SaPaVe: Towards Active Perception and Manipulation in Vision-Language-Action Models for Robotics

Supplementary Material

In the supplementary document, we will explain in detail the parts that are not fully elaborated upon in the main text due to page limitations. These explanations are organized as follows:

- Sec. A: Discussions on certain details within the paper.
- Sec. B: Details of ActiveViewPose-200K, including data filtering, collection, and QA generation.
- Sec. C: Implementation details of ActiveManip-Bench, including environment setup, task design and data collection methods.
- Sec. D: More Details on real world experimental settings, including hardware setup, task protocols and generalization testing.
- Sec. E: Implementation details of SaPaVe, including architecture and training details of each stage.
- Sec. F: More Demonstrations of SaPaVe.
- Sec. G: Discussion on Limitations and Future Work.

A. Discussion

View Variation. We add tasks with larger camera movements and conduct real-world experiments following previous evaluation protocol. Fig. 1 shows that execution rollouts involve large viewpoint shifts ($> 90°$). Only our method succeeds with 45% success rate, validating the effectiveness of our active-view strategy and model design.

Semantic Active Perception Evaluation. The values in Fig.3 represent the final pitch and yaw changes of the camera movement. We detail the inference of different models and the evaluation process as follows: **(1)** Multi-SpatialMLLM [8] directly predicts the total pitch and yaw changes, leveraging its strong camera understanding. **(2)** Gemini-2.5-Pro [2] predicts the center point of the target view after camera movement, and then computes the corresponding pitch and yaw changes based on camera geometry, as it has strong pointing capability. **(3)** Our model directly outputs a camera movement chunk (A_{head}), and we obtain the final pitch and yaw changes by accumulating the deltas across chunks. For evaluation, a prediction is considered successful if it falls within a predefined tolerance of the ground-truth pitch and yaw changes; otherwise, it fails.

Additional Wrist Cameras. We attribute the performance drop in active perception settings with wrist cameras mainly to ***Noise Information*** and ***Data Mismatch***. **(1)** ***Wrist cameras are not inherently detrimental***: They are useful when the ego (head) view is fixed. In Tab. 2, Fixed + Wrist outperforms Fixed alone, especially for out-of-view settings, showing that wrist views offer valuable complementary information when head movement is restricted. **(2)** ***Noise Information***: The head camera alone usually provides sufficient visual input for task completion, and in practice, teleoperators mainly rely on it to collect data. While wrist cameras provide finer localization, especially for distant or small objects, they also introduce noise (*e.g.*, occlusions) in active manipulation, which negatively impacts performance. **(3)** ***Data Mismatch***: The noise information actually further stems from data mismatch, as ActiveViewPose-200K contains only head camera movements for Stages 1 and 2, while the paired head–wrist data is limited (20k samples) and used only in Stage 2. Although we try our best to decouple the head camera action space to specifically enhance active view capability, the scarcity of paired data limits effective learning of how to integrate the information from both head and wrist cameras. We will scale up paired head–wrist data to improve active manipulation.

B. ActiveViewPose-200K

To train the model's semantic active perception capability, we constructed a large-scale synthetic dataset—ActiveViewPose-200K. This section details the four-stage pipeline: 3D asset curation, procedural scene synthesis, task-driven view annotation, and instruction augmentation.

B.1. 3D Asset Curation and Filtering

We source 3D objects from the Objaverse dataset[3], utilizing LVIS annotations for semantic categorization. To ensure the objects are suitable for physical interaction simulation and geometric estimation, we applied a rigorous filtering protocol. The selection criteria are as follows:

Semantic Relevance: Objects are restricted to common indoor categories (*e.g.*, household items, tools, decorations).

Object Integrity: We strictly select single objects rather than object assemblies or pre-arranged scenes to ensure flexible placement.

Geometric Stability: Objects must possess a flat base to ensure stable placement on planar surfaces (*e.g.*, tables,

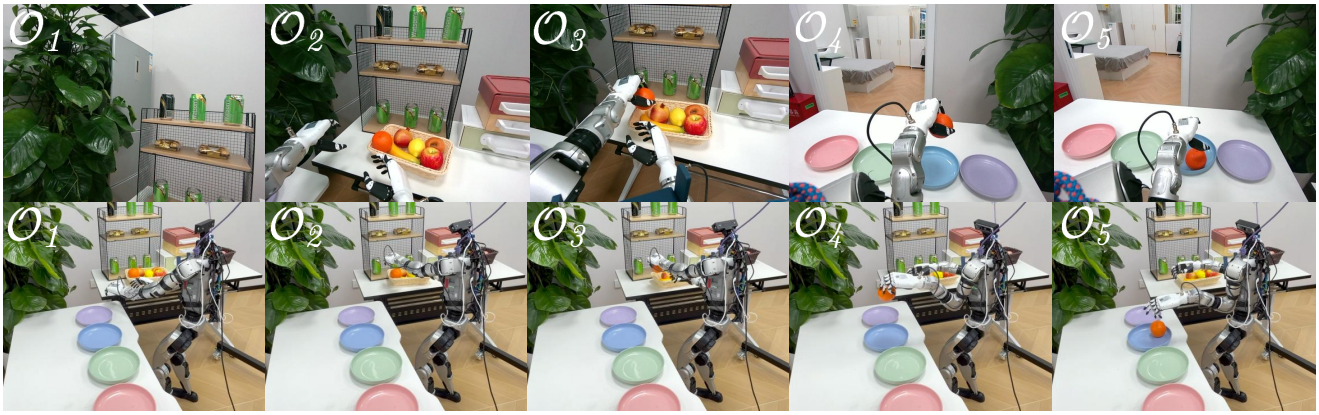

Figure 1. Real-world Execution roll-outs (ego & third view).

counters).

Visual Consistency: Assets must exhibit realistic textures and color styles compatible with synthetic indoor environments.

Topology Cleaning: We filter out objects with extraneous "loose" parts that affect bounding box calculation and collision size, such as wires attached to mice or cables connected to webcams.

B.2. Procedural Scene Generation

We utilize *Infinigen*[7], a procedural scene generation framework, to construct diverse indoor environments. Our curated assets are integrated into the Infinigen asset library, replacing or augmenting default placeholders. To simulate real-world distribution, we generate scenes across four functional room types with specific probability distributions based on common household activity frequency: Kitchen (32%), Living Room (18%), Dining Room (29%), and Bathroom (21%). Figure 3 demonstrates the diversity of the generated environments.

B.3. Task Formulation

The robotic tasks are defined by a combination of atomic actions and specific prompt modalities that guide the camera movement strategies.

