Zero-Shot Long-Horizon Dexterous Manipulation via Multi-View 3D-Grounded VLM Reasoning

Jisoo Kim[1], **Sangwon Baik**[1], **Taeksoo Kim**[1], **Sungjoo Kim**[1]
Junyoung Lee[1], **Mingi Choi**[1], **Hanbyul Joo**[1,2]

[1] Seoul National University [2] RLWRLD

{jlogkim,bsw1907,taeksu98,masterninja,juncong,willi19,hbjoo}@snu.ac.kr
https://jlogkim.github.io/zerodex3d

Abstract: We present a zero-shot framework for long-horizon dexterous manipulation that grounds language instructions into executable 3D task plans from calibrated multi-view RGB images. Rather than training an end-to-end policy, our system uses a vision-language model (VLM) to produce reference-frame task grounding and primitive-level 2D keypoints, then lifts them into 3D via multi-view fusion. This lifting combines triangulation of view-wise VLM groundings with reference-view ray voting, which searches along a semantic camera ray for geometrically consistent candidates across neighboring views. The resulting 3D keypoints support both pick-and-place and tool-use: for tool-use, we retrieve an object-centric atomic action corresponding to the inferred skill category and align its stored 6D tool trajectory to the scene; for dexterous execution, we expand the lifted grasp keypoint into a task-conditioned grasp affordance region and generate feasible grasp-motion pairs with an arm-hand motion generator. Real-world experiments show improved 3D grounding accuracy and execution reliability over single-view RGB-D grounding and fine-tuned VLA baselines. We further demonstrate long-horizon manipulation through closed-loop status verification and replan, enabling zero-shot execution on unseen objects and tool-use tasks in novel scenes.

Keywords: Zero-Shot Manipulation, Dexterous Grasping, Multi-view Grounding

1 Introduction

A long-standing goal in robotics is to build general-purpose systems that perform long-horizon manipulation from high-level language instructions. Beyond recognizing objects, such systems must ground instructions in task-relevant 3D geometry: where to place an object, which part to contact, and how to orient and move a tool during execution. This requirement is especially stringent for dexterous hands, where small 3D grounding errors cause unstable grasps, collisions, inverse-kinematics failures, or contact on the wrong functional region of a tool.

The dominant approach learns this behavior end-to-end. Recent vision-language-action (VLA) models make strong progress toward general-purpose manipulation by predicting actions directly from large-scale robot data [1, 2, 3, 4, 5]. However, achieving reliable performance on diverse manipulation tasks requires extensive data collection, task-specific adaptation, or environment-specific fine-tuning, which makes them hard to scale across the diversity of objects, tools, and spatial configurations found in open-ended settings. Human demonstration retargeting offers another route, but the embodiment gap between human and robotic hands produces physically infeasible contacts or unstable grasps, and often demands additional refinement or reinforcement learning before deployment.

We argue that a modular design is a more efficient route. Instead of learning a single end-to-end policy, we decouple semantic reasoning, handled by a vision-language model (VLM), from physical execution, handled by motion primitives or controllers. The key observation is that modern VLMs already answer, zero-shot, most of the sub-questions that manipulation requires: what to grasp, where to grasp it as a functional affordance, how to move it, and in what order to carry out the steps of a task. Given this competence, we do not need to train a new policy. Once the VLM produces a plan, we decompose a complex task into a sequence of simple atomic tasks such as pick and move, and we execute each atomic motion with a reliable primitive controller. Because most manipulation tasks can be expressed as compositions of such atomic tasks, a small library of primitives driven by VLM reasoning covers a broad range of tasks. We find that VLMs produce accurate plans and decompositions across a wide range of tasks, as demonstrated in our evaluations.

This modular design, however, hinges on a geometric requirement that 2D reasoning cannot satisfy on its own, as the relevant quantities are inherently three-dimensional. Where to grasp must be specified in 3D rather than 2D, and more importantly, how to move the end effector and where to move an object are 3D trajectory quantities. A single view rarely carries enough geometric information to reason about these 3D trajectories reliably, so attaching a VLM to a single image is limited for this purpose. Our central idea is to fuse VLM grounding across multiple views. Multi-view fusion lifts view-dependent 2D predictions into consistent 3D quantities, and it extends VLM-based grounding to a far wider set of problems than single-view reasoning can reach, including 3D contact points, placement targets, and tool-motion anchors. Within this framework we triangulate VLM groundings across views to recover 3D positions and directions. VLM 2D groundings are also semantic and view-dependent, since different views highlight different visible parts of an object, occlusion shifts the predicted location, and ambiguous task context yields predictions that vary across cameras. To handle this, and to anchor estimation more firmly to a chosen reference view, we further introduce a VLM voting scheme that searches along the reference camera ray for the 3D candidate most consistent across the remaining views [6]. Triangulation and reference-view voting are complementary parts of the same multi-view fusion approach, and together they produce reliable 3D grounding under occlusion and partial visibility.

We couple the inferred 3D grounding with a library of reusable atomic primitives. We represent tool-use behaviors as a *Bag of Atomic Actions*, a library of short 6D object trajectories indexed by interaction type. For a new scene, the appropriate primitive is retrieved and aligned to the grounded task geometry. To support dexterous-hand execution, we apply the same multi-view grounding to estimate functional contact regions, generate candidate grasps on those regions, and filter them by inverse-kinematics and collision feasibility over the full tool-use trajectory. For long-horizon tasks, closed-loop verification and retry let the system re-ground or replan after execution failures.

We instantiate these ideas in a single zero-shot system and evaluate it on a diverse set of real-world dexterous manipulation tasks. Our main contributions are as follows. First, we present a framework for long-horizon dexterous manipulation that couples VLM-based task planning and multi-view 3D grounding with a library of reusable atomic action primitives and closed-loop verification and retry, enabling zero-shot execution in unseen scenes. Second, we propose a multi-view VLM grounding approach that fuses view-dependent 2D predictions into reliable 3D quantities by combining cross-view triangulation with reference-view ray voting, improving robustness under occlusion and partial visibility. Third, we extend this multi-view grounding to dexterous-hand execution by estimating functional 3D affordance regions for grasp generation and filtering candidate grasps by trajectory-level inverse-kinematics and collision feasibility. Finally, we validate the complete system on a diverse set of real-world manipulation tasks, demonstrating that a unified multi-view VLM grounding framework enables robust zero-shot generalization across diverse manipulation and tool-use tasks involving unseen objects and novel environments.

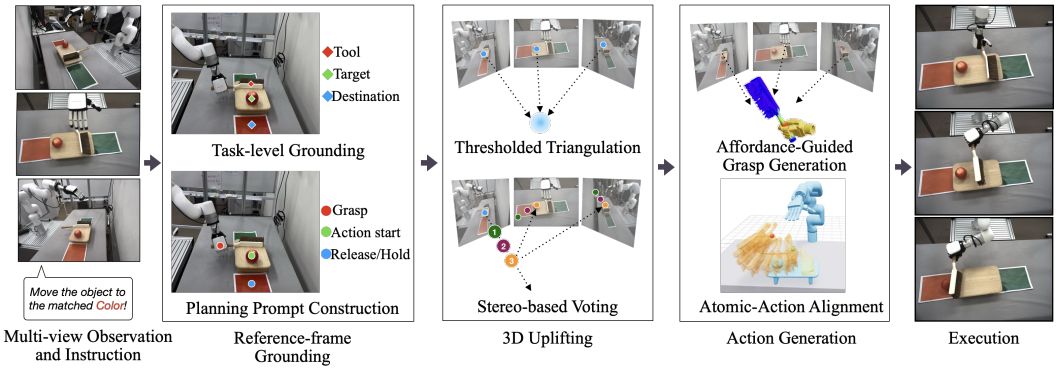

Figure 1: **Overview.** Given a language instruction and calibrated multi-view observations, our framework uses multi-view VLM grounding with robust triangulation and reference-view ray voting to infer task-relevant 3D groundings, generates affordance-aware dexterous grasps, and executes pick-and-place or tool-use plans through reusable action primitives.

2 Related Work

Vision-Language-Action Models for Manipulation Vision-language-action (VLA) models map images and instructions to robot actions, often by adapting vision-language backbones [7, 8, 9] with large robot datasets [10, 11, 12, 13, 14]. Beyond foundational robot policies [1, 2, 4, 3, 5], recent work improves action modeling [5, 15], fine-tuning efficiency [16], spatial grounding [17, 18], open-world generalization [19], and dexterous or humanoid embodiments [20, 15, 21, 22, 23]. Despite strong in-distribution performance, such policies rely on large robot data and often require additional data or adaptation to cover new objects, tools, and embodiments. Retargeting human demonstrations [24, 25, 26] reduces collection cost, but the embodiment gap can yield infeasible contacts that require further refinement. We instead act zero-shot in unseen scenes, without task-specific demonstrations or weight updates.

