

PartInstruct: Part-level Instruction Following for Fine-grained Robot Manipulation

Yifan Yin^{*1} Zhengtao Han^{*2} Shivam Arya¹ Shuhang Xu¹ Jianxin Wang¹ Jiawei Peng¹

Angtian Wang¹ Alan Yuille¹ Tianmin Shu¹

¹Johns Hopkins University

²ShanghaiTech University

<https://partinstruct.github.io>

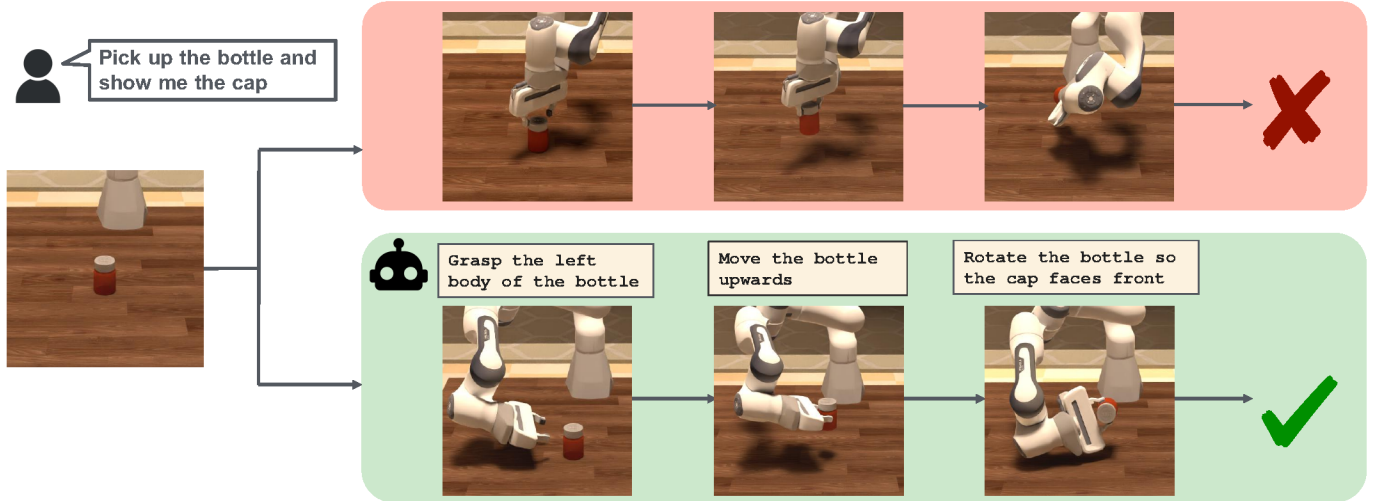


Figure 1: An example fine-grained robot manipulation task in **PartInstruct**. To successfully perform the task described in the instruction (e.g., showing the cap without occluding it), the robot needs to reason about what object parts are relevant, ground the parts to its 3D visual perception, and plan for a sequence of part-level manipulation skills (e.g., the bottom sequence). Native object manipulation without a detailed understanding of object parts will fail to achieve the intended goal (e.g., the top sequence). These tasks thus pose challenges for robust 3D vision and part-level grounding and reasoning. We show more examples on the project website.

Abstract—Fine-grained robot manipulation, such as lifting and rotating a bottle to display the label on the cap, requires robust reasoning about object parts and their relationships with intended tasks. Despite recent advances in training general-purpose robot manipulation policies guided by language instructions, there is a notable lack of large-scale datasets for fine-grained manipulation tasks with part-level instructions and diverse 3D object instances annotated with part-level labels. In this work, we introduce PartInstruct, the first large-scale benchmark for both training and evaluating fine-grained robot manipulation models using part-level instructions. PartInstruct comprises 513 object instances across 14 categories, each annotated with part-level information, and 1302 fine-grained manipulation tasks organized into 16 task classes. Our training set consists of over 10,000 expert demonstrations synthesized in a 3D simulator, where each demonstration is paired with a high-level task instruction, a chain of base part-based skill instructions, and ground-truth 3D information about the object and its parts. Additionally, we designed a comprehensive test suite to evaluate the generalizability of learned policies across new states, objects, and tasks. We evaluated several state-of-the-art robot manipulation approaches including end-to-end vision-language policy learning and bi-level planning models for robot manipulation on our

benchmark. The experimental results reveal that current models struggle to robustly ground part concepts and predict actions in 3D space, and face challenges when manipulating object parts in long-horizon tasks.

I. INTRODUCTION

There has been an increasing interest in training general-purpose vision-language policies for robot manipulation guided by language instructions [24, 17, 50, 16, 32, 51], particularly with the recent advances in large generative models [3, 41]. These models represent a promising type of method for solving general robot manipulation problems, as they have the potential to follow natural language instructions to complete *any* described task. Prior works on language-guided robot manipulation have been mainly focused on high-level manipulation tasks involving simple objects (such as rearranging blocks). However, in the real world, robots often need to perform fine-grained manipulation of diverse everyday objects, in which the robots need to not only identify the target object but also understand and interact with specific parts of that object to perform the intended task as instructed. This involves reasoning about the relationship between the

^{*}Equal contribution. Zhengtao Han completed this work during an internship at JHU.

Table I: Comparison of PartInstruct with existing tabletop robot manipulation benchmarks based on: the number of distinctive part-level instructions, the number of part labels, the number of fine-grained part-level tasks, availability of training demonstrations, and whether these demonstrations include part-level annotations such as 2D and 3D segmentation masks.

Name	# Part Instruct	# Part Labels	# Part-level Tasks	Demo	2D Part Mask	3D Part Mask
CALVIN	6	-	6	✓	✗	✗
RLbench	136	-	64	✗	✗	✗
VIMAbench	-	-	-	✓	✗	✗
LoHoRavens	-	-	-	✓	✗	✗
ManiSkill (SAPIEN)	-	14,068	-	✓	✗	✗
PartManip	-	8,489	1,432	✗	✗	✗
Open6DOR	2,447	-	1,419	✗	✗	✗
PartInstruct (ours)	4,043	4,653	1,302	✓	✓	✓

part and the task, and grounding that understanding into precise motion planning. For instance, to successfully perform the manipulation task defined in the instruction as shown in Figure 1, the robot needs to identify crucial parts of the object relevant to the task (e.g., the label on the cap of the bottle) and reason about a chain of base part-based skills that would lead to the desired goal state implied by the instruction, which is to display the label clearly to the human user without occlusion.

Despite the importance of part-level perception and reasoning for robot manipulation, existing robot manipulation benchmarks on instruction following lack comprehensive integration of part-level semantics in both task instructions and object ground-truth annotations [e.g., 16, 17, 24, 46, 50]. These benchmarks focus on object instance-level manipulation tasks but do not include fine-grained, part-level manipulation tasks like the example in Figure 1. There have been recent benchmarks that evaluate fine-grained, part-level manipulation tasks, but they either lack language instructions [e.g., 27, 9] or do not provide training data for policy learning [e.g., 7].

To address these gaps, we introduce *PartInstruct*, the first large-scale fine-grained robot manipulation benchmark for vision-language policy learning that incorporates part-level semantics. Our core idea is to develop part-level skills that enable robots to perform complex, fine-grained object manipulation tasks, including those requiring long-horizon motion plans. We developed a robot manipulation simulator for part-level instruction following tasks, *PartGym*. Built upon the PartGym simulator, our PartInstruct benchmark supports both training and evaluation models on part-level manipulation tasks. Specifically, we provide a large set of 3D assets of everyday objects richly annotated with part-level information. Using these object assets and detailed annotations, we created a large-scale training dataset of expert demonstrations. Each demonstration is paired with a task instruction as well as a chain of base skill instructions (such as touching or grasping an object part) necessary for performing the overall task. This dataset allows training models for both long-horizon manipulations guided by task instructions for long-horizon planning and base manipulation skills guided by skill instructions. Additionally, we developed a comprehensive evaluation suite consisting of five test sets, each corresponding to a different

type of generalization test. Together, these tests assess how well a learned policy performs in unseen scenarios, including new states, objects, and tasks. We compare PartInstruct with several existing table-top manipulation benchmarks in Table I.

We evaluated multiple state-of-the-art vision-language policy learning methods designed for language-guided robot manipulation. We also combined recent learning-based low-level action policy planning models and VLM-based high-level task planners to create strong bi-level planning baselines for fine-grained manipulation tasks, which explicitly reasons object parts relevant to a task and how to interact with them to achieve the final goal. Our experimental results demonstrate that state-of-the-art methods still struggle with complex fine-grained manipulation tasks. We also show that visual representations based on robust part-level 3D perception can significantly improve model performance. These results help reveal the fundamental building blocks for fine-grained task manipulation.

In summary, our main contribution includes (1) the first part-level instruction following benchmark for both training and evaluating fine-grained robot manipulation models’ capacity for part-level grounding, reasoning, and planning; (2) a large training dataset with diverse assets and detailed annotations; (3) a comprehensive evaluation of state-of-the-art vision-language policy learning and bi-level planning baselines, revealing limitations of current robot manipulation models.

II. RELATED WORK

A. Instruction Following Benchmarks for Table-Top Robot Manipulation

Early benchmarks in robot manipulation primarily concentrated on object-level and object-scene interactions without delving into the manipulation of specific object parts. Notable examples include CALVIN [24], RLbench [16], VIMAbench [17], and LoHoRavens [50]. These benchmarks typically involve tasks such as object placement, scene arrangement, and basic interaction with objects in their entirety. For instance, CALVIN incorporates spatial semantics but lacks explicit part-level semantics, treating components like a “door handle” as standalone objects rather than parts of a larger entity. This limitation restricts the granularity of instructions and the complexity of manipulation tasks that can be evaluated.

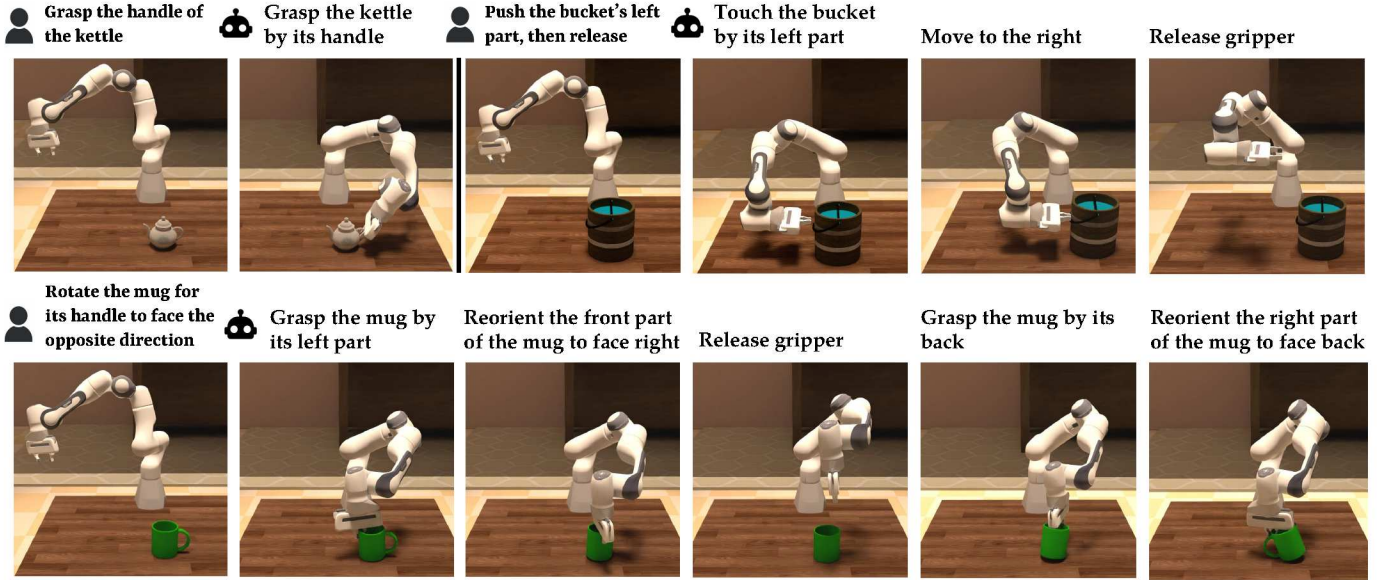


Figure 2: Example tasks and expert demonstrations in the dataset. Each task is defined by a task instruction. Each demonstration is annotated with a chain of base skills and the corresponding skill instructions (the instructions following the task instructions). Specifically, in this figure, the demonstrations for the three tasks have 1, 3, and 5 annotated skill instructions respectively.