Atomic Actions. Our dataset covers a range of manipulation-oriented atomic tasks: `pick`, `reorient`, `open`, `close`, `pour`, and `place`.

Prompt Modalities. We design three distinct types of textual prompts to train the model's capability to interpret different forms of spatial instructions:

- **Visual Centering:** The target is partially visible or off-center. The goal is to center the view for operation.
  Example: "Pick up/Reorient the [Object]. (the object is in the image)"
- **Spatial Directive:** The target is not currently visible. The instruction provides explicit relative direction.
  Example: "Turn left and pick up the [Object] on the table" or "Look up and retrieve the [Object] on the shelf."
- **Common Sense:** The target is not visible, and no explicit direction is given. The agent must infer the location based on semantic context.
  Example: "Pick up the [Object] in the base cabinet" (implying a downward look).

Data Augmentation. To enhance the model's capabilities, we introduce complex scenarios:

- **Conditional Reasoning:** Referring to objects by attributes such as position sequences, size, or color.
  Example: "Pick up the second plate from left to right" or "Place the biggest apple inside the right basket."
- **Container Interaction:** Tasks that require opening a container before grasping an item.
  Example: "Grasp the jar in the left drawer."

For the generated scenes, we randomly place objects within them and, according to different task templates, search for objects or object groups that satisfy the required positional relationships. We then sample a camera pose within a reasonable range and compute the optimal viewpoint for completing the task, which serves as the target view. This target view is subsequently perturbed within a prescribed range to produce the initial view. Representative samples are shown in Figure 4 and Figure 5.

B.4. Instruction Augmentation via LLM

To prevent the model from overfitting to rigid template patterns, we employ GPT-4 to rewrite the template-generated instructions. We prompt the LLM to introduce linguistic diversity while preserving the core semantic meaning and spatial relationships. The prompt design encourages the inclusion of diverse cues, such as explicit directional commands or implicit commonsense references (e.g., implying "cupboard" is "above"). The prompt structure used for instruction rewriting is shown in Listing 1.

Listing 1. Prompt used for diverse instruction generation.

Figure 2. Examples of the asset curation process.

```
You are a task-description rewriting expert for
    robot learning, whose goal is to enrich data
    diversity by converting structured template
    instructions into more natural and varied
    expressions.

Rewriting Rules:

Mandatory Principles:
1. Maintain imperative form: every rewritten
    instruction must be a direct imperative that
    states the action to be performed.
2. Preserve object names exactly: names must be
    kept without synonym substitution or
    generalization.
3. Keep the action equivalent: although wording
    may vary, the intended action must remain
    functionally the same.
4. Do not alter task difficulty: rewriting must
    not introduce new objects or change the
    actual complexity.

Acceptable Variations:
1. Verb diversity: choose more specific or
    natural verbs suited to the object and target
     location.
   - verbs should align with functional use of
      the object and destination.

2. Sentence-structure changes: reorganize the
    sentence.
   - Convert simple to complex by adding purpose
      or manner adverbials.

3. Contextual rationale: add reasons or purposes
    for the action.
   - Leverage common uses of the objects to
      supply functional descriptions.

4. Manner modifiers: include relevant adverbs.

5. Preparatory phrases: occasionally insert
    expressions implying prior steps.

Prohibited Changes:
1. Altering object names.
2. Changing target locations or target objects.
3. Replacing core referents with synonyms.
4. Introducing new objects or locations.
5. Switching to non-imperative forms (e.g., "You
    should...", "It is necessary to...").
6. Adding unrelated task steps.

Task:
Original instruction: {INSTRUCTION}

Provide only the rewritten instruction.
```

C. ActiveManip-Bench

In this section, we provide a detailed description of how ActiveManip-Bench is constructed, including the simulation environment details, simulation task design and efficient data collection methods.

C.1. Simulation Environment Details

Simulation Platform. We build ActiveManip-Bench environment on the NVIDIA's Isaac Sim platform[6], a ded-

Figure 3. Generated synthetic environments using Infinigen integrated with our curated asset library. The scenes feature varying lighting conditions, furniture layouts, and clutter distributions.

icated robotics simulation application that delivers photorealistic rendering and physically accurate simulations for robotics research and development.As illustrated in Figure 9, ActiveManip-Bench provides photorealistic rendering and physically accurate simulations.

Scene Configuration. In ActiveManip-Bench, we include 100 different objects and 20 distinct scenes. These simulation assets are high-quality digital-twin indoor-scene articulated objects with precise collision meshes and dynamic joint parameters, ensuring that simulated motion faithfully replicates real-world physics and kinematics. We collect our simulation assets from [5]. Similarly, we carefully curated a collection of high-quality, large-scale indoor environments from [4, 5]. Additionally, the asset format supports quick scene and object substitution through simple operations, allowing users to tailor environments to their individual requirements.

Detailed Robot Specifications In our simulation, we employ a G1 robot equipped with two Inspire third-generation dexterous hands, each featuring 6 degrees of freedom, while each arm provides 7 degrees of freedom. We have replaced the robot's original fixed head with a 2-DoF active head, and we use this newly designed system to execute tasks. Through our teleoperation algorithm or a deployed model, we can directly control the 7 joints of both arms, the 6 joints of each dexterous hand, and the 2 joints of the head.

Camera Viewpoints. Our simulator supports three types of camera viewpoints: (1) An active ego-camera viewpoint. (2) Wrist-camera viewpoints for the robot's left and right arms. (3) A fixed third-person camera viewpoint.Each viewpoint has a default RGB input resolution of 224×224, although users can flexibly adjust this resolution as needed.

C.2. Active Manipulation Task Suite.

We have designed 12 Active Manipulation Tasks with comprehensive semantic annotations. To rigorously evaluate robotic capabilities, we structure our task suite along two orthogonal dimensions: **(1) Task Objectives**, ranging from atomic skills to long-horizon reasoning, and **(2) Visual Complexity**, ranging from fully visible scenarios to those requiring active semantic exploration.

C.2.1. Hierarchical Task Objectives

We classify the 12 tasks into three categories based on their horizon length and semantic dependencies. Table 1 presents

Figure 4. Atomic Scene Tasks and Viewpoint Movement Examples.