Zero-Shot Manipulation with Foundation Models Another line drives manipulation directly with pretrained foundation models, without task-specific training, and differs mainly in the intermediate representation it extracts. Code-generation methods prompt large language models to produce high-level plans or executable control code over perception and motion APIs [27, 28, 29]. Visual-prompting methods query vision-language models to mark the task on images as keypoints, visual marks, or affordances [30, 31, 32, 33]. Generative methods predict future frames or subgoal images [34, 35, 36, 37], or dense point and object flows as an embodiment-agnostic action signal [38]. These methods share our zero-shot, training-free motivation, but many still rely on image-space or sparse intermediate representations that are brittle for the task-relevant 3D geometry required for dexterous manipulation, including contact points, placement targets, and tool trajectories. This gap motivates grounding tasks directly in 3D.

3D Grounding and Dexterous Execution Lifting 2D observations to 3D is classically studied in multi-view stereo [6]; for manipulation, recent work gives VLMs spatial reasoning [39, 40], aggregates views into 3D groundings [41], or casts VLM reasoning as 3D constraints [29, 31, 42], often aided by depth or pose foundation models [43, 44]. For execution, prior work transfers object-centric skills or functional correspondences [45, 46], synthesizes dexterous grasps via samplers, generative models, or affordance conditioning [47, 48, 49, 50, 51, 30, 52], and uses language or VLM feedback for verification and replanning [53]. Unlike prior systems that treat 3D grounding, tool motion, and dexterous grasping as separate components, we fuse multi-view VLM groundings with robust triangulation and reference-view ray voting, align object-centric tool trajectories, and generate feasible arm-hand motions within one zero-shot pipeline [54].

3 Method

Our system takes as input calibrated multi-view RGB images and a high-level language instruction, and outputs a physically feasible arm-hand execution plan. Our pipeline consists of four main stages: (1) reference-view semantic grounding (Section 3.1), (2) multi-view fusion-based 3D lifting (Section 3.2), (3) object-centric atomic action alignment for tool-use (Section 3.3), and (4) affordance-guided dexterous grasp and motion generation (Section 3.4). Figure 1 shows the overall pipeline.

3.1 Reference-Frame Grounding

Given M multi-view images $\mathcal{I} = \{I_v\}_{v=1}^{M}$ and a language instruction l, a VLM Φ selects a reference view index $r \in \{1, \dots, M\}$, infers the manipulation mode $z \in \{\text{pick}, \text{tool}\}$, and predicts a mode-specific grounding tuple g_z:

$$(r, z, g_z) = \Phi(\mathcal{I}, l). \tag{1}$$

For pick-and-place, $g_{\text{pick}} = (O_{\text{tar}}, \mathbf{p}_{\text{dst}})$, where O_{tar} is the target object and $\mathbf{p}_{\text{dst}}$ is its 2D destination pixel in I_r. For tool-use, $g_{\text{tool}} = (O_{\text{tool}}, c, O_{\text{tar}}, \mathbf{p}_{\text{dst}})$ additionally identifies the tool object O_{tool}, its skill category $c \in \mathcal{C}_{\text{tool}}$ (e.g., pouring, sweeping), and the target object O_{tar}. If O_{tar} remains stationary, we set $\mathbf{p}_{\text{dst}} = \texttt{NULL}$.

Conditioned on a planning prompt l' combining l, z, and g_z, the VLM generates a sequence of primitives $\mathcal{Q}_r$ on I_r:

$$\mathcal{Q}_r = \Phi(I_r, l') = \{(m_t, \mathcal{P}_r^t)\}_{t=1}^{T}, \qquad \mathcal{P}_r^t = \{(\mathbf{p}_r^{t,j}, d_{t,j})\}_{j=1}^{N_t}, \tag{2}$$

where $m_t \in \mathcal{M} = \{\text{grasp}, \text{apply\_action}, \text{waypoint}, \text{release}, \text{hold}\}$ represents the primitive, and $\mathcal{P}_r^t$ contains 2D keypoints $\mathbf{p}_r^{t,j}$ paired with semantic descriptions $d_{t,j}$ for 3D uplifting. The primitive sequence is structured by z. For pick-and-place, the sequence $(\text{grasp}, \text{waypoint}, \text{release})$ identifies a grasp point on O_{tar}, a transport waypoint, and a placement location. For tool-use, the sequence $(\text{grasp}, \text{apply\_action}, \text{release}/\text{hold})$ identifies both a grasp point and a functional tip (e.g., broom head, kettle spout) on O_{tool}, an interaction point on O_{tar}, and a terminal tool location, where release dictates object release and hold specifies maintaining the grasp. The number of keypoints per primitive is $N_t = 2$ exclusively for the tool-use grasp step, and $N_t = 1$ for all other steps across both modes.

3.2 Multi-View Fusion-Based 3D Lifting

To lift the primitive-level 2D keypoints into 3D while overcoming single-view depth ambiguity and multi-view occlusion, we combine geometric triangulation with reference-view ray voting. For each keypoint $\mathbf{p}_r^{t,j}$, we construct a grounding prompt l'' from l, m_t, and $d_{t,j}$. We then query the VLM across all views to obtain view-wise 2D groundings:

$$\mathbf{p}_v^{t,j} = \Phi(I_v, l''), \qquad v = 1, \dots, M. \tag{3}$$

We first fuse these predictions using RANSAC [55]-style triangulation. For each view pair (a, b), we compute a candidate point $X_{a,b}^{t,j} = \text{Triangulate}(\mathbf{p}_a^{t,j}, \mathbf{p}_b^{t,j})$ and score it by the number of views whose reprojection error falls below a pixel threshold ϵ_{tri}:

$$S_{\text{tri}}(a, b) = \sum_{v=1}^{M} \mathbf{1}\left[\left\| \pi_v(X_{a,b}^{t,j}) - \mathbf{p}_v^{t,j} \right\|_2 \le \epsilon_{\text{tri}}\right]. \tag{4}$$

The candidate with the largest consensus support is defined as $X_{\text{tri}}^{t,j} = X_{a^*,b^*}^{t,j}$ where $(a^*, b^*) = \arg\max_{a,b} S_{\text{tri}}(a, b)$.

As a complementary estimate, we perform reference-view ray voting. We sample N_δ depth candidates $X_n^{t,j}$ along the camera ray of $\mathbf{p}_r^{t,j}$ and project them into each non-reference view $v \neq r$ overlaid with numbered visual markers to construct a modified image $\tilde{I}_v^{t,j}$. The VLM selects the

indices $\mathcal{C}_v^{t,j} = \Phi(\tilde{I}_v^{t,j}, l'') \subset \{1, \ldots, N_\delta\}$ that best match $d_{t,j}$. Aggregating these selections yields the voting candidate $X_{\text{vote}}^{t,j}$:

$$S_{\text{vote}}^{t,j}(n) = \sum_{v \neq r} \mathbf{1}[n \in \mathcal{C}_v^{t,j}], \qquad X_{\text{vote}}^{t,j} = X^{t,j}_{\arg\max_n S_{\text{vote}}^{t,j}(n)}. \tag{5}$$

The final 3D keypoint $X_\star^{t,j}$ is dynamically selected based on the geometric consensus of the triangulation step. If the maximum triangulation score meets an inlier threshold τ_{tri}, we adopt $X_{\text{tri}}^{t,j}$; otherwise, we fall back to the robust voting estimate $X_{\text{vote}}^{t,j}$:

$$X_\star^{t,j} = \begin{cases} X_{\text{tri}}^{t,j} & \text{if } \max_{a,b} S_{\text{tri}}(a,b) \geq \tau_{\text{tri}}, \\ X_{\text{vote}}^{t,j} & \text{otherwise.} \end{cases} \tag{6}$$

The complete set of lifted keypoints is defined as $\mathcal{X} = \{X_\star^{t,j} \mid t = 1, \ldots, T,\ j = 1, \ldots, N_t\}$. Each $X_\star^{t,j}$ is transformed into the world frame using calibrated camera extrinsics.

3.3 Object-Centric Atomic Action Alignment

For pick-and-place tasks, the desired object transfer trajectory can be constructed from the current pose of O_{tar} to the lifted release keypoint in $\mathcal{X}$ using an off-the-shelf robot motion generation pipeline [54]. In contrast, tool-use tasks require an additional motion prior that specifies how the tool should move relative to the target object. To this end, we introduce a *Bag of Atomic Actions*: a reusable library of object-centric primitives that encode how a tool moves relative to a target. Each atomic action is defined as

$$\mathcal{A} = (c, \mathcal{T}, X_{\text{s}}, X_{\text{e}}), \qquad \mathcal{T} = \{T_i\}_{i=0}^{N_a}, \quad T_i \in SE(3), \tag{7}$$

where $c \in \mathcal{C}_{\text{tool}}$ is a predefined tool-use skill category, $\mathcal{T}$ is a 6D trajectory of the tool object, and $X_{\text{s}}, X_{\text{e}} \in \mathbb{R}^3$ are the stored start and end anchor points of the tool motion. The atomic action library is constructed offline from recorded demonstrations and generated tool trajectories. In our implementation, generated trajectories are obtained using VLMPose [45].