To bridge this gap, several benchmarks have introduced object part manipulation, including ManiSkill [27], PartManip [9], and Open6DOR [7]. These benchmarks introduce tasks that require finer control and understanding of object components. ManiSkill extends manipulation tasks to include interactions with articulated objects, whereas PartManip focuses explicitly on part-level manipulation within a structured environment. Notably, Open6DOR is the only benchmark identified that incorporates spatial semantic part-level instructions. However, it does not support policy learning; instead, it outputs final goal positions and orientations, relying on an oracle planner to plan for intermediate actions.

There have been recent approaches supporting part-level manipulation such as Composable Part-based Manipulation (CPM) [22], RoboPoint [47], and SAGE [10]. RoboPoint leverages point-based representations to facilitate precise part interactions, but more focus on spatial relationships. SAGE employs semantic grasping techniques to enhance manipulation accuracy mainly for articulated objects. These methods underscore the importance of integrating detailed object part information to achieve more sophisticated manipulation.

B. Vision-Language Policies for Robot Manipulation

The integration of vision and language in robot manipulation has given rise to various policy frameworks designed to interpret and execute instructions. Generalist approaches such as RT-1 [3], OpenVLA [19], and Octo [41] strive to create versatile policies capable of handling a wide range of tasks by leveraging large-scale vision-language models. These models are pretrained on large-scale datasets, enabling them to leverage extensive vision-language knowledge to interpret natural language instructions and translate them into actionable manipulation strategies. Key-pose based manipulation methods, such as PerAct [37], Act3D [11], and RVT series [12]

[13], focus on identifying and executing key poses that align with the desired manipulation objectives. These approaches typically involve detecting pivotal positions or configurations that the robot must achieve to successfully complete a task, thereby simplifying the policy learning process. Additionally, frameworks like DP [5] and DP3 [49] formulate visuomotor robot policies using Denoising Diffusion Probabilistic Models (DDPM), enabling these policies to capture multimodal action distributions and generate high-dimensional action sequences. By leveraging the strengths of generative models, these methods can predict expressive and flexible robot actions.

C. Robot Planning with LLMs and VLMs.

The integration of Large Language Models (LLMs) and Vision Language Models (VLMs) into embodied planning has revolutionized the capabilities of robotic systems by enhancing their understanding, reasoning, and execution of complex tasks. For instance, TaPA [44] and LLM-Planner [38] focus on leveraging the contextual and generative capabilities of LLMs to decompose high-level instructions into actionable sub-tasks. SayCan [1] presents a framework that anchors linguistic instructions in the physical affordances of objects. By aligning language understanding with the robot’s physical capabilities, it ensures that generated actions are both feasible and contextually appropriate. These approaches enable robots to interpret complex, multi-step instructions by breaking them down into manageable components, thereby facilitating more coherent and structured action planning.

III. PARTINSTRUCT BENCHMARK

A. Problem Setup

We define an object part as a geometric sub-component of an object that is either functionally manipulable (e.g., handle)

Table II: Example task instructions and goal states. Row A corresponds to the task illustrated in Figure 1, while rows B to D correspond to the three tasks shown in Figure 2.

	Task Instruction	Goal States
A	Rotate the <i>part</i> of the <i>object</i> to face <i>direction</i> while lifting it	GRASPING(obj), FACING(part, dir), AT_POSITION(obj, POS_INIT_OBJ+VEC(UP))
B	Grasp the <i>object</i> by the <i>part</i>	GRASPING(gripper, part), ON(obj, table)
C	Move the <i>object</i> to <i>direction</i> by pushing it at the <i>part</i> , then free it	Phase1: TOUCHING(part), AT_POSITION(obj, POS_INIT_OBJ+VEC(dir)) Phase2: GRIPPER_OPEN, MIN_DISTANCE(gripper, obj)
D	Rotate the <i>part</i> of the <i>object</i> to face the opposite direction	FACING(part, ~DIR_INIT(part)), ON(obj, table)

Table III: Definitions of base skills.

Skill	Description
grasp_obj (obj, part)	Robot grasps <i>obj</i> at <i>part</i> .
move_gripper (dir, dis=UNIT, grasping=false)	Robot moves gripper along <i>dir</i> <i>dis</i> .
rotate_obj (obj, part, dir)	Robot rotates <i>obj</i> , such that <i>part</i> is facing <i>dir</i> .
touch_obj (obj, part)	Robot touches <i>obj</i> at <i>part</i> .
release_gripper (obj)	Robot releases the gripper and moves away from <i>obj</i> .

or spatially distinct (e.g., front). As shown in Figure 1, a natural language instruction I_{task} describes a part-level instruction following task if it requires that a robot perform a fine-grained manipulation where the robot must interact with a list of object parts in a certain manner to achieve the intended goal g . Critically, the relevant object parts and how the robot needs to interact with them are often not explicitly described in the instructions. Thus the robot must learn to reason about relevant parts and plan how to manipulate them to perform the task successfully. To define g , we first establish a set of goal predicates that specify the states of the object, its parts, the robot’s end effector, and their relationships. For example, **ON**(obj, part, surface) represents physical contact between an object part and a given surface; **FACING**(obj, part, dir) indicates the orientation of an object part from a third-person perspective; and **GRASPING**(obj, part) denotes a “grasp” interaction between the object part and the robot’s end effector. Given these goal predicates, each task goal is defined by a set of goal predicates. Examples of tasks are presented in Table II. For instance, in the task illustrated in Figure 1, the goal is represented by the predicate set {**GRASPING**(bottle, ~cap), **FACING**(bottle, cap, front), **AT_POSITION**(bottle, INIT_POS+VEC(UP))}, where ~cap is any part other than the cap. Note that some tasks consist of multiple phases, where the next phase can only begin after completing the previous one, as the order of interactions is crucial for these tasks. For full task definitions, refer to A3 in Appendix.

To develop an embodied agent capable of executing tasks defined by g , we hypothesize that it would be beneficial to start with a set of base skills that can be combined to handle a wide range of fine-grained manipulation tasks. In particular, we consider five types of base skills: **grasp_part**, **touch_part**, **rotate_obj**, **move_gripper**, and **release_gripper**. As detailed in Table X and Appendix A2, each skill is

parameterized by (1) the object part it interacts with and the type of interaction (e.g., touching or grasping), (2) the degree of rotation required for the part, and (3) the distance and direction in which the gripper or object should be moved. This information is summarized in a skill instruction I_{skill} associated with that skill. As illustrated in Figure 2, a task given by an overall task instruction can be decomposed into a sequence of base skill executions, each described by a skill instruction. For example, the second task shown in Figure 2, “Push the bucket’s left part, then release”, involves three skill executions. To “push” the bucket’s left part, the robot must first touch the left side of the bucket by executing **touch_part**(bucket, left), then move the end effector to the right via **move_gripper**(right). Following the “push” action, the robot executes **release_gripper**() to complete the task. We hypothesize that structuring fine-grained manipulation tasks into sequences of base skills can facilitate the training of hierarchical planning models to compose complex plans with base skills for long-horizon tasks that a end-to-end vision-language policy would struggle with.

B. Simulation Environment

To train and evaluate language-guided part-level manipulation models, we introduce PartGym, a realistic robot simulator for fine-grained manipulation tasks requiring part-level understanding. PartGym provides (1) rich 3D assets of everyday objects, (2) part-level 3D ground-truth annotations, and (3) a large task set for fine-grained robot manipulation with natural language instructions. We used Pybullet [6] as the backbone physics engine to simulate the physical interactions between a robot arm and different objects and their parts. Specifically, the environment includes a 7-DoF Franka Emika Panda robot with a two-finger parallel gripper.

Observations. As shown in Figure 3, we provide multi-modal observations for a robot, including RGB images, depth maps, and point clouds. Additionally, we provide object and part annotations. Lastly, proprioception robot states like joint states and end-effector poses are also available as part of the observations.

Action Space. The Panda robot takes a 7D action vector at each step. The first 6 dimensions represent the end-effector’s Cartesian pose, parameterized by a 3D coordinate as well as the roll, pitch, and yaw angles. The final dimension controls the gripper’s position.

We provide more details about PartGym in Appendix B.

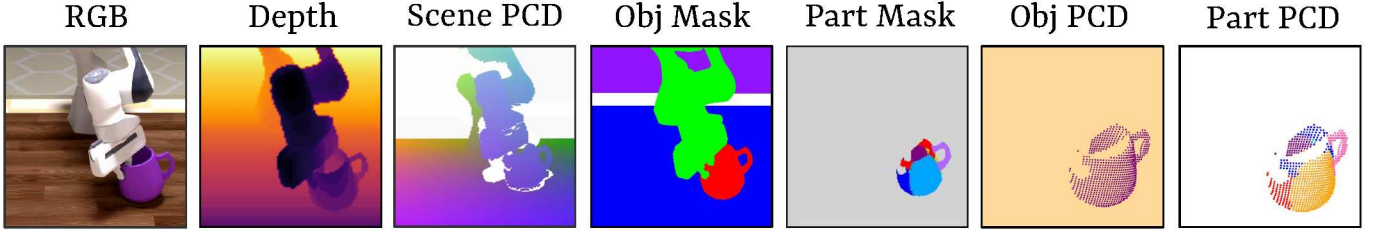


Figure 3: PartGym supports multimodal observations, including RGB images, depth maps, and scene point clouds (PCDs). It also provides object and part annotations, including object segmentations, 2D part segmentation for each object part (part mask), 3D object instance segmentation (obj PCDs), and 3D part segmentations on point clouds (part PCDs) for each object.

