RH20T-P: A Primitive-Level Robotic Manipulation Dataset towards Composable Generalization Agents in Real-world Scenarios

Zeren Chen[1,2]*, Zhelun Shi[2]*, Xiaoya Lu[1,3]*, Lehan He[2]*, Sucheng Qian[3], Zhenfei Yin[1,4], Wanli Ouyang[1,4], Jing Shao[1]†, Yu Qiao[1], Cewu Lu[3]†, Lu Sheng[2]†

[1] Shanghai AI Laboratory, [2] School of Software, Beihang University, [3] Shanghai Jiao Tong University, [4] University of Sydney

{czr1604,shizhelun,lsheng}@buaa.edu.cn, shaojing@pjlab.org.cn

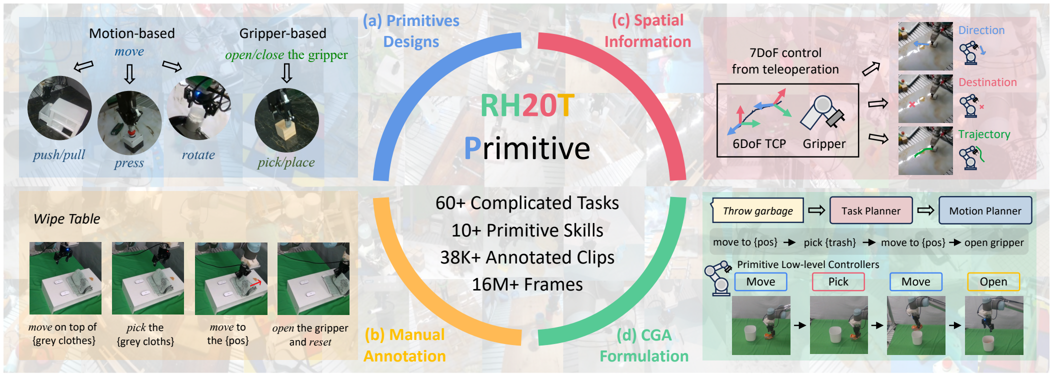


Figure 1: **Overview of our RH20T-P dataset.** We propose a primitive-level robotic manipulation dataset for future development of composable generalization agents. With meticulously-defined primitive skills and multiple forms of spatial information, each manipulation episode is carefully annotated manually. We also standardize a plan-execute paradigm for CGAs based on RH20T-P, which can generalize to novel skills through composable generalization.

Abstract

Achieving generalizability in solving out-of-distribution tasks is one of the ultimate goals of learning robotic manipulation. Recent progress of Vision-Language Models (VLMs) has shown that VLM-based task planners can alleviate the difficulty of solving novel tasks, by decomposing the compounded tasks as a plan of sequentially executing primitive-level skills that have been already mastered. It is also promising for robotic manipulation to adapt such composable generalization ability, in the form of composable generalization agents (CGAs). However, the community lacks of reliable design of primitive skills and a sufficient amount of primitive-level data annotations. Therefore, we propose RH20T-P, a primitive-level robotic manipulation dataset, which contains about 38k video clips covering 67 diverse manipulation tasks in real-world scenarios. Each clip is manually annotated according to a set of meticulously designed primitive skills that are common in robotic manipulation. Furthermore, we standardize a plan-execute CGA paradigm

*Equal contribution.
†Corresponding author.

and implement an exemplar baseline called RA-P on our RH20T-P, whose positive performance on solving unseen tasks validates that the proposed dataset can offer composable generalization ability to robotic manipulation agents.

1 INTRODUCTION

Robotic manipulation tasks are designed to enable robotic systems to comprehend environmental observations and open-world task instructions like language, guiding actuators to execute specific actions in real-world applications. Previous attempts at imitation learning [1, 4, 29, 35] or reinforcement learning [6–9] often struggle to generalize to out-of-distribution tasks. Since collecting all real-world tasks to build in-distribution training datasets is impractical, crafting generalizable robotic manipulation agents in this fashion poses challenges.