At test time, we retrieve the atomic action with the matching skill category c from the library and align its stored tool trajectory to the current scene using the lifted keypoints $\mathcal{X}$. Since tool-use plans always include an apply_action primitive and terminate with either release or hold, we denote by $X_{\text{app}} \in \mathcal{X}$ the lifted apply_action keypoint and by $X_{\text{term}} \in \mathcal{X}$ the lifted terminal keypoint from the release or hold primitive.

To align the stored action with the current interaction state, we determine a rigid transformation $T_{\text{align}} \in SE(3)$ that maps the stored start and end anchors (X_s, X_e) to the target scene keypoints $(X_{\text{app}}, X_{\text{term}})$. We apply this spatial alignment to the stored 6D tool trajectory, yielding the scene-aligned trajectory $\hat{\mathcal{T}}$:

$$\hat{\mathcal{T}} = \{\hat{T}_i\}_{i=0}^{N_a}, \qquad \hat{T}_i = T_{\text{align}} \cdot T_i. \tag{8}$$

The aligned 6D trajectory $\hat{\mathcal{T}}$ is then passed as $\mathcal{T}_{\text{obj}}$ to the downstream module for tool-use grasp and arm-hand motion generation in Section 3.4.

3.4 Dexterous Affordance-Guided Grasp and Motion Generation

The lifted grasp keypoint $X_{\text{grasp}} \in \mathcal{X}$ provides a semantic anchor for manipulation, but a dexterous grasp additionally requires a task-conditioned contact region on the object surface. To define this functional region, let O_m denote the manipulated object ($O_m = O_{\text{tar}}$ for pick-and-place; $O_m = O_{\text{tool}}$ for tool-use). For each view v, we project the grasp keypoint as $\mathbf{p}_v^{\text{grasp}} = \pi_v(X_{\text{grasp}})$ and query the VLM with an affordance prompt l''' combining l, $\mathbf{p}_v^{\text{grasp}}$, and its description d_{grasp} to predict a 2D graspable bounding box:

$$B_v = \Phi(I_v, l'''), \qquad v = 1, \ldots, M. \tag{9}$$

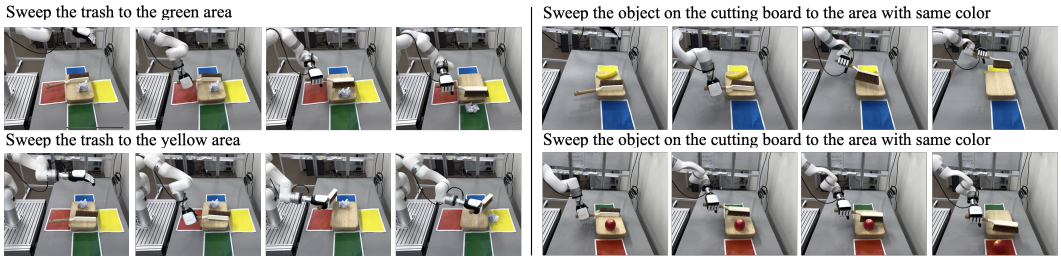

Figure 2: **Qualitative Results.** Given each high-level instruction l, our system infers 3D groundings and, for tool-use cases, aligns an object-centric atomic action to the current scene. We evaluate both direct and indirect styles of instructions and demonstrate successful 3D grounding across diverse environments.

To lift these view-wise affordance boxes into 3D world space, we project each vertex q_i of the mesh of O_m into all views to compute a multi-view inclusion score $s(q_i)$:

$$s(q_i) = \frac{1}{M} \sum_{v=1}^{M} \mathbf{1}[\pi_v(q_i) \in B_v]\,. \tag{10}$$

The 3D affordance region is then defined by filtering the mesh vertices with a voting threshold τ:

$$\mathcal{R}_{\text{aff}} = \{q_i \mid s(q_i) \geq \tau\}. \tag{11}$$

We generate dexterous grasp candidates G from $\mathcal{R}_{\text{aff}}$ using a cylindrical template sampler for handle-like affordances and an optimization-based generator for general geometries.

To ensure physical feasibility before trajectory generation, we apply collision-aware position refinement to grounding points that define object placement or terminal tool poses. For an unrefined grounding point X_{loc}, we search a local vertical grid $\mathcal{G}(X_{\text{loc}})$ to find the nearest collision-free position:

$$X^*_{\text{loc}} = \arg \min_{X' \in \mathcal{G}(X_{\text{loc}})} \|X' - X_{\text{loc}}\|_2 \quad \text{s.t.} \quad \phi_m(X') = 0, \tag{12}$$

where $\phi_m(\cdot)$ denotes the environment penetration depth. If no valid candidate exists, the grounding is considered failed.

Using the refined keypoints, we construct the desired 6D trajectory $\mathcal{T}_{\text{obj}}$ of O_m, which corresponds to the transfer path for pick-and-place or the aligned tool trajectory $\hat{\mathcal{T}}$ from Section 3.3 for tool-use. For each grasp candidate $g \in G$, an off-the-shelf arm-hand motion generator [54] tracks $\mathcal{T}_{\text{obj}}$ to resolve kinematic and collision constraints:

$$(\mathcal{T}_{\text{robot}}, \eta) = f_{\text{motion}}(\mathcal{T}_{\text{obj}}, g), \tag{13}$$

where $\mathcal{T}_{\text{robot}}$ is the generated execution trajectory and $\eta \in \{0, 1\}$ indicates overall physical feasibility. A feasible pair satisfying $\eta = 1$ is selected for physical robot execution. Complete technical details regarding the grasp optimization objectives and sampling templates are deferred to the supplementary material.

4 Experiments

We evaluate our framework on zero-shot robot manipulation in a real-world tabletop setting, assessing its scalability from simple tasks to long-horizon scenarios. Our evaluation covers four key capabilities: (1) target grounding amidst distractors and collision robustness (e.g., placing inferred trash into a basket), (2) spatial-relation reasoning (e.g., placing tools on a stove), (3) affordance-aware tool use (e.g., sweeping objects with a broom), and (4) long-horizon sequencing (e.g., cooking and organizing 3–4 objects). Additional tool-use scenarios are provided in the supplementary material.

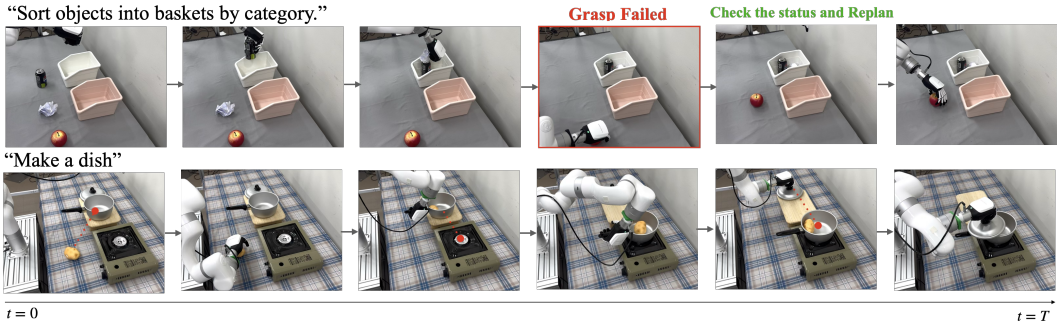

Figure 3: **Qualitative Results.** Long-horizon manipulation examples. The shown scenarios consist of multiple subtasks. In the example above, the grasp fails, and the VLM detects the failure state and replans the next action.

4.1 Hardware Setup

The system features an xArm equipped with an Inspire dexterous hand. The tabletop environment is monitored by multiple calibrated RGB cameras, including a stereo pair. We use FoundationStereo [43] for stereo depth estimation and FoundationPose [44] for multi-object 6D pose estimation.

4.2 Baselines

We compare our zero-shot framework against an RGB-D grounding baseline and two Vision-Language-Action (VLA) models [20, 22]. The RGB-D baseline predicts a 2D keypoint from a single view and lifts it to 3D using the aligned depth map. For the VLA models, we fine-tune pretrained models using 30 task-specific teleoperation demonstrations per task, whereas our method operates entirely zero-shot, relying solely on VLM reasoning for 3D grounding and manipulation.

4.3 Metrics

Success Rate. A trial is considered successful if the robot completes the task according to the text instruction. For tasks with a specified target object or target location, we check whether the target object is placed at the desired location after execution.

Localization Distance. We measure the object-centric localization error of primitive-level 3D groundings. For each evaluated primitive keypoint, we compute the distance between the predicted 3D grounding and the center of the corresponding relevant object or region. In Table 3, we report the grasp-point error L_{grasp} and the apply-action error L_{apply}:

$$L_{\text{grasp}} = \|\hat{X}_{\text{grasp}} - c_{\text{grasp}}\|_2, \qquad L_{\text{apply}} = \|\hat{X}_{\text{apply}} - c_{\text{apply}}\|_2,$$

where $\hat{X}_{\text{grasp}}$ and $\hat{X}_{\text{apply}}$ are the predicted 3D grasp and apply-action groundings, and c_{grasp} and c_{apply} denote the centers of their corresponding evaluation targets.