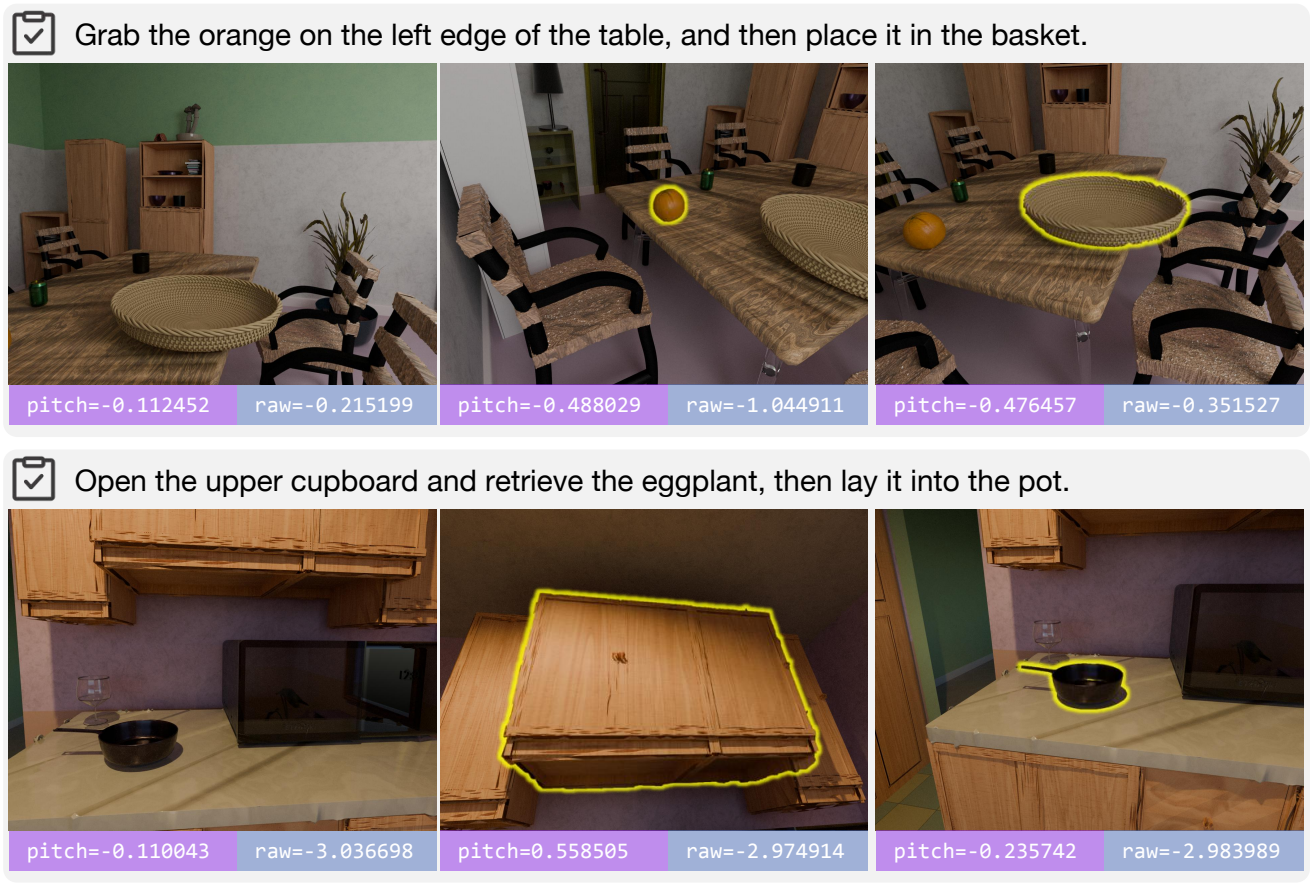

Figure 5. Two-stage scene task and viewpoint movement example.

the success criteria, while Fig. 8, 9 provides detailed visualizations.

- **Atomic Manipulation Tasks.** We include six fundamental tasks focusing on rigid body manipulation and articulated object interaction. In Pick and Reorient, the robot must grasp an object to achieve a target 6-DoF pose. Specifically, Pick focuses on lifting the object to a specified height, while Reorient requires rotating the object to a target orientation from an arbitrary initial state. For the articulated four tasks {Open, Close}{Drawer, Cabinet}, the goal value specifies the geometric state of the articulated joint. Specifically, success for Drawer tasks depends on the linear distance of prismatic joints, whereas Cabinet tasks involve the angular state of revolute joints. The initial state is randomized to ensure the agent learns active adaptation rather than memorizing trajectories.
- **Short-Horizon Composite Tasks.** Building upon atomic skills, we introduce three tasks that require sequential execution: PickAndPlace, OpenCloseDrawer, and OpenCloseCabinet. In PickAndPlace, the agent must establish a stable grasp, transport the object avoiding collisions, and release it at a precise target location. The tasks $\{OpenClose\}\{Drawer, Cabinet\}$ serve as a testbed for reversible manipulation, requiring the robot to drastically alter the environment state (opening) and then restore it (closing), necessitating robust motion planning and continuous state tracking.
- **Long-Horizon Sequential Tasks.** We introduce three high-complexity tasks that embody rich semantic content: FetchFromDrawer, FetchFromCabinet, and PourLiquid. The FetchFrom tasks mimic realistic household scenarios comprising a four-stage chain: *Open-Pick-Place-Close*. Unlike simple picking, the robot must first manipulate the container (drawer/cabinet) to reveal the workspace, then perform confined-space manipulation to grasp the target object inside the occluded area, place it at a designated goal, and finally close the container to complete the task cycle. This imposes strict preconditions where each action's success depends on the geometric state resulting from the previous step.
  Finally, distinct from rigid-body tasks, PourLiquid challenges the robot's ability to manipulate dynamic fluids. The robot must pick up a bottle containing liquid and tilt it at a precise angle to transfer the content into a target container, requiring reasoning about liquid dynamics and volume retention.

Table 1. Overview of the 12 Active Manipulation Tasks. The tasks are categorized by horizon length and complexity. Success is rigorously defined by geometric thresholds (position, rotation, joint state) or physical quantities (liquid volume), requiring the robot to maintain the goal state for a stabilization period (e.g., 2 seconds).

Category	Task Name	Success Criteria (Goal State)
Atomic	PICK	Object height form table $>$ 5cm; velocity ≈ 0.
	REORIENT	Orientation error relative to target $< 15°$.
	OPENDRAWER	Prismatic joint distance (opening) $> 90\%$ of analytical limit.
	CLOSEDRAWER	Prismatic joint distance (opening) $< 5\%$ (fully closed).
	OPENCABINET	Revolute joint angle (opening) $> 80°$.
	CLOSECABINET	Revolute joint angle (opening) $< 5°$.
Short-Horizon	PICKANDPLACE	Object position error $<$ 5cm at target; stable placement.
	OPENCLOSEDRAWER	*Seq:* Fully open ($> 90\%$) $\rightarrow$ Fully closed ($< 5\%$).
	OPENCLOSECABINET	*Seq:* Fully open ($> 80°$) $\rightarrow$ Fully closed ($< 5°$).
Long-Horizon	FETCHFROMDRAWER	*Seq:* Open Drawer $\rightarrow$ Pick Object $\rightarrow$ Place on Counter $\rightarrow$ Close Drawer. (All preconditions must be met).
	FETCHFROMCABINET	*Seq:* Open Cabinet $\rightarrow$ Pick Object $\rightarrow$ Place on Counter $\rightarrow$ Close Cabinet. (All preconditions must be met).
	POURLIQUID	Liquid in target container $> 80\%$ of source volume; Spilled $< 10\%$.