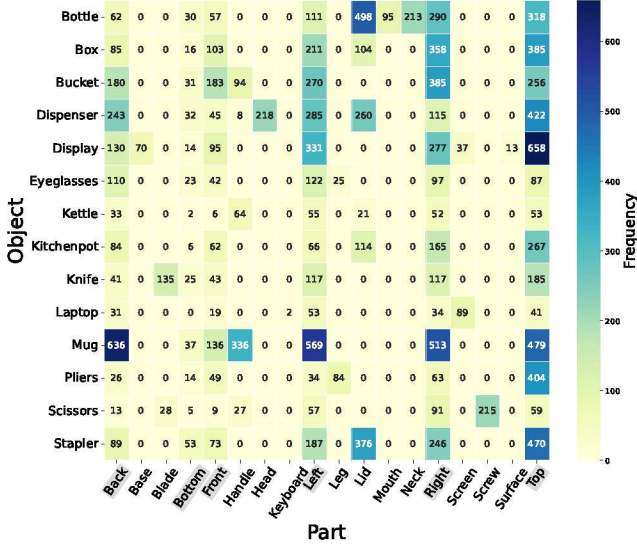


Figure 4: Annotated parts grouped by object categories. The horizontal axis stands for different part names and the vertical axis gives different object categories. The value in the heatmap indicates the frequency of each part for an object category in PartInstruct. A darker color shows a higher frequency. Spatial part names are highlighted in light gray to distinguish them from semantic part names.

C. Dataset

1) *PartInstruct Dataset*: Built upon the PartNet Mobility dataset [45, 26, 4], PartInstruct contains 14 categories of table-top everyday objects annotated with different part labels. In total, there are 513 object instances and 4,653 part labels. Figure 5 shows the object instance distribution across object categories. We also show the distribution of annotated parts for each object category in Figure 4. Figure 6 illustrates the visual diversity of objects and parts in PartInstruct. Each part of an object is unique in terms of its shape, size, texture, and position on the object. Objects of the same class also have different part compositions. For example, there are 7 types of part compositions for bottles, including “(body, closure, neck)”, “(body, handle, lid, neck)”, “(body, handle, mouth)”. Leveraging the richly annotated objects and parts, we procedurally generate a large collection of demonstrations

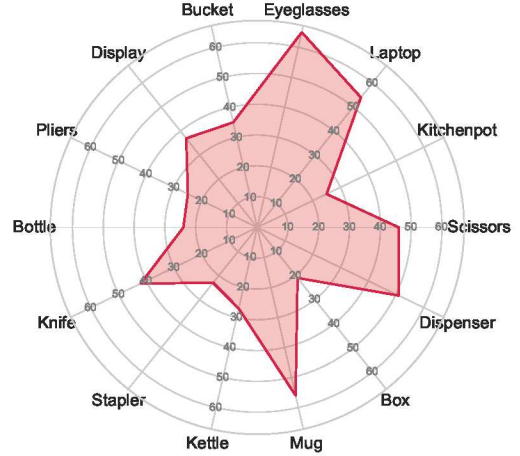


Figure 5: Number of object instances in each object category.

for vision-language imitation learning.

PartInstruct includes 10,000 demonstrations for training and over 1,800 annotated episodes for evaluation. See Figure 2 for several example episodes in PartInstruct. Each episode contains an observation set with different modalities, an expert action trajectory, a natural language description of the overall task, referred to as the **task instruction** I_{task} , as well as a sequence of **skill instructions** I_{skill} that specify the part-level manipulation **subgoals** sg required to complete the task. Each skill instruction contains zero or one object part the robot is manipulating with. It is important to note that skill instructions are provided only during model training. For evaluation, models receive only the overall task instruction as the language input.

2) *Task Categories*: PartInstruct has 16 task categories, including 10 seen categories for training, and 6 unseen categories for testing. Each category is defined by tasks that require the robot to execute a specific combination or sequence of part-level interactions. Some categories require the agent to physically interact with a specific part of the object. For example, “Hold [part] the object and shift it in [direction].” For such tasks, the agent must ground the part mentioned in the task instruction to specific visual representations and predict the actions needed to directly manipulate that part. Other task categories require the agent to change the state of a part. For example, “Rotate the object such that [part] is



Figure 6: Representative object assets from PartInstruct.

facing **[direction]**.” To perform these tasks, the model needs not only to know the location of the part but also to infer its final state. The agent must manipulate some part of the object to achieve that state, even when the part being directly manipulated differs from the part mentioned in the instruction.

In the 5 test task categories, we have also designed more challenging part-level manipulation tasks. One focus is on long-horizon tasks that require the manipulation of multiple parts in sequence. For instance, “Push the object toward **[direction]** while touching **[part]**, lift the object by holding **[part]**, then rotate **[part]** to face **[direction]**.” Another focus is on tasks that demand more complex reasoning about parts, the environment, and their spatial relationships. For example, consider the task, “Rotate **[part]** of the object on the table so that it points to the opposite direction.” Here, instead of explicitly naming the final state (e.g., a specific direction), the task requires the robot to have additional knowledge about the current direction of a certain part, identify its opposite direction, and manipulate the object so that the part points in that direction.

3) *Demonstration Generation*: Each demonstration is a sequential execution of oracle high-level plans of base skills defined in Table X. To generate the trajectories in the demonstrations, we detect grasping point using [2] and then leverage a sampling-based motion planner, BiRRT [20] to generate the motion plan for each base skill.

To generate the task instruction for each task, we first create template-based instructions (Appendix A3). To enrich the language diversity, we prompt GPT-4o with the template-based instruction, task definition, and object metadata to paraphrase the task instruction. This yields between 3 – 8 natural-language variants per template, greatly increasing the language diversity of the dataset. For each base skill, we follow the template in Table X to generate skill instructions.

4) *Evaluation Protocol*: As defined in Section III-C, each part-level skill has a binary success criterion. A completion of the entire task means the agent manages to complete every single skills defined in the skill chain. To systematically evaluate the performance of the learned policy, we designed a five-level evaluation protocol (see Table IV). Each test set evaluates a policy in one type of generalization condition. Specifically, they focus on generalizability over object initial states (OS),

Table IV: Summary of the five test sets and the type of generalization each one addresses.

Test Set	Type of Generalization
Test 1 (OS)	Novel object positions and rotations
Test 2 (OI)	Novel object instances within the same category
Test 3 (TP)	Novel part combinations within the same task categories
Test 4 (TC)	Novel part-level manipulation task categories
Test 5 (OC)	Novel object categories

novel object instances (OI), novel part combinations in the same task type (TP), novel task categories (TC), and novel object categories (OC). Detailed visualization can be viewed in Appendix B.

IV. EXPERIMENTS

To achieve general-purpose robot manipulation, there have been two common types of approaches: (1) end-to-end policy learning that directly maps observation and instruction to actions (e.g., [48, 8, 23, 32, 41, 11, 13, 5, 49]) and (2) bi-level planning that first generates high-level plans (typically subgoals), then compute and execute the low-level action plans to achieve the subgoals [44, 38, 1, 10, 43]. In our benchmark, we evaluate both types of approaches.

A. End-to-End Policy Learning

1) *Baselines*: We evaluate the following state-of-the-art end-to-end robot manipulation policy learning methods:

Octo [41] is a transformer-based generalist robot policy pretrained in diverse large-scale robotic episodes. At each time step, the model outputs an action vector that contains the translation and rotation of the robot end effector, along with one dimension that indicates the gripper open and closed.

Act3D [11] is a 3D feature field transformer for multi-task 6-DoF robotic manipulation. Unlike Octo, it employs a key-frame-based approach to complete tasks. These key pose will then be executed using a motion planner.

RVT2 [13] is a multi-task transformer-based 3D manipulation model. Similar to Act3D, it also applies key-frame based manipulation.

3D Diffuser Actor (3D-DA) [18] trains a policy that is jointly conditioned on a tokenized 3D scene, proprioceptive feedback, and a natural-language instruction. It uses diffusion to generate 3D pose trajectories.

Diffusion Policy (DP) [5] represents a visuomotor policy as a conditional denoising diffusion process in the action space, which allows it to effectively handle multimodal action distributions and high-dimensional action sequences.

3D Diffusion Policy (DP3) [49] combines 3D visual representations with diffusion-based policies, leveraging compact 3D point cloud data for efficient and generalizable visuomotor policy learning.

Note that the original DP and DP3 models do not support language instruction inputs. To fit the setup of PartInstruct,

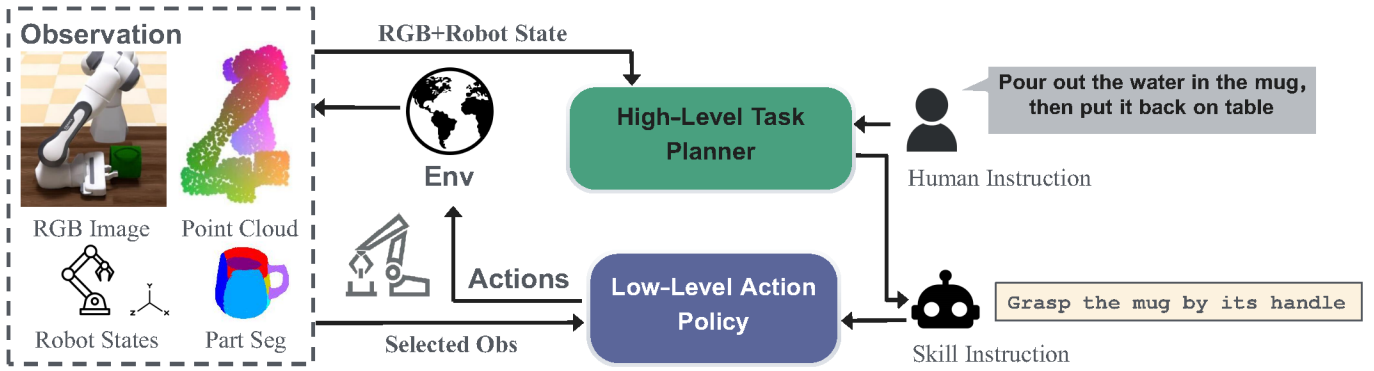


Figure 7: Overview of the bi-level planning framework. The High-Level Task Planner generates a skill instruction as a subgoal for the low-level action policy based on the task instruction and the current observation. Given the subgoal described in the skill instruction, the low-level action policy then generates actions for achieving that subgoal. The high-level task planner updates the skill instruction once every n steps, while the low-level action policy updates the action at every step.

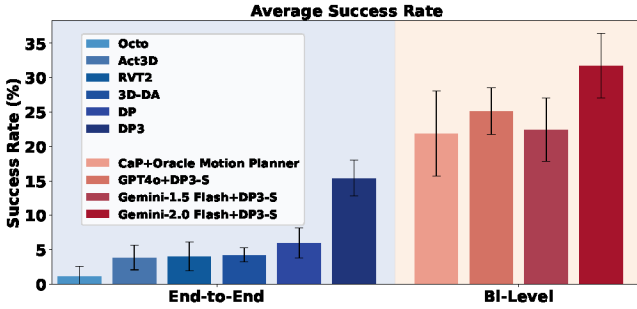


Figure 8: Success Rates of all baselines. The left group represents end-to-end learning policies, while the right group corresponds to bi-level planning models. Error bars denote the standard errors calculated across all evaluation rollouts.

we modify them to incorporate language inputs. Specifically, we use a pre-trained T5 language encoder to get the language embedding [31]. The embedding is then concatenated with other features and used as the observation condition for the denoising diffusion process.

We trained the baselines DP, DP3, Act3D, RVT2, 3D-DA from scratch and fine-tuned the pretrained baseline Octo on our training data. Our hypothesis is that fine-tuning Octo will improve its performance on our benchmark by leveraging its large-scale pretraining on Open X-Embodiment [28]. The implementation details can be found in Appendix D.