Collision Error. We evaluate whether the predicted waypoint or placement grounding causes collision when the manipulated object is placed at the corresponding location. We report

$$\phi_m(X_{\text{wp}}),$$

where X_{wp} denotes the predicted waypoint or placement grounding in the world frame, and $\phi_m(\cdot)$ measures the mean maximum penetration depth between the manipulated object and the surrounding environment.

Long-Horizon Success Rate. For sequential tasks, a trial is considered successful only if all required steps are completed in the correct order. Because long-horizon real-robot trials are time-consuming, the number of trials may differ across tasks. We report both the number of trials and the

Table 1: Success rates on real-robot manipulation tasks.

Task	RGBD	Ours
Throw Away Trash	4/5	**5/5**
Place Pot on Stove	4/5	4/5
Cluttered Precise P&P	2/5	**4/5**

Table 2: Comparison with VLA baselines. Each VLA model is finetuned with 30 task-specific teleoperation demonstrations per task.

Task	GR00T [20]	Being-HO [22]	Ours
Throw Away Trash	0/5	0/5	**10/10**
Broom Clean	0/5	0/5	**8/10**

Table 3: We report object-centric localization errors for the grasp and apply-action groundings, L_{grasp} and L_{apply}, and the collision error $\phi_m(X_{\text{wp}})$.

Method	L_{grasp} (cm)↓	L_{apply} (cm)↓	$\phi_m(X_{\text{wp}})$ ↓
Stereo (RGB-D)	16.43	2.72	9.91
Ours (2 views)	**4.58**	1.70	9.81
Ours (3 views)	4.60	**1.35**	10.95
Ours (5 views)	4.77	1.94	9.78
Ours (*w/ refinement*)	4.77	1.63	**9.60**

Table 4: Success rates for long-horizon manipulation tasks, reported per sub-step and end-to-end completion. Successes after retries within the retry budget are included.

Task	Step 1	Step 2	Step 3	Step 4	End-to-End
Organize Objects	6/6	5/6	3/5	3/3	4/6
Cooking	3/3	3/3	1/3	–	1/3

success rate. When retries are used, we count a trial as successful if the task is completed within the retry budget.

4.4 Results

Real-robot success. Table 1 compares our method with the single-view RGB-D grounding baseline on real-robot tasks. Our method matches or improves the RGB-D baseline, achieving $5/5$ on "Throw Away Trash" and $4/5$ on "Place Pot on Stove". The advantage of our approach becomes more pronounced in cluttered scenes and tasks requiring precise placement, where multi-view 3D grounding provides more reliable localization under occlusion and viewpoint ambiguity. In particular, our method achieves $4/5$ on the "Cluttered Precise Pick-and-Place" task, compared to $2/5$ for the RGB-D baseline. Table 2 further shows that two VLA baselines fine-tuned with 30 task-specific demonstrations fail in all trials, while our zero-shot system succeeds on both evaluated tasks.

3D grounding quality. Table 3 evaluates grounding accuracy and collision error. Compared with the RGB-D baseline, our multi-view fusion substantially reduces grasp localization error and improves apply-action localization. Increasing the number of views gives diminishing returns in our scenes, where wide-baseline pairs already provide strong geometric constraints. Collision-aware refinement (Section 3.4) gives the lowest penetration error.

Long-horizon manipulation. Table 4 reports per-step and end-to-end success rates for long-horizon tasks. Failures mainly arise from arm joint limits, environmental collisions, or unstable grasps. When failures occur, closed-loop retry can recover by re-grounding or replanning the failed subtask within the retry budget. Additional details are provided in the supplementary material.

5 Discussion

We present a zero-shot, long-horizon manipulation framework that bridges VLM reasoning with physical execution via multi-view 3D grounding. By decomposing language instructions into sequences of 3D-grounded manipulation primitives, our system seamlessly supports both standard pick-and-place and complex tool-use tasks by spatially aligning object-centric atomic actions to the target scene. Experimental results demonstrate that our multi-view fusion strategy significantly outperforms single-view RGB-D baselines in spatial accuracy and robustness against occlusion. Furthermore, the primitive-level formulation naturally enables closed-loop execution, allowing the system to verify task progress and dynamically recover from intermediate failures during long-horizon tasks.

Limitations and Future Work. Despite its capabilities, our framework exhibits a few limitations. First, the system's performance is inherently bounded by the reasoning reliability of the underlying 2D VLM. Although multi-view lifting and local collision-aware refinement improve the geometric consistency of grounded predictions, errors in task decomposition, affordance selection, or semantic grounding in 2D can still propagate to downstream execution failures. Second, the physical execution relies on off-the-shelf arm-hand motion planners; consequently, kinematic singularities, collision-checker timeouts, or unstable grasp executions can still induce failures and increase the overall inference-to-execution latency. Third, our current formulation focuses on object-centric manipulation and tool use, and does not explicitly support dexterous in-hand manipulation, such as rotating an object within the hand, operating scissors, or pressing buttons on handheld tools during execution. Future work will explore integrating native 3D vision-language models and reactive low-level policies to further enhance the speed and robustness of zero-shot dexterous manipulation.

References

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[2] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *CoRL*, 2023.

[3] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[4] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[5] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[6] Y. Furukawa and C. Hernández. Multi-view stereo: A tutorial. *FnT CGV*, 9(1-2):1–148, 2015.

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

[8] Z. Chen, J. Wu, W. Wang, W. Su, G. Chen, S. Xing, M. Zhong, Q. Zhang, X. Zhu, L. Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *CVPR*, 2024.

[9] S. Bai, Y. Cai, R. Chen, K. Chen, X. Chen, Z. Cheng, L. Deng, W. Ding, C. Gao, C. Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[10] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

[11] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *CoRL*, 2023.

[12] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *ICRA*, 2024.

[13] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[14] Q. Bu, J. Cai, L. Chen, X. Cui, Y. Ding, S. Feng, S. Gao, X. He, X. Hu, X. Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

[15] J. Wen, Y. Zhu, J. Li, Z. Tang, C. Shen, and F. Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.

[16] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[17] D. Qu, H. Song, Q. Chen, Y. Yao, X. Ye, Y. Ding, Z. Wang, J. Gu, B. Zhao, D. Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

[18] Q. Li, Y. Liang, Z. Wang, L. Luo, X. Chen, M. Liao, F. Wei, Y. Deng, S. Xu, Y. Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

[19] G. R. Team, A. Abdolmaleki, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, A. Balakrishna, N. Batchelor, A. Bewley, J. Bingham, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025.

[20] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[21] Y. Zhong, X. Huang, R. Li, C. Zhang, Z. Chen, T. Guan, F. Zeng, K. N. Lui, Y. Ye, Y. Liang, et al. Dexgraspvla: A vision-language-action framework towards general dexterous grasping. In *AAAI*, 2026.

[22] H. Luo, Y. Feng, W. Zhang, S. Zheng, Y. Wang, H. Yuan, J. Liu, C. Xu, Q. Jin, and Z. Lu. Being-h0: vision-language-action pretraining from large-scale human videos. *arXiv preprint arXiv:2507.15597*, 2025.

[23] D. Kim, H. Jang, M. Koo, S. Jang, T. Kim, B. Kim, B. Yoon, C. Jang, D. Choi, D. Han, et al. Rldx-1 technical report. *arXiv preprint arXiv:2605.03269*, 2026.

[24] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. In *ECCV*, 2022.

[25] A. Sivakumar, K. Shaw, and D. Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube. *arXiv preprint arXiv:2202.10448*, 2022.

[26] C. Wang, H. Shi, W. Wang, R. Zhang, L. Fei-Fei, and C. K. Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *arXiv preprint arXiv:2403.07788*, 2024.

[27] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *ICRA*, 2023.

[28] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Program generation for situated robot task planning using large language models. *AMR*, 47(8):999–1012, 2023.

[29] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

[30] F. Liu, K. Fang, P. Abbeel, and S. Levine. Moka: Open-world robotic manipulation through mark-based visual prompting. *arXiv preprint arXiv:2403.03174*, 2024.

[31] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

[32] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, D. Driess, A. Wahid, Z. Xu, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. *arXiv preprint arXiv:2402.07872*, 2024.

[33] W. Yuan, J. Duan, V. Blukis, W. Pumacay, R. Krishna, A. Murali, A. Mousavian, and D. Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*, 2024.

[34] Y. Du, S. Yang, B. Dai, H. Dai, O. Nachum, J. Tenenbaum, D. Schuurmans, and P. Abbeel. Learning universal policies via text-guided video generation. *NeurIPS*, 2023.