C.2.2. Visual Complexity and Initialization

To evaluate the robustness of agents against visual uncertainties and their ability to utilize language for active perception, we introduce diversity in the initial object placements independent of the task type. For each task, the target object's initial state is sampled from three levels of visual complexity:

- **Unoccluded (Fully Visible).** The target object is initialized at a random position within the robot's reachable workspace and remains fully contained within the camera's field of view (FoV). Although the object is clearly visible, the agent must handle significant variations in the object's 6-DoF pose and background clutter, requiring robust pose estimation and grounding capabilities.
- **Occluded (Partially Observable).** This category challenges the agent with incomplete visual information, presenting two distinct scenarios: (1) *Visual Truncation*, where the target is located at the periphery of the camera frame, leaving only a fraction of the object visible; and (2) *Physical Occlusion*, where the target is blocked by extraneous objects in the environment. Success in this mode requires strategic active behaviors rather than passive perception. Specifically, for truncated views, the agent must actively adjust its viewpoint to bring the complete object into the field of view. Conversely, when facing physical occlusion, the agent must possess the capability to interact with the environment—identifying and moving away the occluding obstacles to reveal the target object.
- **Out-of-View (Semantically Grounded Search).** This represents the highest difficulty level, where the target object is completely absent from the initial observation. In this setting, the agent must rely on the *semantic cues* embedded in the task instruction to locate the object. For instance, given the instruction *"pick the bottle from the bottom cabinet"*, the agent must infer that the target is likely situated in a lower spatial region. Consequently, it must perform active perception—moving its camera or base to explore the environment guided by linguistic priors—before it can commence the manipulation phase.

C.3. Data Collection and Generation Pipeline

To construct a large-scale, high-quality dataset for active manipulation, we implemented a three-stage pipeline encompassing human demonstration collection, automated simulation-based augmentation, and semantic instruction enrichment.

C.3.1. VR-based Human Teleoperation.

We developed a high-fidelity teleoperation system inspired by [1] to capture natural human behaviors. The system utilizes the *Apple Vision Pro* as the interface hardware. This setup allows for the holistic capture of the operator's intent: the headset tracks the 6-DoF head pose to render an egocentric active perception viewpoint, while the hand tracking capabilities capture precision dexterity. The detected human wrist poses and finger states are retargeted in real-time to the simulated robot's end-effector pose and gripper state via analytical Inverse Kinematics (IK), ensuring that the collected trajectories are kinematically feasible for the

robot.

C.3.2. Scalable Data Augmentation.

Relying solely on human teleoperation is labor-intensive. To scale up our dataset diversity, we employ two strategies:

- **Visual and Asset Randomization:** Our simulation framework supports hot-swapping of object assets and background environments. We systematically randomize textures, lighting conditions, and distractors to enhance the visual robustness of the trained policies.
- **Self-Improving Trajectory Generation:** To scale the quantity of trajectory data while maintaining human-like behavioral patterns, we adopt the methodology proposed in DexFlyWheel [9]. Instead of simple replay, we utilize a hybrid framework combining Imitation Learning (IL) with Residual Reinforcement Learning (RL). Specifically, we first train a base policy using human demos. We then employ a self-improving flywheel mechanism where Residual RL explores local adjustments to correct failures in the original demos. This process efficiently generates diverse, high-quality successful trajectories that preserve the nuanced intent of the original human demonstrations, significantly alleviating the scarcity of dexterous manipulation data.

C.3.3. Semi-Automatic Semantic Enrichment.

To bridge the gap between rigid template commands and natural language variation, we propose a semi-automatic semantic augmentation pipeline. We first define structured logical templates for each task, such as:

```
pick the [object] at [position] and
    place to [target_position]
```

These templates are then fed into a Large Language Model (LLM) to generate linguistically diverse and spatially specific instructions. For example, the LLM might expand the template into:

> *"Pick the blue pen located to the left of the computer and place it onto the notebook."*

Finally, to ensure grounding accuracy, all generated instructions undergo a round of human verification to correct for potential hallucinations or spatial mismatches.

D. Real-World Experiments

In order to better evaluate the differences between our model and the baseline, and to further assess the model's generalization and sim-to-real transferability, we conducted extensive experiments on a physical robot platform. The experimental design is structured into three parts: hardware setup, task protocols, and generalization testing.

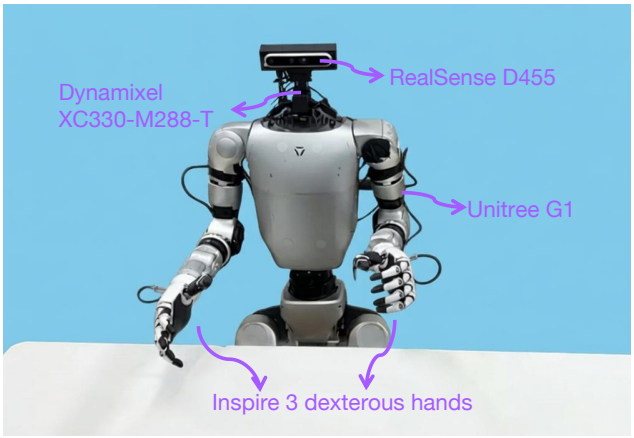

Figure 6. We provide the type of the robot, dexterous hand, camera, and servos used in our real-robot experiments.

D.1. Hardware Setup

As illustrated in Figure 6, our experimental platform is built upon the Unitree G1 humanoid robot. To enable high-fidelity dexterity, the robot is equipped with *Inspire 3* dexterous hands. Crucially, to support active perception research, we developed a custom active head system. The head structure is fabricated using 3D-printed components and actuated by high-precision servos (Dynamixel XC330-M288-T), allowing for independent pitch and yaw movements. A *RealSense D455* RGB-D camera is mounted on this head unit, serving as the primary ego-centric sensor. This setup allows the agent to actively adjust its camera pose to explore the environment, mirroring the setup in our simulation.