Recently, Vision-Language Models (VLMs) [13, 15] have shown impressive potential in following multimodal instructions. Some approaches [2] fine-tune VLMs by aligning the VLMs' language decoders (such as PaLM-E [16]) with the distribution of action sequences in robotic manipulation tasks. While to some extent, they can generalize to tasks involving novel objects or novel compositions between known skills and known objects, the skill set is restricted to those encountered in the training datasets. Alternatively, other methods [16, 27, 28] employ VLMs as task planners, breaking down a compounded task-solving procedure into a sequence of primitive-level skill-execution subroutines. We formulate this promising paradigm as ***composable generalization***, which facilitates that, as shown in Figure 1 (d), a novel skill "throw" that never presents in the training data can be accomplished by executing more straightforward and common skills, such as {*move*, *pick*, *move*, *open*} in composition. We argue that the composable generalization agents (CGAs) that follow this paradigm can mitigate the unpredictability and intricacy when encountering out-of-distribution compounded robotic manipulation tasks.

Existing CGAs [27, 28] primarily focus on planning systems with off-the-shelf VLMs like GPT-4V [13], while the research on entire CGA system remains insufficient. Particularly, ***how to predict reliable primitive-level spatial information that grounds successful executions of primitive skills?*** We refer it as primitive-level motion planning. For example, identifying virtual points in 3D space for generating robust trajectories without touching obstacles. VLMs struggle to provide such necessary primitive-level spatial information. While these agents can delegate motion planning to low-level controllers or integrate an additional motion planner, a lack of primitive-level spatial knowledge in current robotic manipulation datasets makes it hard for acquiring specialized controllers or motion planners, resulting in low execution success rates in more compounded tasks. Thus, we are motivated to collect **RH20T-P**, a robotic manipulation dataset built on **RH20T** [30] at a **P**rimitive level, with meticulously designed primitive skills and diverse primitive-level spatial knowledge, making it feasible to construct generalizable CGAs in real-world scenarios.

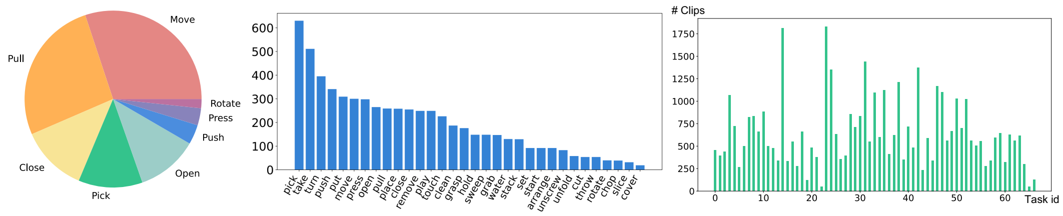
In RH20T-P, we design a set of hierarchical and scalable primitive skills based on two types of skills, *i.e.*, motion-based and gripper-based skills. Each motion-based skill is equipped with various forms of spatial knowledge like trajectories. Next, manipulation episodes in RH20T-P are manually segmented accordingly (about 38k clips covering 67 tasks). Additionally, we standardize a plan-execute CGA paradigm, as shown in Figure 1 (d), and implement an exemplar baseline CGA on RH20T-P, called **RA-P** (**R**obot**A**gent-**P**rimitive). Our RA-P showcases feasibility and generzalization in real-world demonstrations, even on novel skills, validating the composable generalization ability offered by RH20T-P. We believe that the RH20T-P dataset will pave the way for the development of more potent CGAs in the future.

2 RELATED WORK

Vision-Language Models (VLMs). Vision-Language Models (VLMs) have gained significant attention due to their multimodal perception capabilities. Some studies [15, 19–22] incorporate image semantics into language models, dedicated to understanding 2D images. Among these, LLaVA [15] adopts a two-stage instruction-tuning pipeline for general-purpose visual-language understanding. There are also some studies [17, 18, 23] on VLMs in 3D vision. For example, 3D-LLM [23] introduces the 3D semantics into LLMs by rendering point clouds into 2D images, enabling the models to perform a range of 3D tasks.