[35] P.-C. Ko, J. Mao, Y. Du, S.-H. Sun, and J. B. Tenenbaum. Learning to act from actionless videos through dense correspondences. In *ICLR*, 2024.

[36] K. Black, M. Nakamoto, P. Atreya, H. Walke, C. Finn, A. Kumar, and S. Levine. Zero-shot robotic manipulation with pre-trained image-editing diffusion models. In *ICLR*, 2024.

[37] J. Liang, R. Liu, E. Ozguroglu, S. Sudhakar, A. Dave, P. Tokmakov, S. Song, and C. Vondrick. Dreamitate: Real-world visuomotor policy learning via video generation. *arXiv preprint arXiv:2406.16862*, 2024.

[38] C. Yuan, C. Wen, T. Zhang, and Y. Gao. General flow as foundation affordance for scalable robot learning. *arXiv preprint arXiv:2401.11439*, 2024.

[39] B. Chen, Z. Xu, S. Kirmani, B. Ichter, D. Sadigh, L. Guibas, and F. Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *CVPR*, 2024.

[40] C. H. Song, V. Blukis, J. Tremblay, S. Tyree, Y. Su, and S. Birchfield. Robospatial: Teaching spatial understanding to 2d and 3d vision-language models for robotics. In *CVPR*, 2025.

[41] R. Xu, Z. Huang, T. Wang, Y. Chen, J. Pang, and D. Lin. Vlm-grounder: A vlm agent for zero-shot 3d visual grounding. *arXiv preprint arXiv:2410.13860*, 2024.

[42] M. Pan, J. Zhang, T. Wu, Y. Zhao, W. Gao, and H. Dong. Omnimanip: Towards general robotic manipulation via object-centric interaction primitives as spatial constraints. In *CVPR*, 2025.

[43] B. Wen, M. Trepte, J. Aribido, J. Kautz, O. Gallo, and S. Birchfield. Foundationstereo: Zero-shot stereo matching. In *CVPR*, 2025.

[44] B. Wen, W. Yang, J. Kautz, and S. Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *CVPR*, 2024.

[45] S. Baik, G. Kim, M. Choi, and H. Joo. Text-guided 6d object pose rearrangement via closed-loop vlm agents. *arXiv preprint arXiv:2604.09781*, 2026.

[46] C. Tang, A. Xiao, Y. Deng, T. Hu, W. Dong, H. Zhang, D. Hsu, and H. Zhang. Mimicfunc: Imitating tool manipulation from a single human video via functional correspondence. *arXiv preprint arXiv:2508.13534*, 2025.

[47] R. Wang, J. Zhang, J. Chen, Y. Xu, P. Li, T. Liu, and H. Wang. Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation. *arXiv preprint arXiv:2210.02697*, 2022.

[48] Y. Xu, W. Wan, J. Zhang, H. Liu, Z. Shan, H. Shen, R. Wang, H. Geng, Y. Weng, J. Chen, et al. Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. In *CVPR*, 2023.

[49] P. Li, T. Liu, Y. Li, Y. Geng, Y. Zhu, Y. Yang, and S. Huang. Gendexgrasp: Generalizable dexterous grasping. In *ICRA*, 2023.

[50] Y. Zhong, Q. Jiang, J. Yu, and Y. Ma. Dexgrasp anything: Towards universal robotic dexterous grasping with physics awareness. In *CVPR*, 2025.

[51] J. Chen, Y. Ke, L. Peng, and H. Wang. Dexonomy: Synthesizing all dexterous grasp types in a grasp taxonomy. *arXiv preprint arXiv:2504.18829*, 2025.

[52] S. Nasiriany, S. Kirmani, T. Ding, L. Smith, Y. Zhu, D. Driess, D. Sadigh, and T. Xiao. Rt-affordance: Affordances are versatile intermediate representations for robot manipulation. 2025.

[53] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[54] B. Sundaralingam, A. Murali, and S. Birchfield. curobov2: Dynamics-aware motion generation with depth-fused distance fields for high-dof robots. *arXiv preprint arXiv:2603.05493*, 2026.

[55] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *CACM*, 24(6):381–395, 1981.

[56] J. Chen, Y. Ke, and H. Wang. Bodex: Scalable and efficient robotic dexterous grasp synthesis using bilevel optimization. In *ICRA*, 2025.

Supplementary Material

A Additional Qualitative Examples

A.1 Comparison of 3D Grounding Methods

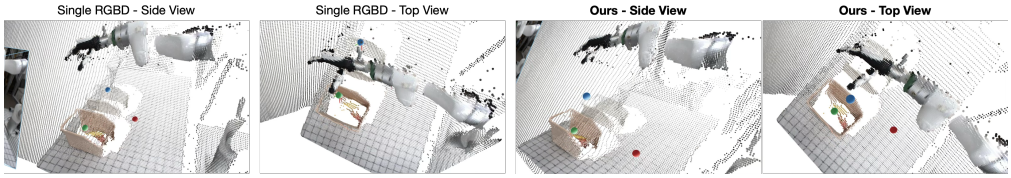

Figure S1: Comparison of grounding results produced by the single-view RGB-D baseline and our multi-view grounding method in a cluttered scene. Red, green, and blue spheres indicate the predicted grasp, waypoint, and destination, respectively.

We further analyze the behavior of the single-view RGB-D grounding baseline and our multi-view grounding approach in cluttered real-world scenes. Fig. S1 shows representative grounding results for grasp, waypoint, and destination prediction. Due to its reliance on a single observation, the RGB-D baseline is sensitive to occlusion and incomplete geometry, often resulting in misplaced 3D targets. In contrast, our multi-view approach aggregates semantic grounding cues across views and produces more consistent task-relevant 3D estimates in cluttered environments.

A.2 Object-centric Atomic Action Library

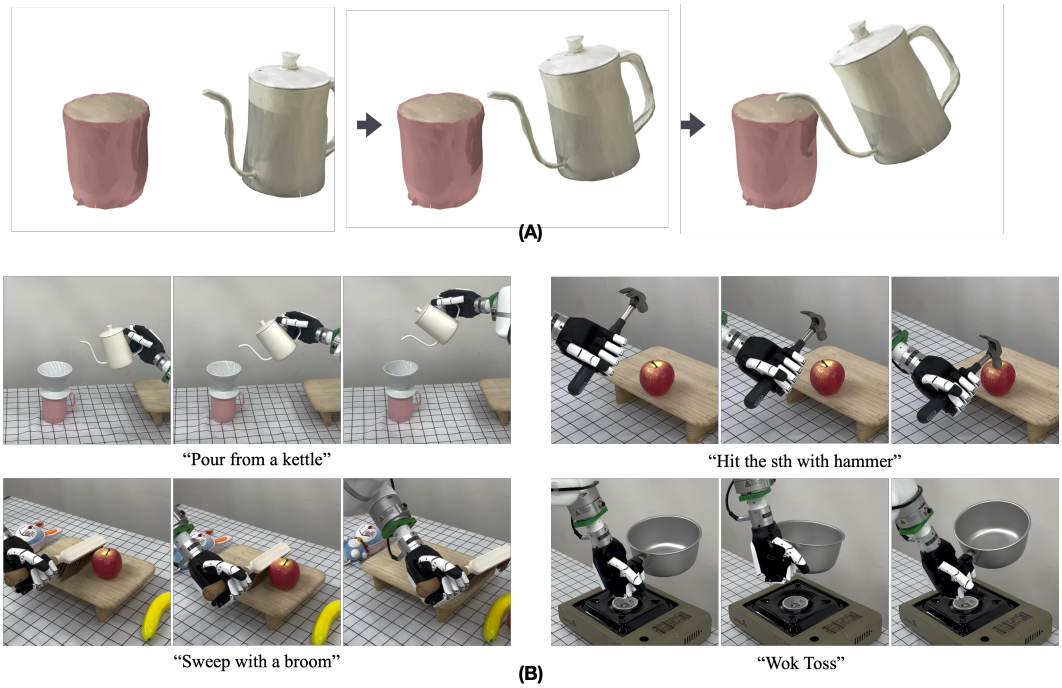

Figure S2: (A): Example object trajectory generated by [45] for the prompt "Pour water from the kettle." (B): Examples of object-centric atomic actions executed on the real robot.