2) *Results*: To evaluate each learned policy, we follow the common practice outlined in recent works [17, 5, 49]. Specifically, we select the top two checkpoints for each baseline and conduct approximately 20 rollouts per object class across all test splits, resulting in over 1,000 rollouts per baseline. We report the *Success Rate* (SR, %) for all end-to-end policy baselines in the left part of Figure 8 and in the top block of Table V. The low success rate across all baselines suggests that it remains challenging to train an end-to-end generalist policy for fine-grained object manipulation tasks given part-level instructions. They particularly struggle with long-horizon

tasks (Test 4) and generalizing to unseen object types (Test 5).

B. Bi-level Planning

1) *Baselines*: We hypothesize that it would be easier to train action policies with skill instruction annotations compared to directly training a policy for the whole task. Such low-level action policies can then be combined with a high-level planner that generates skill instructions given a task instruction to solve the manipulation task intended by the user. To evaluate the efficacy of bi-level planning on our benchmark, we extend common bi-level planning frameworks (e.g., [10]) as shown in Figure 7. Specifically, the bi-level planner consists of two modules: (1) a high-level task planner and (2) a low-level action policy. We describe each module below.

High-level Task Planner. We leverage a VLM for high-level task planning. At step t , we prompt the VLM with the task instruction I_{task} to generate the skill instruction for the current step as the subgoal sg_t , i.e., $\pi_{\text{VLM}}(sg_t|o_t, I_{\text{task}})$, where o_t is the observation at step t . We constrain the skill instructions to the space of base skills defined in Section III-C and Appendix A2, which is also specified in the prompt for the VLM. To facilitate decision-making, we also provide additional observations when prompting the VLM, such as RGB images of the workspace, robot states, etc. See Appendix D3 for the detailed prompt. sg_t will be passed to the low-level action policy for execution and will be updated every n step. Here, n is estimated by the typical length of a skill execution in the training set. It is worth noting that we could potentially incorporate an additional VLM to assess the completion of the current skill and trigger updates to the skill instruction. However, based on our study [42, 25] and preliminary experiments, current VLMs are not yet robust enough to reliably estimate this using multi-modal inputs. We evaluate GPT-4o [15], Gemini-1.5 Flash [40] and Gemini-2.0 Flash [40] for the high-level task planner.

Low-level Action Policy. The low-level action policy is a vision-language policy that generates low-level manipulation actions based on a subgoal and the current observation, i.e.,

Table V: Success Rates (%) of baselines across five test sets. Baselines are categorized into end-to-end policy learning and bi-level planning. Standard errors are reported alongside each value. The best-performing results are highlighted in bold.

	Baselines	Test 1 (OS)	Test 2 (OI)	Test 3 (TP)	Test 4 (TC)	Test 5 (OC)	All
End-to-End Learning	Octo	1.82±1.3	0.0	0.91±0.1	0.0	3.33±3.2	1.11±1.5
	Act3D	6.25±1.8	5.68±1.7	4.55±1.6	0.0	2.08±2.1	3.88±1.8
	RVT2	4.55±2.0	4.55±2.0	6.36±2.3	0.91±0.9	3.33±3.3	4.04±2.1
	3D-DA	8.08±2.7	5.05±2.2	4.04±1.9	0.0	3.70±3.6	4.26±1.0
	DP	7.27±1.8	8.64±1.9	8.18±1.8	3.75±2.1	6.67±3.2	5.96±2.2
	DP3	23.18±2.8	23.18±2.8	18.18±2.6	7.73±1.8	6.67±3.2	15.40±2.6
Bi-Level Planning	CaP + Oracle Motion Planner	22.58±4.9	27.93±9.1	25.95±11.0	6.99±12.2	19.38±9.8	21.90±6.2
	GPT4o+DP3-S	33.64±3.2	32.73±3.2	25.91±3.0	10.00±2.0	23.33±5.5	25.12±3.4
	Gemini-1.5 Flash+DP3-S	30.48±4.5	25.45±4.2	27.62±4.4	1.82±1.8	26.67±8.1	22.41±4.6
	Gemini-2.0 Flash+DP3-S	40.58±4.2	34.56±4.1	33.33±4.1	11.90±2.9	38.24±8.3	31.72±4.7

Table VI: Performance of low-level action policies when paired with ground-truth high-level plans.

Baselines	OS	OI	TP	TC	OC	All
Octo	3.64	5.50	5.90	0.00	6.67	4.34
Act3D	0.45	2.80	3.33	0.00	0.00	1.32
3D-DA	7.27	5.45	3.64	0.00	6.67	4.61
RVT2	1.82	3.64	1.82	0.00	3.33	1.91
DP	6.47	11.42	16.53	0.06	6.94	8.28
DP3	19.34	13.61	17.89	0.00	12.08	12.58
DP-S	20.00	16.36	25.45	0.00	6.67	13.70
DP3-S	23.64	29.09	23.64	1.82	26.67	20.97

$\pi(a_t|o_t, sg_t)$, where a_t is the action at step t . We can train such policies using the skill instructions annotated for training demonstrations in our dataset. We can train the end-to-end policy learning models evaluated in Section IV-A on skill instructions to create low-level action policies.

We hypothesize that an explicit visual understanding of object parts can facilitate part-level instruction grounding. It is difficult to visualize all object parts due to occlusion. However, in our tasks, the robot needs to interact with at most one part for the subgoal sg_t defined in each skill instruction, making it possible to give additional vision inputs about the target object part to the low-level action policies. We select the best-performing end-to-end policy learning baselines, DP and DP3, to train the low-level action policies with object part segmentation as part of the input.

For DP, we provide a part segmentation mask as an extra vision input. There have been general-purpose segmentation models like Segment Anything Model 2 (SAM 2) [33]. We adopt the approach of Grounded-SAM-2 [35] to leverage SAM 2 to segment and track object parts. Specifically, given an RGB image and language input, we first utilize a VLM, e.g. Florence-2 [34] to ground the language onto the target part, then prompt SAM 2 to generate segmentation masks and track the object part in real-time. At each step, we add the obtained part segmentation mask as an extra channel on top of the original RGB, make the input a 4-channel image. The image is then encoded using a ResNet18 [14] encoder before feeding into the DP model. We refer to this model as *DP-S*.

For DP3, we use a part point cloud as an additional vision input. Since there has not been a general-purpose object part segmentation model on 3D point cloud [39, 36], we obtain the 3D part segmentation using a lift-to-3D method.

In detail, we first apply the same method in DP to obtain a 2D segmentation mask tracked using SAM2. We then lift the 2D mask into 3D with the depth map using the pinhole camera model and camera intrinsics. To represent a 3D part mask, we append a binary mask channel to the original point cloud observation. This modified point cloud is encoded using an MLP, following the approach described in the original implementation [49]. Additionally, as outlined in the original work, the point cloud was cropped to match the minimum workspace, which includes only the robot arm and the object. We refer to this action policy as *DP3-S*.

We train the low-level action policies using the training demonstrations and the skill instructions annotations, where each demonstration is truncated into clips corresponding to individual skill instructions. The implementation details of bi-level planning baselines can be found in Appendix D3.

Additionally, we evaluate Code-as-Policies (CaP) [21] as an alternative bi-level planning framework. CaP leverages an LLM to compose API calls to generate robot policy code. In our experiment, we define API calls as the skill primitives implemented by the oracle motion planner as described in Section III-C3. We use GPT-4o for the LLM.

2) *Results*: We adopt the same evaluation protocol described in Section IV-A2 for bi-level planning baselines. To evaluate different low-level action policies without considering the effect of high-level task planners, we first pair each low-level action policy with ground-truth skill instructions. As shown in Table VI, *DP3-S* has the highest success rate across all test sets.

Given this result, we then adopt *DP3-S* as the low-level action policy and pair it with different high-level planners to create bi-level planning baselines. The results are reported in the right part of Figure 8 and the bottom block of Table V. We can see from the results that the bi-level planning baselines outperform the end-to-end learning in every test set by a large margin. This demonstrates the effectiveness of training a separate low-level action policy for base skills and using VLM as high-level task planner. Among all high-level planning baselines, Gemini-2.0 Flash paired with DP3-S performs the best. However, bi-level planning still struggles with many tasks, particularly when the tasks require longer chains of base skills (e.g., Test 4). In these longer-horizon tasks, there is a higher chance for the high-level task planner

Table VII: Impact of high-level task planners on bi-level planning models. We pair each high-level task planner with an oracle motion planner to execute the skill instructions.

Baselines	OS	OI	TP	TC	OC	All
Gemini-1.5 Flash	20.41	19.07	19.36	0.15	29.24	17.65
Gemini-2.0 Flash	27.73	25.94	26.75	0.00	32.70	22.62
GPT4o	20.08	18.87	14.87	0.79	22.45	17.94

to make mistakes. Errors from the low-level action policy are also more likely to be accumulated.

C. Ablation Studies

In Section IV-B, we demonstrate that bi-level planning models with low-level action policies informed by part segmentation perform significantly better than state-of-the-art end-to-end policies. To evaluate the effect of each component of the high-level planning models, we conduct the following ablation studies.

1) *Effects of High-level Planners*: To evaluate the effectiveness of different VLMs as high-level planners on the overall task performance, we construct bi-level planners by combining each VLM with an oracle motion planner to perform the skill instructions generated by the VLM. Specifically, we use the same oracle planner used for generating the training demonstrations. Unlike the full version of the bi-level planning baselines, here the oracle planner can decide when the subgoal for a skill instruction is achieved. Thus, instead of updating the skill instruction at a fixed frequency, the high-level planner will generate the next skill instruction when the oracle planner has reached the subgoal of the current skill instruction.

We report the results in Table VII. Interestingly, compared with the results in Table V, every bi-level planner that uses a VLM for high-level task planning performs worse when paired with an oracle motion planner. This is likely because the oracle motion planner has to finish the entire execution of a subgoal, even if it is incorrect. In such cases, VLM-based high-level task planners struggle to recover from earlier mistakes. In contrast, when we used a learned low-level action policy, the same instruction will only be executed for a few steps (as described in Section IV-B). Consequently, the VLM has a better chance to correct those mistaken instructions in subsequent steps.

2) *Effects of Different Visual Inputs*: To examine the impact of different visual representations, particularly 2D and 3D part masks, on policy learning, we conduct another ablation study, where we evaluate the low-level action policies with various visual inputs. Specifically, in addition to *DP-S SAM2* and *DP3-S SAM2*, we also trained low-level action policies using ground-truth mask information, *DP-S GT* and *DP3-S GT*, as well as the vanilla models without any part-level mask, *DP* and *DP3*. The results are summarized in Table VIII. With part segmentations, either 2D or 3D, the low-level action policies can achieve significantly better performance. The performance gap between the policies trained with ground-truth part segmentation and SAM2-based part segmentation also suggests that there improvement in both the VLM’s ability

Table VIII: Impact of various vision inputs on low-level action policies. We pair low-level action policies using different vision inputs with ground-truth high-level plans.

Baselines	OS	OI	TP	TC	OC	All
DP	6.47	11.42	16.53	0.06	6.94	8.28
DP-S GT	15.45	20.91	26.36	0.91	13.33	15.39
DP-S SAM2	20.00	16.36	25.45	0.00	6.67	13.70
DP3	19.34	13.61	17.89	0.00	12.08	12.98
DP3-S GT	45.45	36.36	36.36	1.82	40.00	32.00
DP3-S SAM2	23.64	29.09	23.64	1.82	26.67	20.97

to ground fine-grained parts and in the capacity of state-of-the-art segmentation methods to accurately segment object parts.