Table 1: **Comparison with Existing Robotic Manipulation Dataset.**

Dataset	Amount	Modality		Annotation		
		Image	Depth	Action Seq.	Task Desc.	Plan Segmentation
Bridge [37]	7.2k	multi-view	✗	✓	✓	✗
Open-x-embodiment [38]	1M	multi-view	✓	✓	✓	✗
Droid [49]	76k	multi-view	✓	✓	✓	✗
Lang-cond [39]	50k	multi-view	✗	✓	✓	free-form language
Language Table [34]	594k	single-view	✗	✓	✓	free-form language
RH20T-P (ours)	38k	multi-view	✓	✓	✓	pre-defined primitives



(a) Primitive skills in RH20T-P (b) Distribution of skills in RH20T-P (c) Total number of annotated clips in each task

Figure 2: **Data statistics of RH20T-P dataset.**

VLMs as Task Planners. Applying VLMs for planning [16, 27, 28, 47] in robotic tasks have shown great potential. PaLM-E [16] develops an embodied VLM and trains it jointly on web-scale datasets, but the unavailability of primitive skills makes the granularity of output actions changing. VILA [27] and GPT4Robotics [28] conduct ICL with GPT-4V [13], to generate primitive skills. The proprietary nature of GPT-4V leads to extensive prompt engineering and inflexibility to utilize other multi-modal observations, *e.g.*, depth.

Primitive-level Robotic Manipulation Datasets. Using VLMs as planners for task decomposition has urgently increased the demand for primitive-level datasets tailored for CGAs. We provide a comparison with existing robotic manipulation dataset in Table 1. Most robotic manipulation datasets [30–33, 35, 37, 38] either lack textual annotations or only provide language descriptions for entire tasks, which are insufficient for CGAs. While several datasets [34, 36, 39] feature hindsight free-form language for manipulation video clips, these coarsely-grained decompositions may confuse the low-level controllers. The absence of robotic manipulation datasets with fine-grained primitive skills hinders the development of CGAs.

3 RH20T-P: A Primitive-level Robotic Manipulation Dataset

3.1 Preliminary of RH20T and Data Sampling

RH20T[3] [30], data source of RH20T-P, encompasses a broad spectrum of real-world robot manipulation demonstrations, with each episode featuring diverse contexts, camera viewpoints, and language descriptions. Its diversity and executability are pivot components for the development of intelligent robotic agents. This motivates us to conduct primitive-level annotations for RH20T. Specifically, we sample a subset of tasks in RH20T that are suitable for CGAs (*e.g.*, visual reasoning) to construct a primitive-level robotic manipulation dataset. The dataset still retains numerous complex skills beyond pick-and-place skills after sampling, such as wiping the table with a sponge or arranging pieces on a chessboard to complete the initial setup.

3.2 Hierarchical and Scalable Primitive Skills

Primitive skills are crucial for RH20T-P dataset, as the way we decompose tasks shapes the design of the CGAs paradigm. Applying free-form natural language [34, 36, 39] as primitive skills are overly coarse-grained, increasing difficulty for controllers to grasp task semantics, deviating from the original intent of CGAs. Formally, we define primitive skills from the perspective of the robot arms,

[3]https://rh20t.github.io, MIT license.