D.2. Real-World Task Protocols

We designed four distinct categories of tasks to evaluate the agent's manipulation and active perception capabilities under varying visibility constraints. The assets used in these experiments are shown in Fig. 7.

1. **Occluded Pick-and-Place.** *Definition:* The target object or the placement goal is partially outside the initial camera field of view (FoV), or both are partially truncated. Success requires the agent to infer the full spatial information from partial observations and actively adjust its viewpoint to ensure precise manipulation. *Instance:* The agent is tasked with transferring food items (e.g., fruits or bread) from a basket on the left side to a plate on the right side, where the containers are only partially visible initially.
2. **Out-of-View Pick-and-Place.** *Definition:* The target object or goal location is completely absent from the initial FoV. The agent must interpret semantic instructions (e.g., "on the left", "upper shelf") to actively rotate its head and locate the targets before manipulation. *Instance:* The task involves retrieving a cup from a high

shelf and placing it into a basket on the table's left side. The agent must utilize the semantic cue "upper shelf" to look up and locate the cup.

3. **Occluded Articulated Manipulation.** *Definition:* A long-horizon task where an articulated container (drawer) is partially visible. The agent must actively perceive to open the drawer, retrieve an object, close the drawer, and then actively search for the placement goal based on semantic cues. *Instance:* The agent must identify a partially visible drawer on the left side of the desk, open it to retrieve an object, and subsequently place it into a basket on the right side.
4. **Out-of-View Articulated Manipulation.** *Definition:* This is the most challenging setting where the articulated container is completely out of sight initially. The agent relies entirely on contextual priors and language instruction (e.g., "cabinet under the counter") to guide its active exploration (e.g., looking down) to locate the handle, perform the manipulation chain (Open-Pick-Close), and locate the final placement goal. *Instance:* We task the robot with retrieving a dish from a cabinet located underneath the countertop and placing it onto the tabletop. The agent must infer "under the counter" implies a downward gaze shift.

D.3. Generalization Evaluation

To rigorously assess the robustness of our policy, we evaluate calibration in Out-of-Distribution (OOD) settings across three dimensions, as shown in Fig. 7:

- **Unseen Objects:** We select two novel objects and drawers that were never present during training to test geometric generalization.
- **Unseen Scenes:** The robot is deployed in a completely novel table setup and background arrangement distinct from the training environment.
- **Lighting Variation:** We introduce lighting conditions unseen in the dataset (e.g., dim evening light or strong directional sunlight) to evaluate visual robustness.

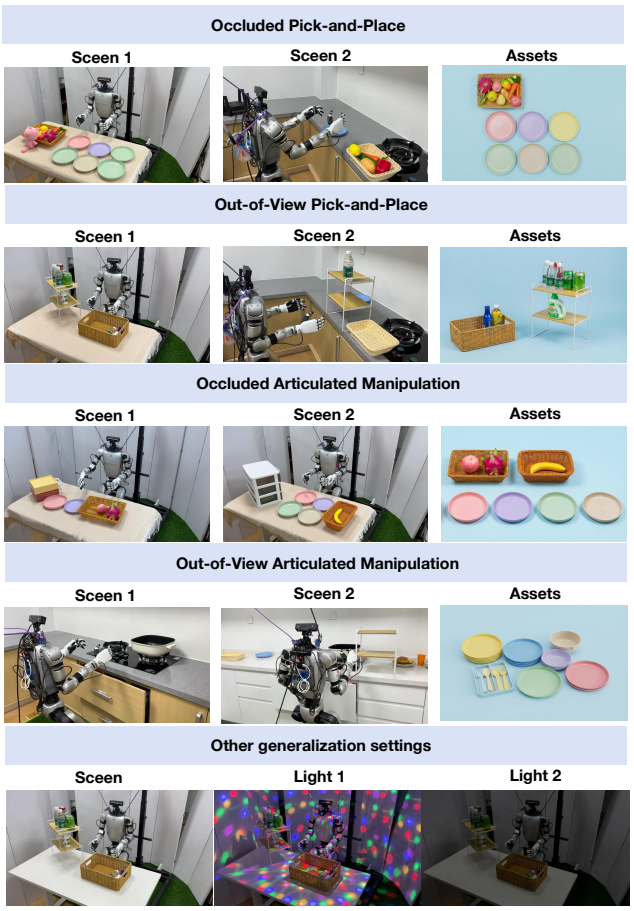

Figure 7. This diagram illustrates the four tasks in our real-world experiment categories, along with the employed scenarios, assets, and varying lighting conditions.

E. SaPaVe Architecture and Implementation

SaPaVe is constructed upon a modular foundation comprising a decoupled diffusion policy, a parameter-efficient vision-language adapter, and a universal spatial encoder. In this section, we detail the architectural innovations, training protocols, and evaluation metrics.

E.0.1. Decoupled Action Heads via Diffusion Transformer

For precise manipulation and active perception, we model the joint policy π using a variant of the Diffusion Transformer (DiT) analogous to System 1 in GR00T N1. Given the current observation state q_t and a target action chunk $A_t \in \mathbb{R}^{T\times D}$, the model learns to predict the noise ϵ added to the action latent. However, unlike monolithic outputs, we propose a **Decoupled Action Head** architecture to handle the distinct dynamics of the active head and the robotic body. The DiT backbone consists of alternating self-attention (handling action temporal dependencies) and cross-attention blocks (conditioning on multi-modal embeddings). At the final decoding stage, instead of a single projection, the latent features split into two specialized Multilayer Perceptron (MLP) decoders:

1. **Camera Action Decoder:** Predicts the 2-DoF kinematics for the active head (pitch and yaw), denoted as $\hat{A}_t^{cam} \in \mathbb{R}^{T\times 2}$.
2. **Manipulation Action Decoder:** Predicts the 26-DoF joint positions for the dual-arm dexterity, denoted as $\hat{A}_t^{body} \in \mathbb{R}^{T\times 26}$.

Formally, given a diffusion timestep τ and noisy action chunk A_t^τ, the network V_θ approximates the noise ϵ by min-

imizing the decoupled denoising loss:

$$\mathcal{L}_{diff} = \mathbb{E}_{\tau,\epsilon}\big[\lambda_1 \left\| V_\theta^{cam}(\dots) - \epsilon^{cam} \right\|^2 + \lambda_2 \left\| V_\theta^{body}(\dots) - \epsilon^{body} \right\|^2\big] \quad (1)$$

where $V_\theta(\phi, A_t^\tau, q_t)$ conditions on the fused embedding ϕ. This bifurcation enables the agent to learn high-frequency visual servoing (head) independently from complex manipulation planning (body).

