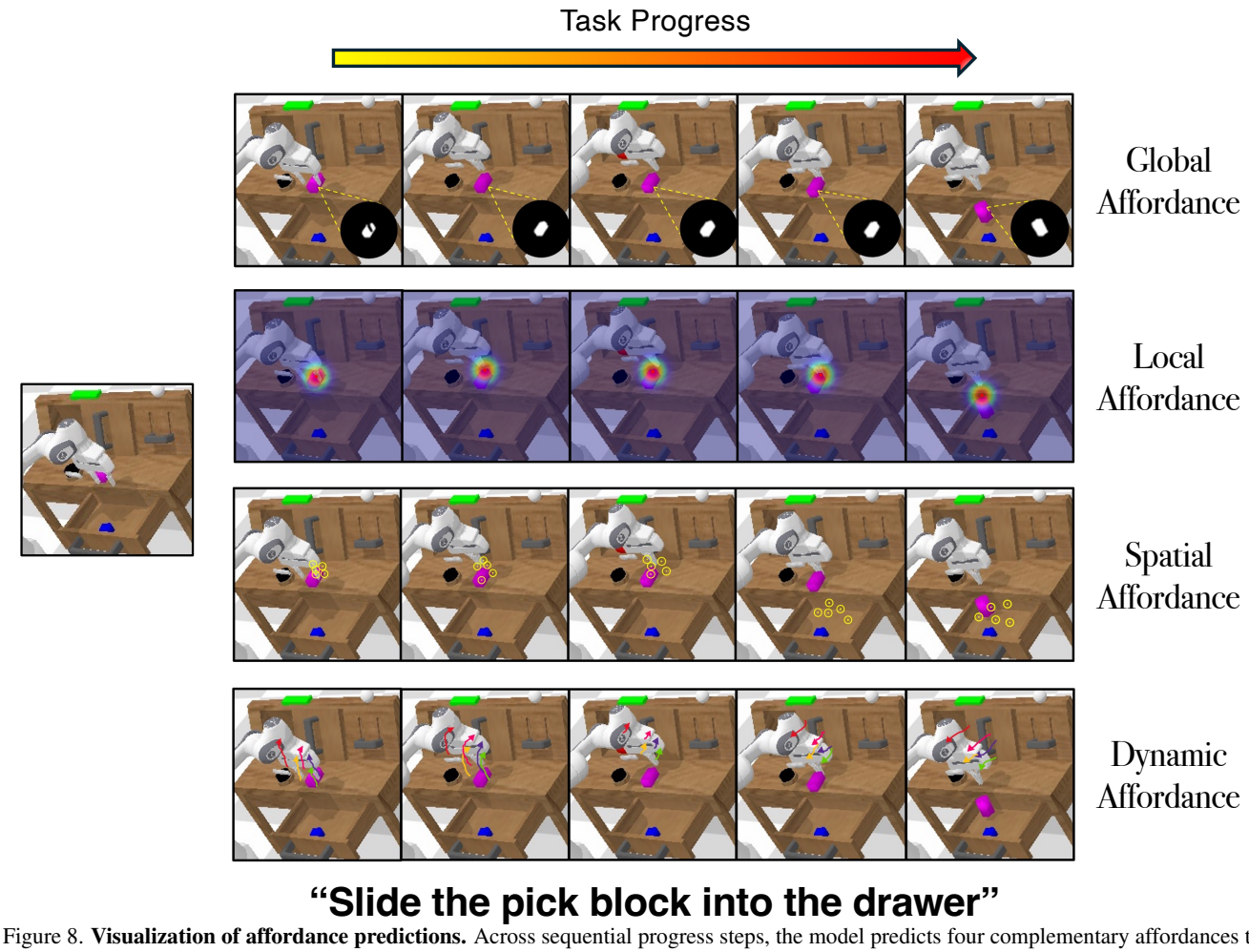

Figure 8. **Visualization of affordance predictions.** Across sequential progress steps, the model predicts four complementary affordances to guide policy generation: **Global Affordance** segments task-relevant objects and goals; **Local Affordance** generates heatmaps for precise contact points; **Spatial Affordance** predicts candidate placement regions; and **Dynamic Affordance** forecasts motion trajectories.

These visualizations confirm that **PALM** builds a comprehensive, structured representation of the task. By explicitly forecasting *what* to interact with (Global/Local), *where* to move (Spatial), and *how* the scene will evolve (Dynamic), the model achieves precise and robust control, successfully completing the task where baselines often fail due to ambiguity or lack of spatial reasoning.

E. Broader Impacts

PALM advances robotic manipulation by enhancing intermediate reasoning to mitigate common failures in long-horizon tasks. Rather than direct sensorimotor mapping, the model first anticipates a structured set of future affordances that encode task-relevant object identities, interaction points, spatial goal regions, and motion patterns. It then couples these predictions with a mechanism for estimating progress within the current subtask. This combination of structured affordance prediction and progress-aware state tracking yields an internal representation that helps policies remain coherent and effective across complex multi-stage activities where traditional models often fail. At the application level, progress-aware, affordance-based policies can help move long-horizon manipulation from controlled labs to open-world settings (*e.g.*, homes, warehouses, hospitals) when paired with sufficient data and stronger inductive priors. Because PALM exposes explicit affordance and progress signals, it integrates more naturally with safety monitors, constraint checks, and human-in-the-loop control, and it simplifies visualization and debugging relative to black-box policies.

However, these structured representations may reflect dataset biases (*e.g.*, which objects are prioritized or what counts as success), and premature deployment in safety-critical environments could amplify such risks. Realizing the benefits, therefore, requires standardized evaluation, robust sim-to-real transfer, and careful safety and societal-impact assessments so that capability gains do not come at the expense of safety, employment, or privacy.

F. Limitations and Future Work

While PALM has significantly improved long-horizon performance, its online recovery capability remains limited under execution drift caused by partial observability, contact uncertainty, occlusions, and geometric deformations. We plan to improve deviation detection and state-consistency checks to obtain more accurate state estimates and, after localizing drift sources at the subtask, object, and contact levels, trigger replanning or robust rollback based on the current state. At the same time, affordance-oriented segmentation, state grounding, and VLM-based semantic interpretation still introduce nontrivial overhead in annotation, perception, and inference [24, 59–62, 95, 116, 132, 145]. We further plan to combine this task-conditioned control framework with reinforcement learning (RL) and failure analysis from experience [32, 44, 82, 115, 122, 140], while extending it to 3D world perception [14, 43, 133, 141, 142, 147], to further improve execution stability, robustness, and scalability for broader industrial applications [56, 68, 127–129, 131, 134].