Bag of Atomic Actions. Following the BoAA formulation in Sec.3.3, we provide additional examples of instantiated atomic actions. Each action stores a tool-use skill category, an object-centric 6D tool trajectory, and start/end anchor points used to align the stored motion to the current scene. These

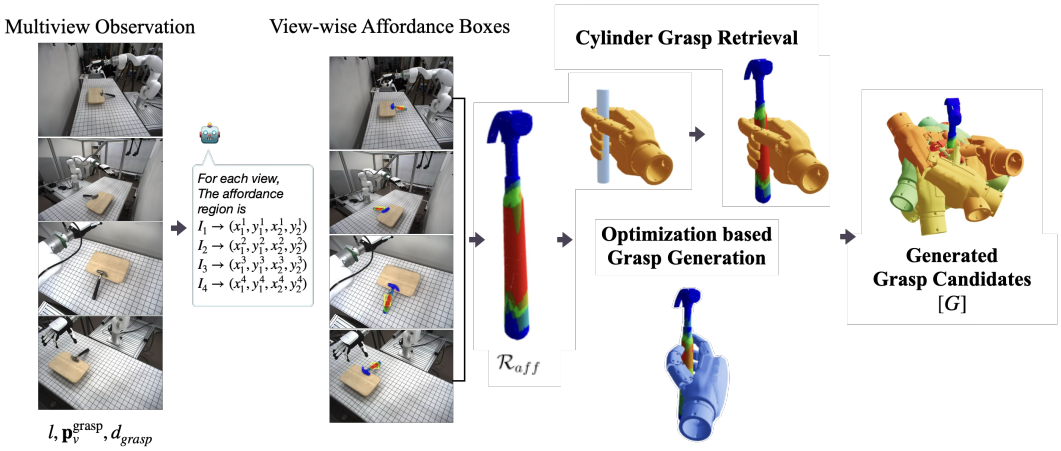

Figure S3: Visualization for affordance-grounding and grasp generation pipeline. Affordance bounding boxes from multi-view are combined to generate 3D affordance region.

trajectories are either recorded from demonstrations or generated using existing trajectory generation methods [45], as shown in Fig. S2-A. We demonstrate four representative examples: pouring from a kettle, sweeping with a broom, hammering, and wok tossing, as shown in the supplementary video and Fig. S2-B.

B Grasp Generation

In Sec.3.4 of the main paper, we use the 3D affordance region $\mathcal{R}_{\text{aff}}$ as the contact prior for dexterous grasp generation; the pipeline is shown in Fig. S3. Here, we describe the two grasp generators used in our implementation: a cylindrical-template-based sampler for handle-like affordances and an affordance-aware optimization-based generator [56] for general object geometries.

B.1 Cylindrical-Template-Based Grasp Generation

For tool-use tasks, directly optimizing fingertip contacts can be insufficient, because successful tool use requires grasps that remain stable and action-consistent throughout motion execution. We observe that many household tools contain approximately cylindrical grasp affordances, such as broom handles, bottles, and pan handles. When the estimated affordance region $\mathcal{R}_{\text{aff}}$ corresponds to such a cylindrical region, we exploit this structural prior to initialize palm poses.

Specifically, we sample a surface vertex $\mathbf{q} \in \mathcal{R}_{\text{aff}}$ near the region center and use its outward surface normal $\mathbf{n}_\mathbf{q}$ to define a palm pose anchor:

$$\mathbf{p}_{\text{palm}} = \mathbf{q} + d\,\mathbf{n}_\mathbf{q}, \qquad \mathbf{n}_{\text{palm}} = -\mathbf{n}_\mathbf{q}, \qquad d \sim \mathcal{U}(0, 5\ \text{cm}). \tag{14}$$

Here, $\mathbf{p}_{\text{palm}}$ is the palm reference point, $\mathbf{n}_{\text{palm}}$ is the desired palm normal, and d controls the palm-to-surface offset. To cover diverse grasp styles, we sample different palm orientations around the approach direction while preserving the normal alignment.

For each sampled palm pose, we optimize finger closure using the simulation-based grasp refinement procedure of Dexonomy [51]. The resulting candidates are validated in simulation by applying external forces and torques along all six axes to assess grasp stability.

B.2 Affordance-Aware Optimization-Based Grasp Generation

For general, non-handle object geometries, we build upon the optimization-based grasp generator BODex [56] by augmenting its objective function with an affordance-guided contact term. This term biases fingertip placements toward the task-conditioned affordance region $\mathcal{R}_{\text{aff}}$, while preserving

the original force-closure and collision objectives of BODex. This complements the cylindrical-template-based sampler in Sec. B.1, which is used for handle-like affordances. Let

$$\mathcal{P}_{\text{hand}} = \{\mathbf{p}_1^h, \ldots, \mathbf{p}_{N_p}^h\}, \qquad N_p = 5, \tag{15}$$

denote the fingertip positions of a candidate dexterous grasp. We encourage fingertip contacts to lie near the affordance region using a nearest-neighbor distance objective:

$$\mathcal{L}_{\text{dist}} = \frac{1}{N_p} \sum_{i=1}^{N_p} \min_{\mathbf{q} \in \mathcal{R}_{\text{aff}}} \|\mathbf{p}_i^h - \mathbf{q}\|_2. \tag{16}$$

For the directional objective, let $\mathbf{p}_i^o$ be the nearest affordance vertex to $\mathbf{p}_i^h$:

$$\mathbf{p}_i^o = \text{argmin}_{\mathbf{q} \in \mathcal{R}_{\text{aff}}} \|\mathbf{p}_i^h - \mathbf{q}\|_2. \tag{17}$$

To further encourage physically meaningful contacts, we align the direction from the affordance surface to the fingertip with the outward surface normal. Let $\hat{\mathbf{n}}_i$ be the outward normal at $\mathbf{p}_i^o$. We define

$$\cos\theta_i = \frac{\mathbf{p}_i^h - \mathbf{p}_i^o}{\|\mathbf{p}_i^h - \mathbf{p}_i^o\|_2} \cdot \hat{\mathbf{n}}_i, \tag{18}$$

and penalize contacts whose alignment is below a margin γ:

$$\mathcal{L}_{\text{dir}} = \frac{1}{N_p} \sum_{i=1}^{N_p} \text{ReLU}(\gamma - \cos\theta_i), \qquad \gamma = 0.2. \tag{19}$$

The final affordance-aware objective is

$$\mathcal{L}_{\text{aff}} = w_{\text{dist}} \mathcal{L}_{\text{dist}} + w_{\text{dir}} \mathcal{L}_{\text{dir}}, \tag{20}$$

which is optimized jointly with the original objectives of BODex. This encourages the generated grasp to remain physically stable while placing contacts on the task-relevant affordance region.

C Real-world Experiments

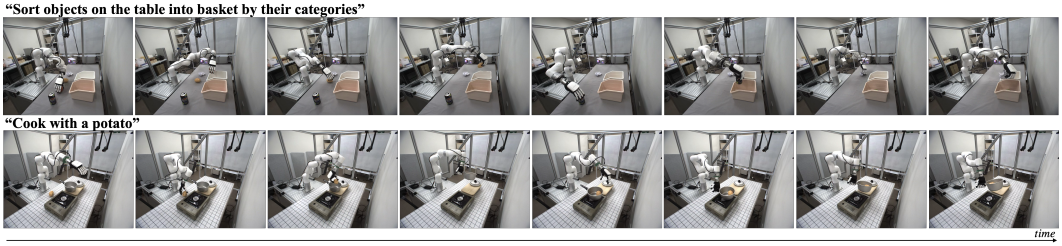

Figure S4: Example of long horizon manipulations

We use 4–6 RGB cameras, two of which are used by a stereo-based metric depth estimator [43]. Across all tasks and test settings, object configurations and language instructions are randomized. Additional examples are shown in the supplementary video.

C.1 Application to Long-Horizon Manipulation

We extend the single-step manipulation pipeline in the main paper to long-horizon tasks by wrapping it with a high-level planning and closed-loop verification stage. At time t, let $\mathcal{I}_t = \{I_{t,v}\}_{v=1}^M$ denote the current calibrated multi-view observations. Given the initial observations $\mathcal{I}_0$ and a high-level language instruction l, the VLM first produces an initial scene description d_0 and a subtask queue $\mathcal{L}^0 = (l_1^0, \ldots, l_S^0)$:

$$(\mathcal{L}^0, d_0) = \Phi(\mathcal{I}_0, l). \tag{21}$$

Each subtask l_t is then executed using the same 3D-grounded manipulation pipeline described in the main paper: the VLM performs reference-frame grounding, predicts primitive-level keypoints, lifts them to 3D using multi-view fusion, and generates a feasible arm-hand motion.

After each execution attempt, we acquire updated multi-view observations and query the VLM to verify task progress and update the remaining plan:

$$(\alpha_t, d_{t+1}, \mathcal{L}^{t+1}) = \Phi(\mathcal{I}_{t+1}, l, \mathcal{L}^t, d_t), \tag{22}$$

where $\alpha_t \in \{\text{continue}, \text{retry}, \text{replan}, \text{done}\}$ denotes the closed-loop decision, d_{t+1} is the updated scene description, and $\mathcal{L}^{t+1}$ is the remaining subtask queue. If $\alpha_t = \text{continue}$, the system proceeds to the next subtask. If $\alpha_t = \text{retry}$, the failed subtask is re-executed with fresh 3D grounding from $\mathcal{I}_{t+1}$. If $\alpha_t = \text{replan}$, the remaining subtask queue is replaced by the updated plan $\mathcal{L}^{t+1}$. This allows the system to recover from intermediate failures, object displacement, or newly observed scene changes within the retry budget. In experiments, we set the retry budget to four.