V. DISCUSSION

How well can current vision-language policies perform in our part-level manipulation tasks? The experimental results on our benchmark systematically reveal the performance of current vision-language policies in our part-level manipulation tasks. Specifically, we find that vision-language policies perform adequately on object-level tasks but struggle with precise part-level grounding. While they can follow simple part-based instructions such as “grasp” or “touch,” instructions like “touch the left part” introduce fine-grained spatial reasoning that these models have not fully mastered. We observed that these policies can learn the broad action of “touch” but neglect the exact location of “left.” Second, zero-shot inference using pretrained generalist vision-language policies on our benchmark fails to achieve any success (see Appendix D2). This is likely due to the absence of part-level skills and detailed spatial reasoning in their training data. Current large-scale robotic datasets do not adequately capture the detailed spatial and part-specific annotations required for fine-grained part-level manipulation. This suggests the value of our training dataset and PartGym simulator in training part-level manipulation policies as we provide detailed part annotations as well as fine-grained manipulation tasks that require part grounding and reasoning.

Why is part-level instruction following challenging for vision-language policy learning? Our experimental results demonstrate that the part-level instruction following tasks in our PartInstruct benchmark remains extremely difficult for state-of-the-art end-to-end vision-language policy learning methods. There are several main challenges that these methods cannot yet solve for part-level instruction following. First, learned policies must recognize and track object parts over time, which can be difficult as parts of the same kind can have distinctive appearances. For instance, the object part “lid”, may look differently across object categories (e.g., the lid of a bottle vs. the lid of a pot, the top of a stapler vs. the top of a mug). This variability requires the model to correctly associate the same part name with distinct visual representations based on context. Second, relevant objects and the corresponding manipulation of these objects for performing a task may not be explicitly defined in the task instructions. Thus a policy must reason about what parts to interact with and in which manner. Third, the fine-grained nature of these tasks imposes a stricter success criterion than typical object-level manipulation.

For instance, in a general mug-picking task, any point on the mug’s surface might serve as a grasping point, whether on the handle, top, or body. In contrast, a task requiring grasping specifically by the handle demands precision in targeting the handle area alone, with the need for detailed semantic and spatial awareness.

Why is bi-level planning helpful? One important feature of bi-level planning is that it decomposes a complex task into a chain of subgoals, each interacting with at most one object part at a time. Focusing on a single part-level skill at a time simplifies the training of low-level action policies, as the policy only needs to ground the skill instruction into a relatively simple manipulation of the specified object part. Part-level manipulation also requires more fine-grained vision grounding than object-level tasks, since the part-level information is much more detailed and changes dynamically over time (e.g., the front of a mug at the current step may no longer be the front in future steps after rotation). By decomposing the task into part-level tasks, we reduce the burden of grounding and tracking different parts over time, enabling the low-level action policy to focus on the most relevant visual information at the moment. Additionally, separating reasoning from action execution allows us to incorporate pretrained foundation models. Specifically, high-level task planning is performed by VLMs pretrained on internet-scale data, which endows them with extensive prior knowledge and proficiency in high-level reasoning, planning, and language-guided decision making. As multimodal foundation models continue to advance in vision-language reasoning, their built-in knowledge is expected to further boost overall performance.

What kinds of visual representations are useful in fine-grained manipulation? As the tasks in PartInstruct require a model to have a detailed visual understanding of object parts, visual representations of the scene and objects may play a central role in a model’s performance. Our ablation study on the effect of visual representations in the model input reveals the following findings. First, 3D representations, such as point clouds, are more effective than 2D images. Unlike 2D methods, which can misinterpret depth and lead to positioning errors, point clouds provide precise 3D shape and location information, improving action success. Second, explicit object part segmentation provides a significant performance boost in the performance of part-level policy learning, as shown in Table VIII. The improvement is particularly noticeable for 3D part segmentation. In fact, DP3-S outperforms DP3 by approximately 20%, more than doubling the performance.

What are the difficulties of learning part-level skills, and what kind of skills are harder to learn than others? We found that part-level manipulation skills can be particularly challenging when they require indirect actions to achieve a goal state for a target part. To analyze which part-level skills are generally difficult to learn and which object parts tend to pose challenges for the robot, we conducted an impact study, detailed in Appendix C. The study shows that “grasp” and “touch” achieve success rates over 50%, likely because they involve direct physical contact with the part. By contrast,

“Rotate” achieves only 18.2%, since it specifies the orientation of the part rather than direct manipulation. For example, to show a bottle’s “cap,” the robot may avoid grasping the cap directly (which obstructs the view) and must instead grasp another part of the bottle, making it a more complex skill to learn. Additionally, compared to common parts (e.g., “handle” and “lid”), spatial parts (e.g., “left” or “right”) are much more challenging because these references can change as the object moves. For instance, an instruction like “Rotate the bottle, so that the left part faces the opposite direction” requires the policy to remember the original “left” region while also recognizing the updated orientation as the bottle rotates. Maintaining both the original reference and the changing spatial context makes these tasks particularly difficult.

How well can current VLMs perform in the planning for fine-grained manipulation tasks? Our experiments show that bi-level planning baselines significantly outperform end-to-end policy learning approaches, as indicated in Table V. This suggests that current VLMs possess certain capabilities in understanding and reasoning about part-level manipulation tasks, as well as generalizing pretrained knowledge to perform high-level task planning across diverse object- and part-related scenarios. However, VLM-based planners can still fail during task planning, particularly in tasks that require a long chain of skill instructions (e.g., tasks in Test 4). This poses challenges for future research to further improve VLMs’ reasoning and planning capacities for fine-grained manipulation tasks.

VI. LIMITATIONS

Our current study focuses on part-level manipulation tasks in a controlled 3D simulator, which has certain limitations when considering real-world deployment. First, we have not fully evaluated sim-to-real generalization. Although the dataset includes diverse objects and tasks, there is no guarantee that the learned policies will transfer seamlessly to physical robot platforms. Exploring techniques such as domain randomization or policy fine-tuning on real-world data could improve the robustness of the policies. Second, the demonstrations in our training set are generated by an oracle motion planner, which may have limited behavioral diversity. In the future, we plan to integrate a teleoperation interface in PartGym to collect human demonstrations. Third, our current benchmark focuses on single-object manipulation. Studying scenarios where multiple objects are present or clustered closely together is another important future direction. Finally, while we have diverse object instances, we can further enrich the object assets by including articulated objects (e.g., cabinets, drawers), which can be used for evaluating part-level manipulation under dynamic constraints.

VII. CONCLUSION

In this work, we introduced PartInstruct, a large-scale benchmark designed to advance fine-grained robot manipulation using part-level instructions. By curating a diverse set of objects, tasks, and expert demonstrations, PartInstruct provides a foundation for training and evaluating robot manipulation

models that require reasoning about object parts and their relationships with tasks. Our evaluations of state-of-the-art models highlight critical challenges in grounding part concepts and executing long-horizon tasks. With comprehensive experiments and ablation studies, our work provides key insights for future research, highlighting the need for further innovation in perception, reasoning, and planning to enable robots to effectively perform fine-grained, part-aware manipulation.

REFERENCES

- [1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.
- [2] Michel Breyer, Jen Jen Chung, Lionel Ott, Roland Siegwart, and Juan Nieto. Volumetric grasping network: Real-time 6 dof grasp detection in clutter. In *Conference on Robot Learning*, pages 1602–1611. PMLR, 2021.
- [3] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.
- [4] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.
- [5] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.
- [6] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. <http://pybullet.org>, 2016–2021.
- [7] Yufei Ding, Haoran Geng, Chaoyi Xu, Xiaomeng Fang, Jiazhao Zhang, Songlin Wei, Qiyu Dai, Zhizheng Zhang, and He Wang. Open6dor: Benchmarking open-instruction 6-dof object rearrangement and a vlm-based approach. In *First Vision and Language for Autonomous Driving and Robotics Workshop*, 2024.
- [8] Peter Florence, Lucas Manuelli, and Russ Tedrake. Self-supervised correspondence in visuomotor policy learning. *IEEE Robotics and Automation Letters*, 5(2):492–499, 2019.
- [9] Haoran Geng, Ziming Li, Yiran Geng, Jiayi Chen, Hao Dong, and He Wang. Partmanip: Learning cross-category generalizable part manipulation policy from point cloud observations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2978–2988, 2023.
- [10] Haoran Geng, Songlin Wei, Congyue Deng, Bokui Shen, He Wang, and Leonidas Guibas. Sage: Bridging semantic and actionable parts for generalizable articulated-object manipulation under language instructions. *arXiv preprint arXiv:2312.01307*, 2023.
- [11] Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3d: 3d feature field transformers for multi-task robotic manipulation. In *7th Annual Conference on Robot Learning*, 2023.
- [12] Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pages 694–710. PMLR, 2023.
- [13] Ankit Goyal, Valts Blukis, Jie Xu, Yijie Guo, Yu-Wei Chao, and Dieter Fox. Rvt-2: Learning precise manipulation from few demonstrations. *arXiv preprint arXiv:2406.08545*, 2024.
- [14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [15] Raisa Islam and Owana Marzia Moushi. Gpt-4o: The cutting-edge advancement in multimodal llm. *Authorea Preprints*, 2024.
- [16] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.
- [17] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: Robot manipulation with multimodal prompts. 2023.
- [18] Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.
- [19] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.
- [20] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.
- [21] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.
- [22] Weiye Liu, Jiayuan Mao, Joy Hsu, Tucker Hermans, Animesh Garg, and Jiajun Wu. Composable part-based manipulation. *arXiv preprint arXiv:2405.05876*, 2024.
- [23] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Silvio Savarese, and Li Fei-Fei. Learning to generalize across long-horizon tasks from human demonstrations.

arXiv preprint arXiv:2003.06085, 2020.

- [24] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.
- [25] Aoran Mei, Jianhua Wang, Guo-Niu Zhu, and Zhongxue Gan. Gamevlm: A decision-making framework for robotic task planning based on visual language models and zero-sum games. *arXiv preprint arXiv:2405.13751*, 2024.
- [26] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [27] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhaio Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021.
- [28] Abby O’Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.
- [29] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017. URL <https://arxiv.org/abs/1706.02413>.
- [30] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.
- [32] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3758–3765. IEEE, 2018.
- [33] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. URL <https://arxiv.org/abs/2408.00714>.
- [34] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos, 2024. URL <https://arxiv.org/abs/2408.00714>.
- [35] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks, 2024. URL <https://arxiv.org/abs/2401.14159>.
- [36] Sushmita Sarker, Prithul Sarker, Gunner Stone, Ryan Gorman, Alireza Tavakkoli, George Bebis, and Javad Sattarvand. A comprehensive overview of deep learning techniques for 3d point cloud classification and semantic segmentation. *Machine Vision and Applications*, 35(4): 67, 2024.
- [37] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.
- [38] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009, 2023.
- [39] Yuliang Sun, Xudong Zhang, and Yongwei Miao. A review of point cloud segmentation for understanding 3d indoor scenes. *Visual Intelligence*, 2(1):14, 2024.
- [40] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.
- [41] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.
- [42] Beichen Wang, Juexiao Zhang, Shuwen Dong, Irving Fang, and Chen Feng. Vlm see, robot do: Human demo video to robot action plan via vision language model. *arXiv preprint arXiv:2410.08792*, 2024.
- [43] Lionel Wong, Jiayuan Mao, Pratyusha Sharma, Zachary S Siegel, Jiahai Feng, Noa Korneev, Joshua B Tenenbaum, and Jacob Andreas. Learning adaptive planning representations with natural language guidance. *arXiv preprint arXiv:2312.08566*, 2023.