E.0.2. Camera Adapter with Eagle-2 and LoRA

To imbue the agent with high-level semantic understanding for active search, we utilize **Eagle-2** as our Vision-Language Model (VLM) backbone. Eagle-2 integrates a SigLIP-2 image encoder with a SmolLM2 LLM, providing strong priors for grounding text instructions to visual observations. We sought to preserve the original VLM's high-level semantic information while efficiently learning an alignment between high-level semantics and low-level camera movements.Since full fine-tuning of the billion-parameter VLM is computationally prohibitive and risks catastrophic forgetting, we implement a **Camera Adapter** using Low-Rank Adaptation (LoRA). Referencing the parameter-efficient fine-tuning paradigm, we inject trainable rank decomposition matrices into the linear layers of the LLM's attention blocks while freezing the pre-trained weights W_0. For a linear projection $h = W_0 x$, the adapted forward pass is given by:

$$h = W_0 x + \frac{\alpha}{r} BAx \quad (2)$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ are trainable matrices with rank $r \ll d$. This adapter allows SaPaVe to align the generic semantic features of Eagle-2 with the specific requirements of active robotic perception using less than 2% of the trainable parameters, effectively bridging the gap between "internet-scale" knowledge and "robot-scale" control.

E.0.3. Universal Spatial Knowledge Injection

A core contribution of SaPaVe is the integration of explicit 3D geometric knowledge. We leverage the encoder from **MapAnything**, a foundation model capable of processing arbitrary combinations of visual and geometric modalities.

Multi-Modal Encoding Process. As shown in the reference architecture, the MapAnything encoder processes N input views. It handles two streams of data:

- **Visual Stream:** RGB images are encoded using a frozen *DINOv2* (ViT-L) backbone to extract high-level patch features F_{rgb}.
- **Geometric Stream:** Optional geometric priors (Camera Poses $\hat{P}$, Ray Directions $\hat{R}$, and Depths $\hat{D}$) are factorized and projected. Specifically, ray directions and normalized depths are processed via a shallow Convolutional Encoder with pixel unshuffle to match the specialized dimension of DINOv2 features. Global quantities (rotations, scales) are projected via a 4-layer MLP.

These features are fused via a sum-based aggregation:

$$F_{spatial} = \text{LayerNorm}(F_{rgb} + \text{Proj}(F_{geo})) \quad (3)$$

This results in a dense spatial token representation $F_{spatial}$ that inherently encodes metric 3D structure (positions and occupancy) aligned with the visual patches.

Injection via Element-wise Addition and Cross-Attention. To inject this spatial awareness into the action policy, we employ a late-fusion strategy within the DiT blocks. Let ϕ_{vlm} be the semantic tokens output by the Camera Adapter (Eagle-2 + LoRA). We first align the dimension of the spatial tokens $F_{spatial}$ to match ϕ_{vlm} via a linear projection. The multi-modal conditioning context ϕ_{fused} is obtained by **Element-wise Addition**:

$$\phi_{fused} = \phi_{vlm} + \beta \cdot \text{Linear}(F_{spatial}) \quad (4)$$

where β is a learnable scaling factor. Finally, this fused representation ϕ_{fused} serves as the *Key* and *Value* in the **Cross-Attention** layers of the Decoupled Action Heads (DiT), while the noised action tokens serve as the *Query*. This mechanism ensures that every denoising step is guided by both the semantic intent (from VLM) and the precise 3D geometry (from MapAnything Encoders).

E.1. Detailed Training Protocols

To effectively learn the complex coordination between active perception (head movements) and dexterous manipulation (dual-arm control), we employ a curriculum-style **Two-Stage Training Strategy**. This approach prevents the policy from falling into local minima where the robot attempts to manipulate without first establishing a valid visual observation.

E.1.1. Stage 1: Semantic Active Perception Alignment

Objective. In the initial stage, our primary goal is to bridge the gap between the VLM's high-level semantic understanding and the low-level kinematic control of the active head. We aim to equip the model with a strong *prior* for "where to look" based on language instructions, independent of the arm manipulation complexity.

Data Configuration. We use proposed ActiveViewPose-200K dataset. Unlike standard manipulation datasets collected from fixed, near-optimal viewpoints, this dataset comprises 200k tuples of $(Image, Instruction, \text{CameraAction})$. It specifically targets the transition from suboptimal or out-of-view initial states to optimal task-oriented viewpoints.

Optimization Strategy. During this stage, we freeze the *Universal Spatial Encoder* (MapAnything) and the main *Manipulation Action Decoder*. We strictly enforce updates on:

1. The **Camera Adapter** (LoRA layers on Eagle-2), allowing the VLM to adapt to active viewing instructions.
2. The **Camera Action Decoder** (the 2-DoF MLP head), learning to translate semantic features into pitch and yaw velocities.

Loss Function. The training objective minimizes the Mean Squared Error (MSE) between the predicted camera movement $\hat{A}_{\text{head}}$ and the ground-truth trajectory A^*_{head}:

$$\mathcal{L}_{\text{Stage1}} = \mathcal{L}_{\text{MSE}}(\hat{A}_{\text{head}}, A^*_{\text{head}}) = \frac{1}{T}\sum_{t=1}^{T} ||\hat{A}^t_{\text{head}} - A^{*t}_{\text{head}}||^2 \tag{5}$$

By the end of Stage 1, the model develops a robust capability to perform semantic active search, effectively becoming a "embodied cameraman" that can locate target objects specified by natural language.

E.1.2. Stage 2: Active Manipulation Fine-tuning

Objective. Building upon the semantic active perception prior established in Stage 1, the second stage introduces the full complexity of dexterous manipulation. This stage aims to achieve *Active-View Execution*, where the robot coordinates its gaze to support hand-object interaction.

Data Configuration. To prevent catastrophic forgetting of the active perception skills while learning manipulation, we employ a **Data Mixture Strategy**. The training batch consists of samples from both the *ActiveViewPose-200K* dataset (to maintain look-at capability) and the *Active Manipulation Robot Data* (generated via DexFlyWheel, covering the 12 core tasks),as well as the real world collected data.

Optimization Strategy. In this stage, we unlock the **Decoupled Action Heads**. We jointly train the entire action generation pipeline, including both the Camera Action Decoder and the Manipulation (Body) Action Decoder.