D Details

D.1 Implementation Details

We use the following hyperparameters in all experiments.

Multi-view Fusion. Following the VLM-based reference-view selection described in the main paper, we use 4–6 input views for multi-view fusion. For RANSAC-based triangulation, we set the reprojection inlier threshold to $\epsilon_{\text{tri}} = 20$ pixels for images of resolution 640×480. A triangulated point is accepted when the consensus score exceeds $\tau_{\text{tri}} = 0.5 \times M$ of the available views; otherwise, the system falls back to the reference-view ray-voting estimate. For ray voting, candidate 3D points are uniformly sampled along the reference camera ray over the depth range $[0.5, 2.0]$m with a step size of 0.05m.

Affordance Region Generation. For 3D affordance region generation, we retain mesh vertices whose multi-view inclusion score satisfies $s(q_i) \geq \tau$, where $\tau = 0.65$.

Collision-aware Refinement. For grounding-position refinement, candidate keypoints are sampled within a local neighborhood of ± 4cm around the initial estimate using a grid resolution of 2cm. Vertical-axis adjustments are prioritized over lateral displacements to preserve the intended grounding location.

Dexterous Grasp Generation. For cylindrical affordances, the palm offset d is sampled from $0.00, 0.01, 0.02, 0.03$m. In affordance-aware optimization, we set the directional margin to $\gamma = 0.2$ and use objective weights $w_{\text{dist}} = 1.0$ and $w_{\text{dir}} = 0.5$ for the distance and directional terms, respectively.

D.2 Querying Vision-Language Model

We use *gemini-robotics-er-1.6-preview* as the vlm model for our experiments.

Long-horizon Planner

```
You are a robot task planner.  You see the CURRENT scene image.  The OVERALL GOAL is fixed; the scene may have changed
mid-execution.
Between steps a human may ADD, REMOVE, or MOVE objects --- the CURRENT IMAGE is the single source of truth; re-read it
fully, plan for any new goal-relevant object, and do not assume the scene matches the history.
OVERALL GOAL:
{goal}
ORIGINAL TASK PLAN (the task breakdown you produced at the very first step):
{initial_plan}
Refer to this ORIGINAL TASK PLAN --- it holds the destination you already decided for each object.  Stay consistent with
those destinations; only deviate if the current image clearly requires it.
EXECUTION HISTORY:
{history_block}
CLAIMED STATE (may be wrong --- verify against the image):
{claimed_state}
Emit ONE JSON object:
{
"scene":      "<one line describing the scene:  every visible object with its position / relative layout, and include
how many of each category --- a natural description>",
"holding":  {
"is_holding":  true | false,
"object":          "<name>" | null,
"note":              "<short reason if it disagrees with the log>"
},
"status":    "<one line:  done / remaining>",
"tasks":      ["<task string>", ...]
}
### Task Planning
TASK RULES --- each task is a SELF-CONTAINED, SINGLE-OBJECT manipulation
(one moving object, one goal) phrased in NATURAL LANGUAGE. It must be ONE
of these 3 atomic actions:
1.  PICK + PLACE  --- pick an object and put it on/in a destination
2.  RELEASE       --- drop the currently held object on/in a destination (after tool action case)
3.  TOOL ACTION   --- act on a target with a tool
(sweep / wipe / cut / push / write / pour / press / ...).
If not currently holding, the pick-up of the tool is part of this same task;
if already holding the tool, just continue the action with it.
Phrasing:
- Plain string, natural language.  No pixel coordinates.
- Use object names / colors VISIBLE in the current image --- no hallucination.
Do NOT emit tasks whose objects aren't in the scene.
- Keep the OVERALL GOAL's original language.
Ordering --- pick the MOST APPROPRIATE atomic actions for the goal and
arrange them in a sensible execution order:
- Easier-to-grasp / less-occluded objects FIRST.
- Collision-aware:  don't queue a task whose path would be blocked by an object that an earlier task hasn't moved yet.
- Earlier tasks change the scene for later ones --- assume the result of each task before planning the next.
Holding constraint:
- holding.is_holding == TRUE  -> only (2) or (3).  Cannot pick anything new besides continuing with the held tool.
- holding.is_holding == FALSE -> only (1) or (3).
Termination:
- Do not repeat tasks the image shows are already done.
- Empty tasks array = goal satisfied (or nothing executable remains).
Example output:
{
"scene":      "two red apples on the left side (one near the robot, one farther back), a pepsi can at center, a white
basket at right with a pink basket just behind it",
"holding":  {"is_holding":  false, "object":  null, "note":  ""},
"status":    "neither object placed yet",
"tasks":      ["pick up the pepsi can and put it in the pink box",
"pick up the apple and put it on the plate"]
}
Return ONLY the JSON --- no prose, no markdown fences.
```

Multi-view Roles and Plan Selector

```
You are given multiple camera views of the SAME scene captured at the same instant from different angles.
Views provided (in order): [{view_id_1}, {view_id_2}]
Scenario: "{scenario}"
Your job has FOUR parts. Do them in order --- earlier parts ground later parts. You do NOT need to output any (x, y)
point --- a downstream stage will localize each subtask's point on the chosen best_view. Your job is to reason about
ROLES, the PLAN, and where each subtask's point should geometrically lie (free text).
### PART 0 --- DESCRIBE EACH VIEW first (forces grounded reasoning):
For EVERY view_id, output 1--2 short sentences listing what is actually visible in that image AND what is occluded /
cropped. Mention: which objects you can see, what is held by the robot hand/arm, what the robot arm/hand is covering,
what is cropped at the frame edge, and what is in the background vs. foreground.
Be concrete --- name the objects (kettle, dripper, pot, sponge, etc.). If you can't tell what something is, say so.
Do NOT pretend to see something you cannot see --- if the robot arm covers most of an object, say 'kettle body mostly
hidden behind robot arm'.
This description is the EVIDENCE you must use for PART A visibility --- if your description says 'kettle hidden behind
arm' you cannot then mark tool.visibility=true for that view.
### PART A --- Identify the roles in the scenario:
- TOOL: a handheld external object used to act on another object (broom, kettle, knife). NEVER robot hand / arm /
gripper. Set need_tool=false if the scenario does not require an external tool.
- TARGET: the main manipulated item --- what TOOL acts on, or what is grasped directly if no tool.
- DESTINATION: where the action ends up (e.g., pot for pouring into, trashcan for discarding). null if the scenario
does not imply one.
For each role and each view, mark per-view visibility STRICTLY:
visible (true) = at LEAST ~70% of the object's body silhouette is unobscured AND nothing important (handle, spout, tip,
opening) is covered by the ROBOT ARM, ROBOT HAND, gripper, or any other object, AND the object is NOT cropped at the
frame edge. You can identify the object's full shape without guessing.
occluded (false) = ANY of the above fails --- partial occlusion by the robot arm/hand counts as occluded, distant + tiny
silhouette counts as occluded, edge-cropped counts as occluded.
BE CONSERVATIVE. When in doubt, mark as false.
### PART B --- Build a PLAN with EXACTLY 4 entries (fixed structure):
step 0-0    = GRASP            (grasp the TOOL handle / TARGET grasp area)
step 0-1 = pre-action reference of the MOVING OBJECT --- type varies:  # Another localization for grasp object
'FUNCTIONAL_TIP'   if need_tool=true (tool's functional tip)
'TARGET_BODY_REF'  if need_tool=false (target body reference) # which will be used for pick and place location
step 1    = APPLY_ACTION    (only when need_tool=true)
The contact pose where the TOOL acts on the TARGET (sweep contact, pour pose, cut entry, press, etc.). For view
selection, identify WHERE on the target this contact happens so you can pick the view that shows that region clearly.
If need_tool=false, OMIT step 1 from the plan (just steps 0, 0-1, 2). WAYPOINT-type intermediate poses are NOT decided
here --- the downstream subtask grounding handles waypoints with collision-aware reasoning.
step 2    = RELEASE  or  HOLD          (pick whichever fits)
For EACH entry output:
- label : 1-line natural description of what this subtask does.
- moving_ref {kind, desc} : the reference point's semantic.
step 0   :  kind in {'tool_handle', 'target_grasp_area'}
0-1/1/2  :  kind in {'tool_functional_tip', 'target_body_reference'}
desc     :  short phrase identifying that part of the moving object (e.g., 'kettle handle', 'broom bristle tip').
- geometric_meaning : free-text 1-2 sentences describing WHERE on the scene that subtask's point should land - relative
to scene geometry, object parts, colors, or positional relationships to other objects. DO NOT use camera-view-relative
terms (left/right/upper/lower of the image) --- they flip between views.
### PART C --- Pick a SINGLE best_view for the whole plan:
All 4 subtasks share ONE best_view (downstream grounding runs on this single image). Choose the view_id where,
holistically, the TOOL (if any) and TARGET (and DESTINATION when present) are most clearly visible and well-separated
from occluders --- so a downstream point-grounding model can place accurate (x, y) for every subtask on that one image.
HARD RULE: in your own PART A visibility map, the chosen view MUST have tool.visibility[view] == true (if need_tool) AND
target.visibility[view] == true. If NO view satisfies both, pick the least-occluded one and state the limitation in
best_view_reason.
### PART D --- RECIPE selection: # tool use case
Available recipe names: [{recipe_1}, {recipe_2}]. If exactly ONE atomic skill clearly fits the scenario on the TARGET
(e.g., opening a drawer, pouring), set "recipe" to its EXACT name. If this is a plain pick-and-place or none fits, set
"recipe": null.
Output: ONLY a single JSON object (no markdown fences, no preamble). Use exactly this schema:
{
"view_descriptions": {"<view_id>": "<1-2 sentence description of what is visible / occluded / cropped in this view>",
...},
"need_tool": <bool>,
"tool":                {"name": "<short>", "visibility": {"<view_id>": <bool>, ...}},
"target":            {"name": "<short>", "visibility": {"<view_id>": <bool>, ...}},
"destination": {"name": "<short>", "visibility": {"<view_id>": <bool>, ...}} or null,
"best_view": "<view_id>",
"best_view_reason": "<short>",
"recipe": "<one of the names>" or null,
"plan": [
{"step": 0-0,    "type": "GRASP",          "label": "<short>", "moving_ref": {"kind": "tool_handle" |
"target_grasp_area", "desc": "<short>"}, "geometric_meaning": "<1-2 sentences>"},
{"step": 0-1, "type": "FUNCTIONAL_TIP" | "TARGET_BODY_REF", "label": "<short>", "moving_ref": {"kind":
"tool_functional_tip" | "target_body_reference", "desc": "<short>"}, "geometric_meaning": "<1-2 sentences>"},
{"step": 1,    "type": "WAYPOINT" | "APPLY_ACTION", "label": "<short>", "moving_ref": {"kind":
"tool_functional_tip" | "target_body_reference", "desc": "<short>"}, "geometric_meaning": "<1-2 sentences>"},
{"step": 2,    "type": "RELEASE" | "HOLD",      "label": "<short>", "moving_ref": {"kind": "tool_functional_tip" |
"target_body_reference", "desc": "<short>"}, "geometric_meaning": "<1-2 sentences>"}
]}
```