- [44] Zhenyu Wu, Ziwei Wang, Xiuwei Xu, Jiwen Lu, and Haibin Yan. Embodied task planning with large language models. *arXiv preprint arXiv:2307.01848*, 2023.
- [45] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [46] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11097–11107, 2020.
- [47] Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*, 2024.
- [48] Maryam Zare, Parham M Kebria, Abbas Khosravi, and Saeid Nahavandi. A survey of imitation learning: Algorithms, recent developments, and challenges. *IEEE Transactions on Cybernetics*, 2024.
- [49] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy. *arXiv preprint arXiv:2403.03954*, 2024.
- [50] Shengqiang Zhang, Philipp Wicke, Lütfti Kerem Şenel, Luis Figueredo, Abdeldjallil Naceri, Sami Haddadin, Barbara Plank, and Hinrich Schütze. Lohoravens: A long-horizon language-conditioned benchmark for robotic tabletop manipulation. *arXiv preprint arXiv:2310.12020*, 2023.
- [51] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 5628–5635. IEEE, 2018.

APPENDIX

A. PDDL Definitions

1) *Predicate Definitions*: This subsection gives the definition of the basic predicates utilized by the motion planner.

Table IX: Definition of Predicates

Predicate	Description
ON(obj, part, contact)	Whether obj is on the contact.
TOUCHING(obj, part)	Whether the gripper is in contact with obj at part; part can be empty to indicate a general touch on any parts.
GRASPING(obj, part)	Whether the gripper is carrying obj at part; part can be empty to indicate a general grasp with any parts.
FACING(obj, part, dir)	Whether part of obj is facing or pointing dir.
AT_POSITION(obj, pos)	Whether obj is at position pos = [x,y,z].

2) *Skill Definitions*: This subsection shows the detailed definition of the five skills.

Table X: Definition of Base Skills

Skill	Description	Preconditions	Effects
grasp_obj(obj, part)	Robot grasps <i>obj</i> at <i>part</i> .	ON(table, obj); ~GRASPING(obj); ~TOUCHING(obj)	GRASPING(obj, part)
move_gripper(dir, dis=UNIT, grasping=false)	Robot moves gripper along <i>dir</i> <i>dis</i> .	If grasping==True: GRASPING(obj)	AT_POSITION(gripper, last_gripper_pos + vec(dir) × dis); If grasping==True: GRASPING(obj)
rotate_obj(obj, part, dir)	Robot rotates <i>obj</i> , such that <i>part</i> is facing <i>dir</i> .	GRASPING(obj)	GRASPING(obj); FACING(part, dir)
touch_obj(obj, part)	Robot touches <i>obj</i> at <i>part</i> .	ON(table, obj); ~GRASPING(obj); ~TOUCHING(obj)	TOUCHING(obj, part)
release_gripper(obj)	Robot releases the gripper and moves away from <i>obj</i> .	ON(table, obj); GRASPING(obj) <i>or</i> TOUCHING(obj)	ON(table, obj); ~GRASPING(obj); ~TOUCHING(obj)

3) *Task Definitions*: This subsection shows the detailed definition of different task types in PartInstruct.

Table XI: Seen Task Instructions and Goal States

Seen (10)		
Order	Example Task Instruction	Goal States
1	Grasp the <i>object</i> by the <i>part</i>	GRASPING(gripper, part), ON(obj, table)
2	Touch the <i>object</i> at the <i>part</i>	TOUCHING(part), ON(obj, table)
3	Hold the <i>part</i> of the <i>object</i> and move it to <i>direction</i>	GRASPING(part), AT_POSITION(obj, POS_INIT_OBJ+VEC(dir))
4	Push the <i>object</i> towards <i>direction</i> by touching <i>part</i>	TOUCHING(part), AT_POSITION(obj, POS_INIT_OBJ+VEC(dir))
5	Slide the <i>object</i> on the table towards <i>direction</i> while keeping hold of <i>part</i> , then release it	Phase1: GRASPING(part), AT_POSITION(obj, POS_INIT_OBJ+VEC(dir)) Phase2: GRIPPER_OPEN, MIN_DISTANCE(gripper, obj)
6	Move the <i>object</i> to <i>direction</i> by pushing it at <i>part</i> , then free it	Phase1: TOUCHING(part), AT_POSITION(obj, POS_INIT_OBJ+VEC(dir)) Phase2: GRIPPER_OPEN, MIN_DISTANCE(gripper, obj)
7	While keeping hold of <i>part</i> , move the <i>object</i> towards <i>direction</i> in the air	GRASPING(part), AT_POSITION(obj, POS_INIT_OBJ+VEC(dir)+VEC(UP))
8	Rotate <i>part</i> of the <i>object</i> to face <i>direction</i> while lifting it	GRASPING(obj), FACING(part, dir), AT_POSITION(obj, POS_INIT_OBJ+VEC(UP))
9	Move the <i>object</i> towards <i>direction</i> after raising it, while keeping hold of <i>part</i> , then put it down	Phase1: GRASPING(part), AT_POSITION(obj, POS_INIT_OBJ+VEC(dir)+VEC(UP)) Phase2: GRASPING(part), AT_POSITION(obj, POS_INIT_OBJ+VEC(dir))
10	Move the <i>object</i> towards <i>direction1</i> in the air, then rotate <i>part</i> to point towards <i>direction2</i>	GRASPING(obj), AT_POSITION(obj, POS_INIT_OBJ+VEC(UP)+VEC(dir1)), FACING(part, dir2)

Table XII: Unseen Task Instructions and Goal States

Unseen (6)		
Order	Example Task Instruction	Goal States
11	Rotate <i>part</i> in the air so it points towards <i>direction</i> , then put it down	Phase1: GRASPING(obj), FACING(part, dir), AT_POSITION(obj, POS_INIT_OBJ+VEC(UP)) Phase2: GRASPING(obj), FACING(part, dir), AT_POSITION(obj, POS_INIT_OBJ)
12	Shift the <i>object</i> towards <i>direction1</i> in the air while grasping <i>part1</i> , turn <i>part2</i> to <i>direction2</i> , then set it down	Phase1: GRASPING(part1), FACING(part, dir), AT_POSITION(obj, POS_INIT_OBJ+VEC(dir1)+VEC(UP)) Phase2: GRASPING(part1), FACING(part, dir), AT_POSITION(obj, POS_INIT_OBJ+VEC(dir1))
13	Turn <i>part</i> of the <i>object</i> to point to <i>direction1</i> while keeping it on the table, then push it towards <i>direction2</i>	Phase1: ON(obj, table), FACING(part, dir1); Phase2: ON(obj, table), AT_POSITION(obj, POS_INIT_OBJ+VEC(dir2))
14	While keeping it on the table, push the <i>object</i> towards <i>direction1</i> while touching <i>part1</i> , then rotate <i>part2</i> to face <i>direction2</i>	Phase1: ON(obj, table), TOUCHING(part1), AT_POSITION(obj, POS_INIT_OBJ+VEC(dir1)) Phase2: ON(obj, table), FACING(part2, dir2)
15	Rotate <i>part</i> of the <i>object</i> to face the opposite direction	FACING(part, ~DIR_INIT(part)), ON(obj, table)
16	Push the <i>object</i> to <i>direction1</i> and rotate <i>part</i> to point towards <i>direction2</i> in the air, finally place it down	Phase1: FACING(part, dir2), AT_POSITION(obj, POS_INIT_OBJ+VEC(dir1)+VEC(UP)) Phase2: FACING(part, dir2), AT_POSITION(obj, POS_INIT_OBJ+VEC(dir1))

B. PartInstruct Benchmark Details

1) *Observation and Action Space*: Table XIII shows the observation and action space available in *PartGym*.

Table XIII: Observation and Action Space details.

Observation Space	
Static View - RGB	$300 \times 300 \times 3$
Static View - Depth	300×300
Static View - PCD	3×1024
Static View - Semantic	300×300
Static View - Target Part PCD	3×1024
Static View - Target Part Mask	300×300
Wrist View - RGB	$300 \times 300 \times 3$
Wrist View - Depth	300×300
Wrist View - PCD	3×1024
Wrist View - Semantic	300×300
Wrist View - Target Part Mask	300×300
Wrist View - Target Part PCD	3×1024
Proprioceptive state	EE position (3) EE orientation (3) Joint positions (7) Gripper action (1)
Action Space	
Absolute cartesian pose (w.r.t. world frame)	EE position (3) EE orientation (3) Gripper action (1)

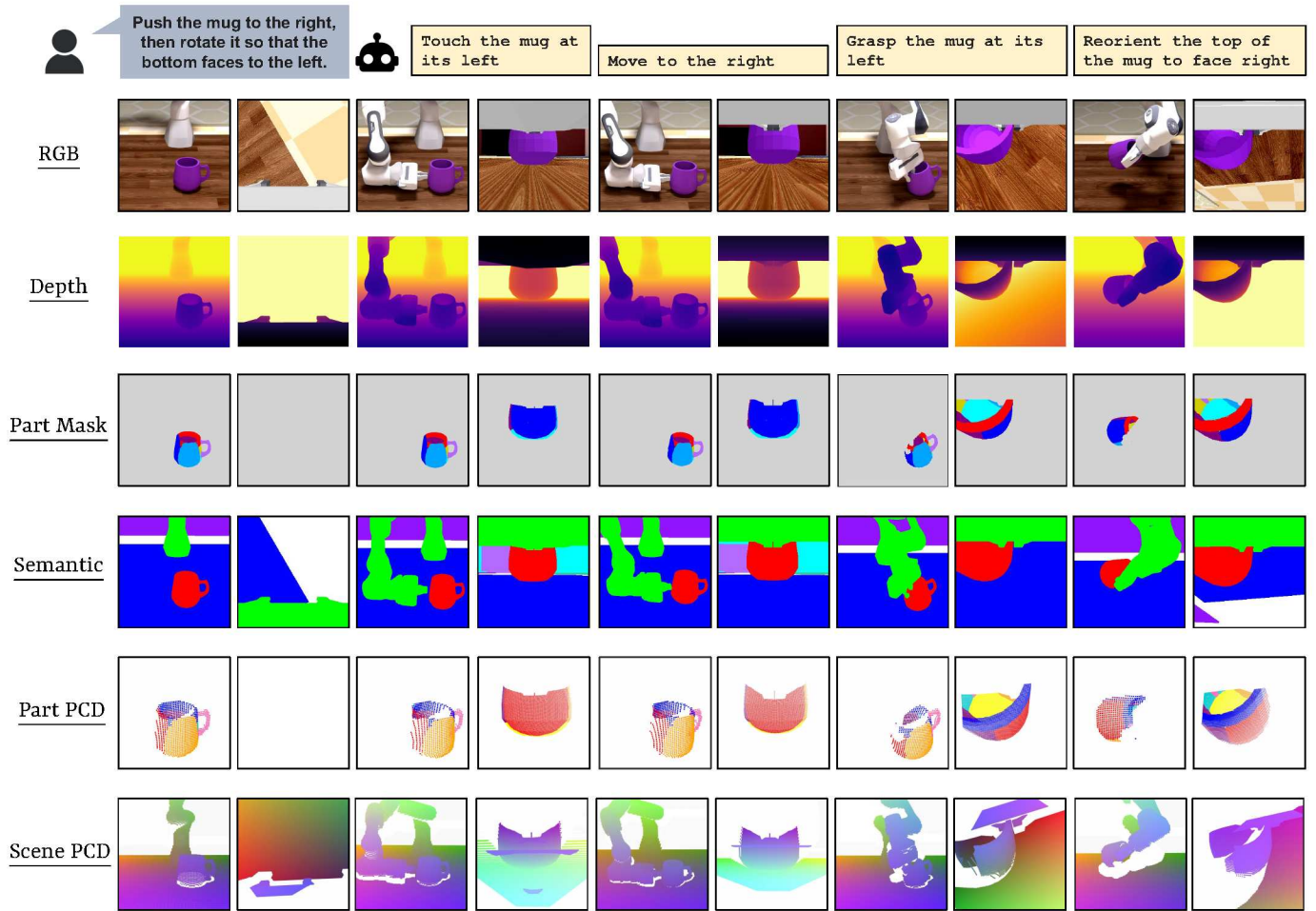



Figure 9: Selected Visual Modalities in PartGym.