Loss Function. The objective is a weighted sum of the head movement loss and the body manipulation loss. We define the total loss as:

$$\mathcal{L}_{\text{Stage2}} = \lambda_{\text{head}}\mathcal{L}_{\text{head}} + \lambda_{\text{other}}\mathcal{L}_{\text{other}} \tag{6}$$

where:

- $\mathcal{L}_{\text{head}} = ||\hat{A}_{\text{head}} - A^*_{\text{head}}||^2$ ensures the camera continues to track relevant objects.
- $\mathcal{L}_{\text{other}} = ||\hat{A}_{\text{body}} - A^*_{\text{body}}||^2$ supervises the 26-DoF arm and hand joints.
- λ_{head} and λ_{other} are balancing coefficients. In practice, we set $\lambda_{\text{head}} = 1.0$ and $\lambda_{\text{other}} = 10.0$ to prioritize the higher-dimensional manipulation space while maintaining visual stability.

This two-stage approach ensures a "bottom-up" acquisition of skills: first learning to perceive, and then learning to act upon that perception.

F. Additional Demonstrations

In this section, we provide a comprehensive qualitative analysis of SaPaVe through extensive visualizations across both simulation and real-world environments.

F.1. Simulation Demonstrations

We present time-lapse sequences of the 12 active manipulation tasks in Fig. 8 9. These visualizations highlight the distinct behavior of our model compared to passive baselines:

- **Active View Finding:** In OUT-OF-VIEW initialization settings, the agent is shown rapidly rotating its camera to scan the environment based on semantic instructions (e.g., looking down for "bottom cabinet") before minimizing arm movement.
- **Handling Occlusion:** In the FETCHFROM tasks, the visualizations depict the agent actively maneuvering its head to peer inside opened drawers or cabinets to locate objects effectively before attempting a grasp.

Please refer to the supplementary video/website for full-length trajectory playbacks, which demonstrate the temporal smoothness and stability of the decoupled head-body coordination.

F.2. Real-World Robot Deployment

We further illustrate the robustness of SaPaVe on the Unitree G1 humanoid platform. Figure 10 11showcases execution traces for the OUT-OF-VIEW PICK-AND-PLACE and OCCLUDED ARTICULATED MANIPULATION tasks. Crucially, we demonstrate the model's ability to recover from disturbances. In scenarios where the target object is manually moved out of the current frame during execution, SaPaVe successfully re-initiates an active search to reacquire the target, validating the closed-loop nature of our active perception pipeline.

G. Limitations and Future Work

While SaPaVe demonstrates promising capabilities in active semantic manipulation, we acknowledge certain limitations in our current setup that point towards exciting directions for future research.

Workspace Constraints under Static Base. The primary limitation of the current implementation is that the robot operates with a fixed base. Although the active head significantly expands the effective *perceptual* field of view, the *manipulation* workspace remains bounded by the stationary

reach of the robot's arms. Consequently, if an object is located or moved beyond the maximum arm span (even if actively perceived and located by the head), the agent cannot complete the task. The current active manipulation is thus confined to "local" exploration rather than "global" search.

Towards Mobile Active Manipulation. To address the aforementioned constraint, our future work aims to extend SaPaVe to Mobile Manipulation settings. We envision a holistic framework where active perception controls not only the head (pitch/yaw) but also the mobile base (navigation). This would enable the robot to:

1. **Navigate to Look:** Move to different rooms or distinct areas of a room to locate objects described by high-level instructions (e.g., "Find the apple in the kitchen").
2. **Approach to Manipulate:** Actively position its base to maximize manipulability scores for grasping objects in confined spaces (e.g., inside deep cabinets).

Integrating whole-body control with VLM-guided active semantic search will be a critical step towards truly general-purpose embodied agents.

References

[1] Xuxin Cheng, Jialong Li, Shiqi Yang, Ge Yang, and Xiaolong Wang. Open-television: Teleoperation with immersive active visual feedback. *arXiv preprint arXiv:2407.01512*, 2024. 7

[2] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025. 1

[3] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13142–13153, 2023. 1

[4] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10933–10942, 2021. 4

[5] Zhao Jin, Zhengping Che, Zhen Zhao, Kun Wu, Yuheng Zhang, Yinuo Zhao, Zehui Liu, Qiang Zhang, Xiaozhu Ju, Jing Tian, et al. Artvip: Articulated digital assets of visual realism, modular interaction, and physical fidelity for robot learning. *arXiv preprint arXiv:2506.04941*, 2025. 4

[6] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021. 3

[7] Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, et al. Infinigen indoors: Photorealistic indoor scenes using procedural generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21783–21794, 2024. 2

[8] Runsen Xu, Weiyao Wang, Hao Tang, Xingyu Chen, Xiaodong Wang, Fu-Jen Chu, Dahua Lin, Matt Feiszli, and Kevin J Liang. Multi-spatialmllm: Multi-frame spatial understanding with multi-modal large language models. *arXiv preprint arXiv:2505.17015*, 2025. 1

[9] Kefei Zhu, Fengshuo Bai, YuanHao Xiang, Yishuai Cai, Xinglin Chen, Ruochong Li, Xingtao Wang, Hao Dong, Yaodong Yang, Xiaopeng Fan, et al. Dexflywheel: A scalable and self-improving data generation framework for dexterous manipulation. *arXiv preprint arXiv:2509.23829*, 2025. 8