Point Localization from a Fixed Plan [Reference View]

```
Scenario: "{scenario}"
Coordinates: normalized [y, x] in [0, 1000]. [0, 0] = top-left, [1000, 1000] = bottom-right.
Roles (already identified):
- TOOL: '{tool_name}'
- TARGET: '{target_name}'
- DESTINATION: '{dest_name}'
A multi-view selector has already produced the 4-step plan below (reference points + geometric meaning). Your ONLY job
is to locate each step's reference point as an [y, x] on THIS image using the geometric descriptions provided. Do NOT
change the step types or labels. Output exactly one entry per step in the SAME order.
Plan:
- step 0 type=GRASP label='{label}'
reference: {kind} ({desc})
where: {geometric_meaning}
Also output 'functional_tip' [y, x] = the acting end of the held object (broom bristles, knife edge, kettle spout). For
simple pick-and-place objects (mug, block, bottle), functional_tip = the obj's body center.
- step {step} type={TYPE} label='{label}'
reference: {kind} ({desc})
where: {geometric_meaning}
Constraints:
- The point MUST lie on the reference part described above; do NOT pick the handle when 'functional_tip' is the
reference, etc.
- For WAYPOINT entries: the point is a TRANSIT point the moving object must pass through to AVOID COLLISIONS along the
path - it must be distinct from the step 2 (RELEASE/HOLD) point. Use the geometric_meaning to place it where geometry
forces a detour around obstacles.
- Avoid extreme edges, heavy occlusion, or background pixels.
- Never output null. Pick your best estimate even when the reference is partially occluded.
Output format: ONLY a JSON array (no markdown fences, no preamble). Schema per entry: {"step": <as-given>, "type":
<as-given>, "point": [y, x], "functional_tip": [y, x] (GRASP only)}
```

Per-view Point Localization for Triangulation

```
Scenario: "{scenario}"
Coordinates: normalized [y, x] in [0, 1000]. [0, 0] = top-left, [1000, 1000] = bottom-right.
- MOVING OBJECT: '{moving_label}'
- TARGET: '{target_label}'
- DESTINATION: '{dest_label}'
You are looking at camera view '{serial}'. The subtasks below were already decided. For each subtask, locate the SAME
semantic point on THIS image (the points across views are triangulated to 3D, so consistent localization is critical).
Subtasks:
- step 1 type=GRASP label='{label}' (also locate functional_tip)
plan: {plan}
note: GRASP point must lie ON the affordance region of the grasped object (handle if any, else a natural graspable
area).
- step {step} type={TYPE} label='{label}'
plan: {plan}
If a subtask's point is OUT OF FRAME or OCCLUDED in this view, set its point to null.
Output format: ONLY a JSON array (no markdown fences, no preamble). One entry per subtask, in the SAME order/step as
above.
{"step": <int>, "point": [y, x] or null}
GRASP subtask additionally has 'functional_tip': [y, x]
Example:
[
{"step": 1, "type": "GRASP", "point": [320, 540], "functional_tip": [410, 540]},
{"step": 2, "type": "WAYPOINT", "point": [380, 500]},
{"step": 3, "type": "RELEASE", "point": [600, 720]}
]
```

Ray Voting - Choosing candidates from other views

```
GOAL (scenario):  "{scenario}"
Current subtask:
step/type :  {type}
label     :  "{subtask_label}"
plan      :  "{subtask_plan}"
Moving object:  '{moving_label}'.
Target:               '{target_label}'.
Destination:      '{dest_label}'.
PICK CRITERION -- depends on the subtask type (ONE of the following is inserted here for the current subtask's type):
PICK CRITERION (WAYPOINT -- collision-aware waypoint):
Candidate = an INTERMEDIATE waypoint for '{moving_label}' on the path toward '{dest_label}'.  Prefer candidates that
let the gripper AVOID obstacles (lift above clutter, route around tall items).
**SAFETY MARGIN (critical)**:  STRONGLY prefer candidates with GENEROUS clearance (~10% of image extent) from every
obstacle / container wall.  REJECT 'barely clear' / edge-skimming / narrow-gap candidates.  Between two clear options,
pick the one FURTHER from the nearest obstacle (or higher above the scene).
PICK CRITERION (RELEASE -- final pose at DESTINATION):
Candidate = the FINAL POSE of '{moving_label}' AT '{dest_label}'.  Use the plan text + destination geometry.  Must be
AT '{dest_label}'.        [RELEASE and HOLD use the same criterion]
(GRASP: no explicit pick criterion is inserted -- the on-object grasp affordance hint constrains it to a single best
point per view.)
[GRASP only] REASONING STEP (do this BEFORE choosing):  # for refinement, CoT
Candidate #{idx} is the position chosen so far (the current estimate to refine).
1) Judge whether that current position is actually ON '{moving_label}' that this subtask must grasp.
2) If it is off, decide WHICH DIRECTION (relative to '{moving_label}' in this image) the correct region lies, then pick
the candidate number(s) toward it.  If already correct, keep it.
[GRASP only] For GRASP the model writes the reasoning above first, then outputs the JSON array on a NEW FINAL line
(reason-then-answer).
You are given ONE image from camera serial={serial}.  It shows {K} numbered candidate points (1 to {K}).  These
candidates are different 3D hypotheses (perturbed along the camera ray + small lateral uv noise) projected into this
view; exactly one best represents the true 3D location that satisfies the plan above.
Choose UP TO {top_m} candidate numbers (1 to {K}) that best match the plan (with the goal and label as secondary cues).
If very confident, return just 1; otherwise up to {top_m}.
Output ONLY a JSON array of 1 to {top_m} integer indices (best first).  No markdown fences, no extra text.
Example format:  [4, 3, 5]
```

Grasp Affordance

```
You are given a camera view of a scene.  Manipulation task:  "{scenario}"
Grasp step from the task plan:  "{grasp_hint}".
Use this to locate the contact region (it describes where/how to grip).
A multi-finger robot hand needs to grasp '{grasp_object}'.
Mark the CONTACT REGION on '{grasp_object}' in THIS image --- the whole surface patch the hand's fingers AND palm wrap
around and touch for a stable grasp (handle / grip body, NOT the functional tip).
This is an AREA, not a point.  The bbox MUST enclose the full hand footprint (the span closed fingers cover a hand
width), across the object's thickness --- not a tiny box, not a single corner.  If '{grasp_object}' is elongated (can /
handle / bottle), cover the length the hand grips, not just one end.  Localize precisely on THIS image's pixels.
Return ONLY this JSON (no comments), affordance_point = region center:
{
"affordance_description":  "<brief>",
"affordance_point":  [y, x],
"affordance_bbox":  [y1, x1, y2, x2]
}
All coordinates normalized [0-1000].
```