2) *Key Features of PartGym*: The aim of PartGym is to boost embodied AI research related to interaction with table-top object parts. PartGym support real-time rendering of different visual modalities (see Figure 9). In addition to the typical modalities like RGB, depth and object segmentation, PartGym also provides part-related visions like part masks and part point cloud, including spatial parts and semantic parts of an object. These part-related vision modalities are rendered by *PyBullet* [6] simulation engine using the ground-truth part assets given by *PartNet Mobility* [45] [26] [4].

Additionally, PartGym provides a framework to implement bi-level planning models for part-level manipulation tasks in simulation environments. It provides a template skill instruction generator, an oracle skill execution checker, as well as a systematical way to render part-related modalities shown in any skill instruction.

3) *Visualization of Test Splits*: We provide the visualization of all 5 test sets in this section.

 **Move the bottle on the table away from me while holding top, then free it.**

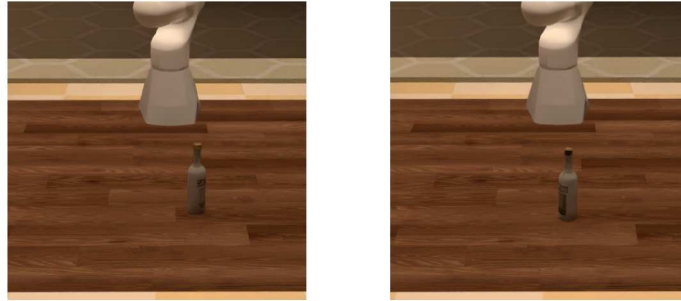


Figure 10: Left: Training set. Right: Test 1(OS).

 **Place gripper tip on the screw of the scissors.**

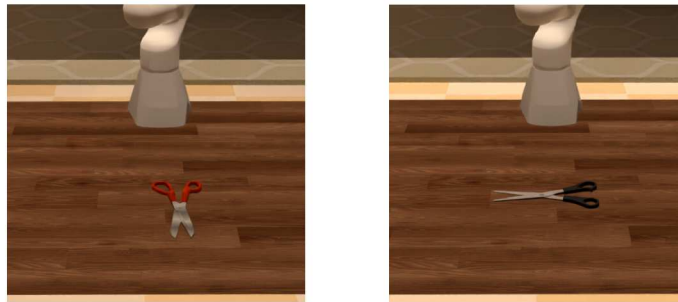




Figure 11: Left: Training set. Right: Test 2(OI).

 **Grab the top of the mug and move it forwards.**



 **Grab the handle of the mug and move it backwards.**

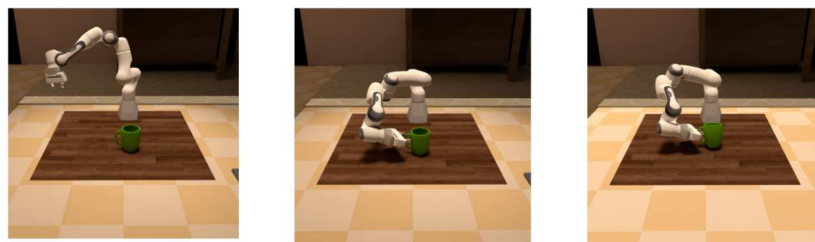
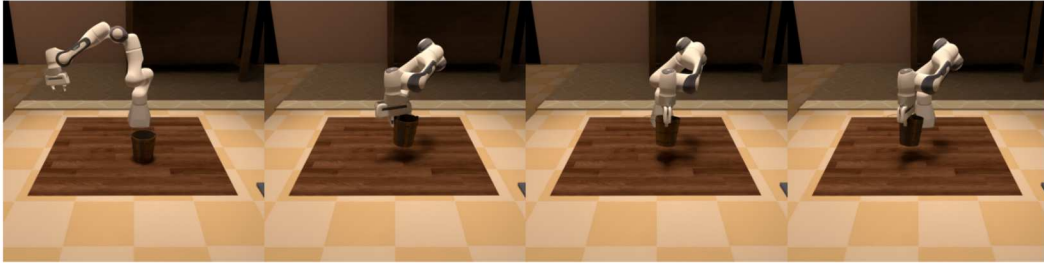



Figure 12: Above: Training set. Below: Test 3(TP).

 **Lift the bucket by its left, then rotate the left part to face front, then move it to the left.**



 **Move the bucket towards to the left in the air, then rotate handle to point towards back.**

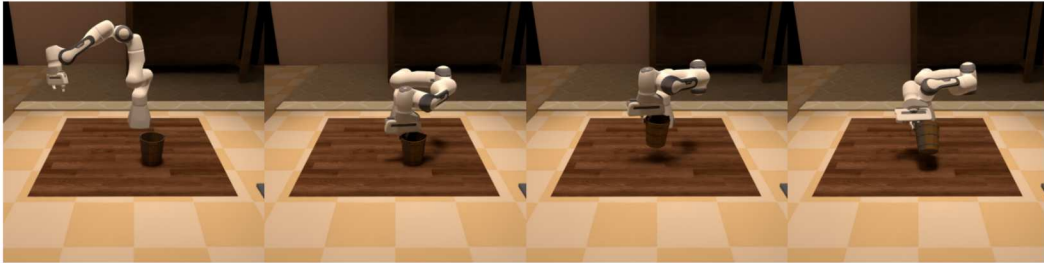


Figure 13: Above: Training set. Below: Test 4(TC).

 **Lift the bottle by its top.**



 **Lift the knife by its base body.**



Figure 14: Left: Training set. Right: Test 5(OC).

4) *Statistics of PartInstruct Episodes*: We provided detailed statistics about parts within each object type.

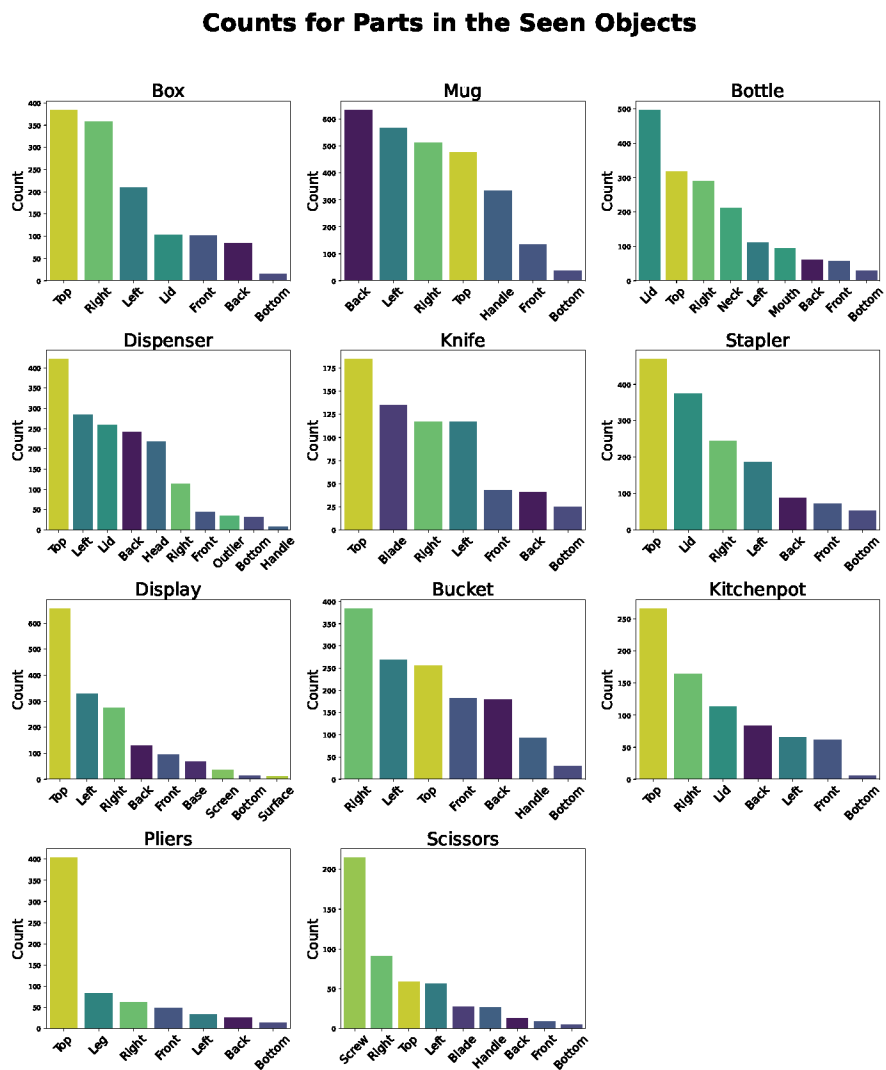


Figure 15: Parts in PartInstruct episodes, grouped by seen object types.

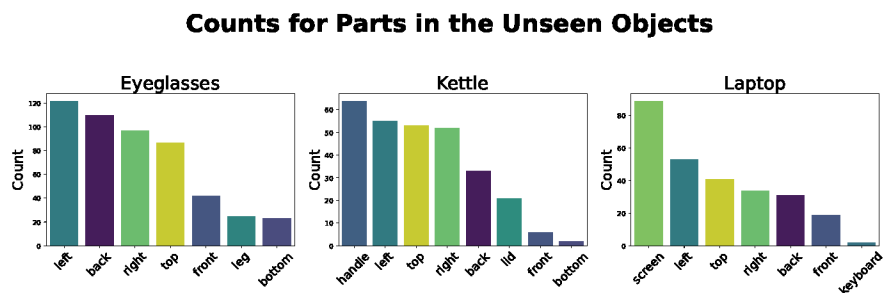


Figure 16: Parts in PartInstruct episodes, grouped by unseen object types.

C. Skill and Object Part Impact Study

Here, we selected the rollout logs of the best-performing policy and analyzed the impact of different skill types and object parts. Specifically, we evaluated the success rate and failure causes for each skill and part.

The Success Rate was calculated by dividing the number of successful executions of each skill or part by the number of times it appeared in the skill chain. The Failure Cause was calculated by dividing the number of times a skill chain failed because of a specific skill or part by the total number of skill chain failures.