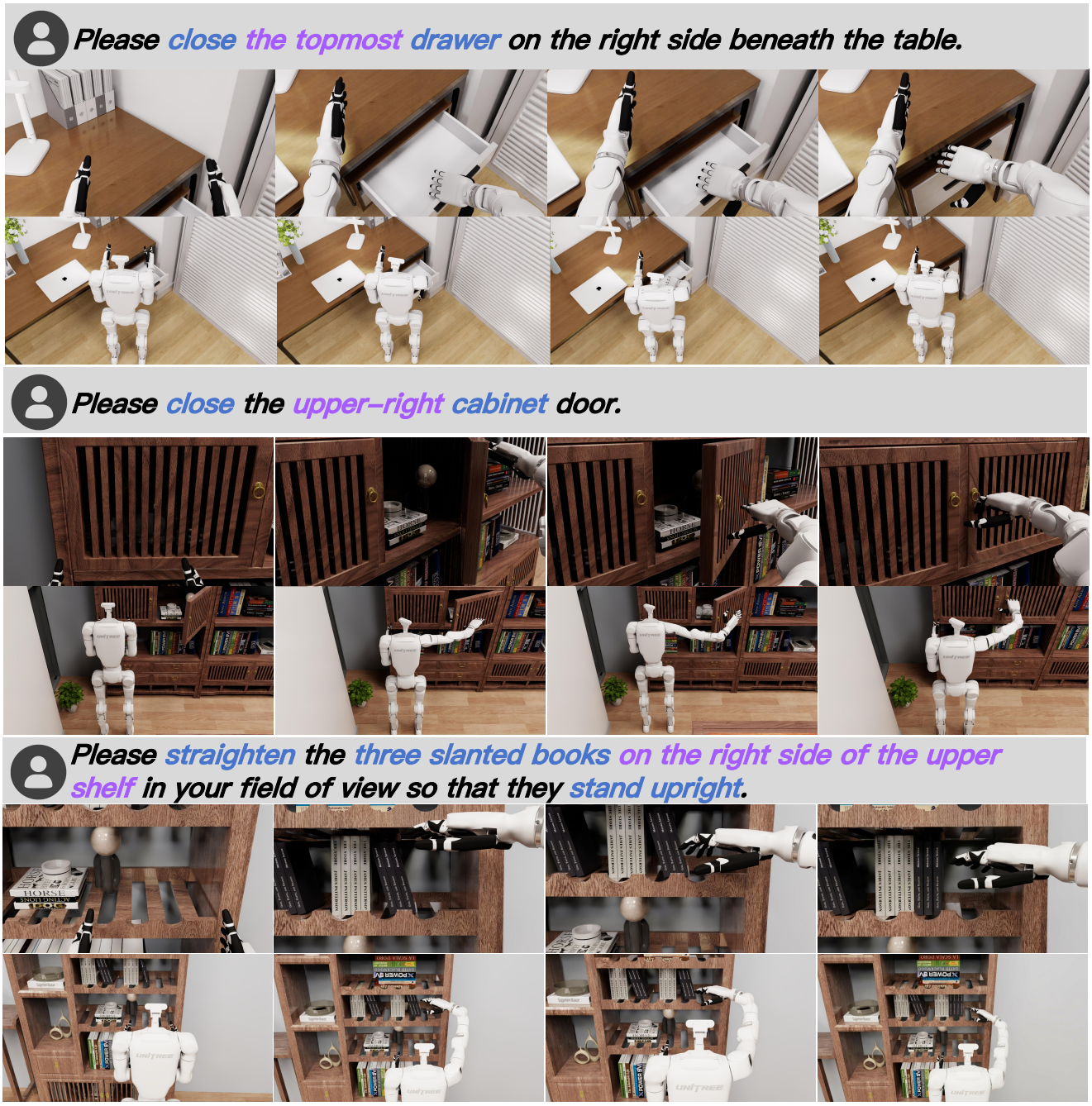

Figure 8. This image illustrates the active manipulate task performed by the robot in a simulation scenario.

Figure 9. This image illustrates the active manipulate task performed by the robot in a simulation scenario.

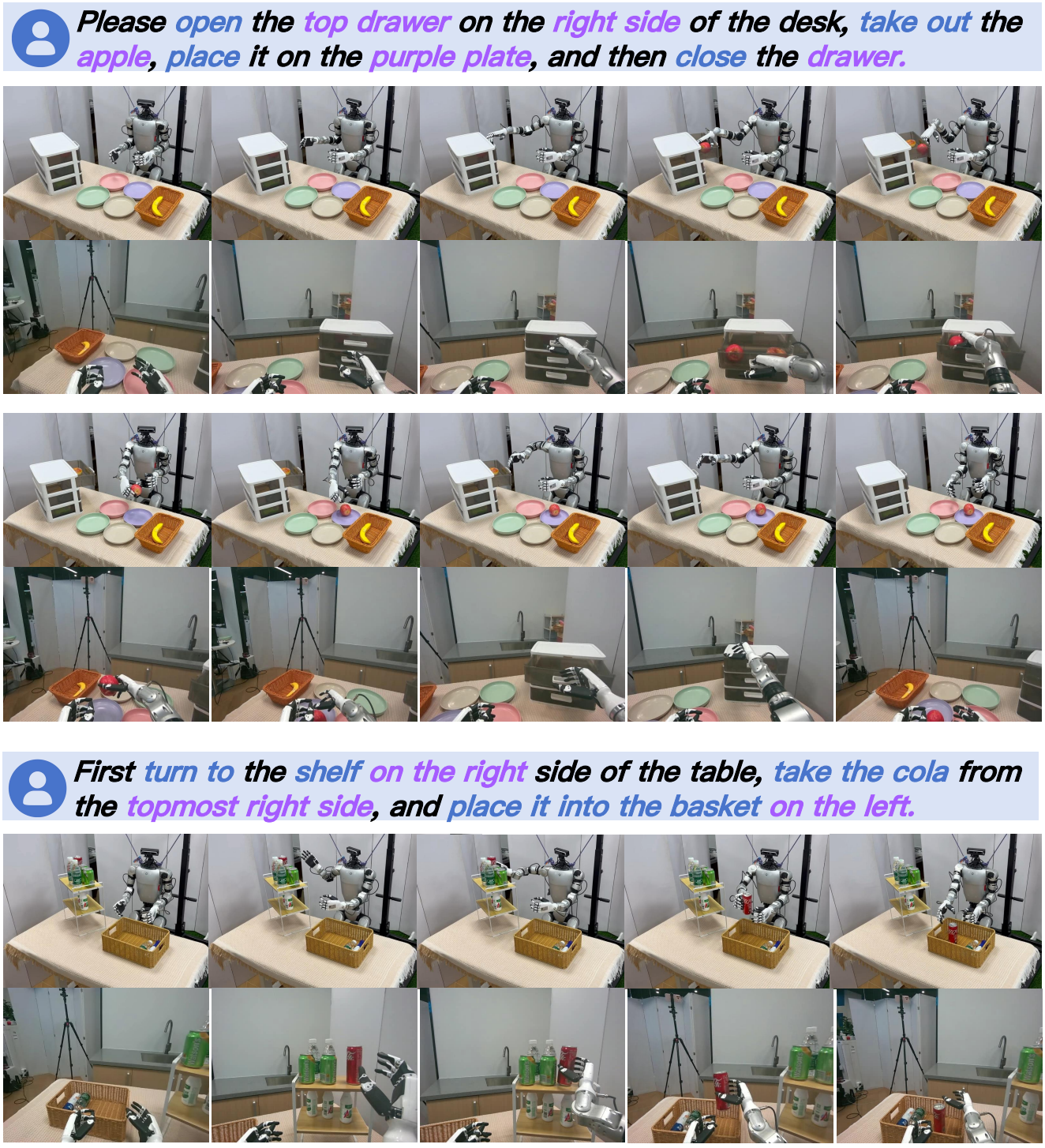

Figure 10. This image illustrates the active manipulate task performed by the robot in a real-world scenario.

Figure 11. This image illustrates the active manipulate task performed by the robot in a real-world scenario.