Table XIV: Average success rate and failure cause for the three part-level skills

Skill	Grasp Object	Rotate Object	Touch Object
Success Rate (%)	53.55	18.18	54.55
Failure Cause (%)	43.51	6.11	16.79

Table XV: Average success rate and failure cause for selected parts.

Part	Blade	Left	Neck	Top	Screen	Mouth	Bottom
Success Rate (%)	46.67	45.10	66.67	52.78	60.00	66.67	30.00
Failure Cause (%)	4.60	13.79	1.15	24.14	2.30	0.00	0.00

Part (Continued)	Handle	Leg	Lid	Front	Right	Back	Screw	Head
Success Rate (%)	64.29	33.33	63.64	29.03	36.49	41.03	0.00	16.67
Failure Cause (%)	4.60	2.30	4.60	5.75	21.84	9.20	1.15	3.45

D. Implementation Details

1) Training Details in End-to-End Policy Learning: We trained the baseline models, including Diffusion Policy (DP) [5], 3D Diffusion Policy (DP3) [49], and Act3D [11], from scratch. For RVT2 [12] and Octo [41], we implemented both fine-tuning of the pretrained models and training from scratch on our dataset. All trained models are using vision modalities from a static-view camera put with the same extrinsics in the workspace, as well as the real-time robot states information. Experiments were conducted on cluster nodes of A100 or H100 using Distributed Data Parallel (DDP). Training from scratch generally took about two days, while fine-tuning required one day.

Diffusion Policy (DP): We train CNN-based DP from scratch on our dataset. The action prediction horizon is set to 16 steps, with an observation horizon of 2 steps and action steps of 8. The input RGB images are cropped to a size of 76×76 . For language instructions, we use a pre-trained T5-small language encoder to obtain a language embedding of 512 dimensions. This language embedding is then concatenated with other features to form the final feature representation.

3D Diffusion Policy (DP3): The DP3 model is trained under a similar setup as DP, with an action prediction horizon of 16 steps, an observation horizon of 2 steps, and action steps of 8. For the point cloud observations, we use an input size of 1024 points, which are downsampled from the original point cloud using the Iterative Farthest Point Sampling algorithm [29]. The language instructions are processed in DP3 following the same approach as in DP.

Act3D: Act3D takes an image input size of 256×256 . The action prediction horizon is set to 6 steps, and the observation horizon is 1 step. Following the raw work [11], we use ResNet50[14] as the vision encoder, and use CLIP [30] embeddings for vision-language alignment. For 3D action map generation, the number of “ghost” points is set to be 10,000, with a number of sampling level of 3.

3D Diffuser Actor (3D-DA): For 3D-DA, we use the front-view RGB and scene point cloud as vision inputs. The RGB image has a resolution of 256×256 . Following Ke et al. [18], we extract visual features with a pre-trained CLIP ResNet-50 encoder and use CLIP [30] embeddings for vision-language alignment. We use an interpolation length of 5 steps and an observation history of 3 steps.

Octo: For fine-tuning, we use the released checkpoint of the `octo-base-1.5` model and fine-tune its output head for 20,000 iterations. We use both the static view camera and the wrist view camera. The input image sizes are 256×256 for the static view and 128×128 for the wrist view. The window size is set to 2 steps, and the action horizon is set to 16 steps.

RVT2: To adapt RVT2 in our benchmark settings, we first convert the depth map from the static camera view into a point cloud in the camera coordinates, then apply camera extrinsic to transfer the point cloud into the world coordinates, where the action heat maps will be generated and apply supervision in. The action prediction horizon is chosen to be 6 steps, and the observation horizon is set to be 1 step.

2) *Zero-Shot Evaluation of the Generalist Policy*: We selected several popular generalist policy, including RT-1, Octo and OpenVLA, and evaluated their zero-shot performance on our test sets. For RT-1, we followed the implementation of Open X-Embodiment project and used the released `rt_1_x_tf_trained_for_002272480_step` checkpoint for inference. For Octo, we used `octo-base-1.5` model, following the same setup as described in section D.1. For OpenVLA, we used the pretrained model `openvla-7b`. We followed the same evaluation protocol as other baselines, and our results shows that these generalist policies fail to achieve any success on our test sets.

3) *Design Details of Bi-Level Planning*: We outline the bi-level planning pipeline’s implementation here as a supplement to Section IV-B.

Implementation of the High-Level Task Planner: The high-level task planner features a skill inference mechanism that leverages comprehensive contextual information—including user task instructions, previously executed skill chains, and real-time state data such as vision and pose information—to determine the next appropriate action. Recall that the high-level task planner updates the skill instruction once every n steps. Here n is determined by the average number of steps typically required for each skill in the training dataset. Specifically, we use 130 for **grasp_obj**, 30 for **move_gripper**, 68 for **touch_obj**, 40 for **release_obj**, and 22 for **rotate_obj**. Once the execution counter reaches these average values, the VLM is prompted to infer the subsequent skill based on the current state. This design ensures that the decisions are grounded in both historical and real-time data. In addition, the planner incorporates an exception handling measure to maintain output consistency and reliability. Any unacceptable terms generated by the VLM—such as directional indicators that out of our definition, will be normalized to their prescribed equivalents. Also, despite embedding all acceptable part names within the prompt, a dedicated mechanism cross-references any inferred part names against a stored list of valid part names. For VLM baselines, we use `gemini-1.5-flash-002` and `gemini-2.0-flash-exp` for Gemini [40], and `GPT4o` for OpenAI models [15].

Training of the Low-Level Action Policies: We trained the low-level action policy using skill instructions retrieved from our training data, assuming the presence of an oracle planner that decomposes the overall task. Apart from the skill instructions, the training setup remains identical to the end-to-end learning approach described in Section D.1.

Part Grounding and Tracking: We selected `sam2_hiera_small` as our mask generation and tracking model due to its fastest tracking time among all configurations of SAM 2. For language grounding, we chose `Florence-2-large` as our Vision-Language Model (VLM). To evaluate the performance, we used the rollout logs of *DP-S SAM2*.

The performance was assessed using two key metrics: Grounding Success, Intersection over Union (IoU). Grounding Success is calculated as the ratio of successfully grounded parts to the total number of parts during a task. A grounding is considered successful if: 1) after language grounding, the prompt points given by the VLM consist of one positive and one negative point (to prompt SAM 2), and 2) the IoU of the generated mask is greater than zero. If either of these conditions is not met, the grounding is deemed a failure. The IoU measures the overlap between the predicted mask generated by SAM 2 and the ground-truth mask retrieved from PartGym environment, it is defined as the area of intersection divided by the area of union of the predicted and true regions. The results across different test sets are summarized in Table XV.

Table XV: Performance of part grounding and tracking across different test sets.

Metric	Test1	Test2	Test3	Test4	All
Grounding Success (%)	25.26	35.73	36.87	15.06	27.58
IoU	0.15	0.18	0.19	0.25	0.20

VLM Prompts: We provide an example VLM prompt used in our bi-level planning pipeline below.

Prompt Example

You are an expert at planning manipulation tasks. You will be given one task instruction for each manipulation task. Each task instruction can be divided into a chain of skill instructions. Your job is to infer the next skill instruction (you only need to output one immediate next skill instruction each time, even if the entire task requires multiple skills) for the robot to execute, based on the following information and the attached image (current rgb frame) without outputting any intermediate inference and explanation:

- **Task Instruction:** {user_input}
- **Executed Skill Instructions:** {executed_skill_instructions}
- **Gripper State:** gripper_state (The gripper is open when the value is around 0.04 and it is closed when the value is around 0.00.)
- **Previous TCP Pose:** {previous_tcp_pose}
- **Current TCP Pose:** {current_tcp_pose}
- **Previous RGB Observation:** {prev_image_desc}
- **Current RGB Observation:** The {current_image_position} image in the contents array shows the current state.

Here is relevant information about the task.

1. The task instruction helps you understand the overall task goal. The executed skill instructions shows the sequence of actions taken so far.
2. The gripper state shows whether the gripper is open or closed. The TCP poses and images together illustrate the state transitions of the previous action.
3. The current object state, relative to the gripper, and the object motion (from TCP and images) can help determine if the last action was successful.

Skill Descriptions:

1. **grasp_obj:**

- **Description:** This skill grasps an object by a specific part.

- **Parameters:**

part_grasp: The exact part of the object to be grasped. Must match the user's input (e.g., 'blade', 'lid').

- **Format:** Grasp the {obj_class} at its {part_grasp}

2. **move_gripper:**

- **Description:** This skill moves the gripper in a specified direction while optionally keeping an object grasped.

- **Parameters:**

dir_move: Direction to move the gripper. Can only be 'top', 'bottom', 'left', 'right', 'front', or 'back'.

- **Format:** Move {dir_str}, where 'dir_str' is mapped from 'dir_move' by: - 'front' → 'forwards' - 'back' → 'backwards' - 'top' → 'upwards' - 'bottom' → 'downwards' - 'left' → 'to the left' - 'right' → 'to the right'

3. **rotate_obj:**

- **Description:** This skill rotates an object in a specific direction based on a given part.

- **Parameters:**

dir_rotate: Direction to rotate the object. Must be one of 'top', 'bottom', 'left', 'right', 'front', 'back'.

part_rotate: The part of the object that should be rotated.

- **Format:** Reorient the {part_rotate} of the {obj_class} to face {dir_str}, where 'dir_str' is mapped from 'dir_rotate'.

4. **touch_obj:**

- **Description:** This skill touches a part of an object.

- **Parameters:**

part_touch: The part of the object to be touched.

- **Format:** Touch the {obj_class} at its {part_touch}

5. **release_obj:**

- **Description:** This skill releases an object from the gripper.

- **Parameters:** None.

- **Format:** Release

Prompt Example (Continued)

Part Names:

- **Scissors:** blade, handle, screw, left, right, top, bottom, front, back
- **Kitchen Pot:** base body, lid, left, right, top, bottom, front, back
- **Laptop:** base frame, screen, touchpad, keyboard, screen frame, left, right, top, bottom, front, back
- **Eyeglasses:** base body, leg, left, right, top, bottom, front, back
- **Bucket:** handle, base body, left, right, top, bottom, front, back
- **Display:** base support, surface, frame, screen, left, right, top, bottom, front, back
- **Pliers:** base body, leg, outlier, left, right, top, bottom, front, back
- **Bottle:** mouth, lid, body, neck, left, right, top, bottom, front, back
- **Knife:** base body, translation blade, rotation blade, left, right, top, bottom, front, back
- **Stapler:** base body, lid, body, left, right, top, bottom, front, back
- **Kettle:** handle, base body, lid, left, right, top, bottom, front, back
- **Mug:** handle, body, containing things, left, right, top, bottom, front, back
- **Box:** rotation lid, base body, left, right, top, bottom, front, back
- **Dispenser:** base body, pressing lid, head, handle, outlier, left, right, top, bottom, front, back

Task Splitting Example:

Break down the task: Split the task instruction into individual steps. Example: \Move the box in the air towards the right while keeping touch of right, then put it down."

Steps: (1) Grasp the box at its right, (2) Move upwards, (3) Move to the right, (4) Move downwards. *Return only the next skill instruction in the specified format.*

Notes:

- Do not modify or assume alternate names for object parts.
- The task sequence should follow the user's input as strictly as possible.
- Do not replace object parts with similar or inferred names.