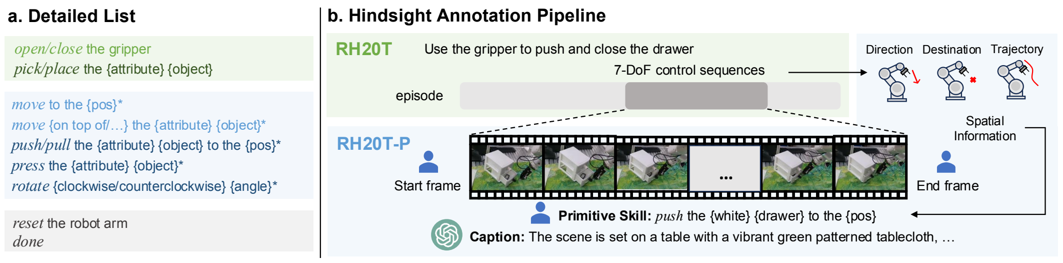

a. Detailed List
open/close the gripper
pick/place the {attribute} {object}
move to the {pos}*
move {on top of/…} the {attribute} {object}*
push/pull the {attribute} {object} to the {pos}*
press the {attribute} {object}*
rotate {clockwise/counterclockwise} {angle}*
reset the robot arm
done
b. Hindsight Annotation Pipeline
RH20T
Use the gripper to push and close the drawer
7-DoF control sequences
episode
Direction
Destination
Trajectory
Spatial Information
RH20T-P
Start frame
...
End frame
Primitive Skill: *push* the {white} {drawer} to the {pos}
Caption: The scene is set on a table with a vibrant green patterned tablecloth, …

Figure 3: **(a) Detailed list of primitive skills in RH20T-P.** "*" indicates this primitive skill contains spatial information of multiple form, *e.g.*, destination or trajectory. **(b) Process of hindsight primitive-level annotation.**

focusing on the state changes that primarily occur in the robot arm's motion and gripper during the manipulation process. As shown in Figure 1 (a), we can divide the primitive skills into two categories: ***motion-based*** and ***gripper-based*** skills.

Motion-based Skills. Motion-based skills are designed to describe each movement of robot arms. To clearly characterize their behaviors beyond language, we equip each motion-based skill with corresponding spatial information extracted from teleoperation records in RH20T dataset. These spatial information can be represented in multiple forms, such as reaching a destination or following a trajectory. In contrast, primitive skills defined in VILA [27] and SayCan [24] lack precise spatial information, which may confuse low-level controllers when executing ambiguous primitive skills like "move forward". Moreover, this deficiency often leads to inconsistency in granularity when executing subsequent primitive skills, *e.g.*, a succeeding "pick" might necessitate an extra long movement to compensate for previous imprecise positioning, diverting the focus of low-level controllers from interacting with diverse objects.

In our design, we define a set of hierarchical motion-based skills. Among them, *move* is the most foundational and versatile motion-based skill, encapsulating all types of motion. Building on *move*, we further define more specialized motion-based skills, such as *pull* and *press*, with complex and context-specific semantics in various scenarios. It allows for tailored motion planning for different motion-based skills, *e.g.*, pushing the object following a certain direction or moving the robot arm along a specific trajectory. Also, more proficient controllers can be assigned to execute them. Moreover, with such hierarchical definition, we can also easily expand more specialized semantics based on basic *move* to tackle more challenging and novel tasks in the future.

Gripper-based Skills. Gripper-based skills are intuitive. They include operations related to the gripper, such as *pick*, which requires first opening the gripper and then closing.

Primitive Skill Template. To interact with different objects and attributes more flexibly, we design a primitive skill template, *e.g.*, "*pick* the {attribute} {object}". The placeholders in templates are completed during annotation, serving as the ground truth in RH20T-P. The task planners in CGAs can perceive generalizable object categories and attributes in templates with their broad knowledge. Such design can avoid situations like SayCan [24] where defining all skill-object composition as primitive skills exhaustively.

All Primitive Skills. Besides motion-based and gripper-based skills, we also define *done*/*reset* to indicate the task has been completed and the robot arm is required to be reset. All primitive skills in RH20T-P are listed in Figure 3 (a).

3.3 Hindsight Primitive-level Annotation

We first ask the annotators to watch complete episode and segment each episode into video clips. As shown in Figure 3 (b), each of video clip is annotated with start frame and end frame, as well as corresponding primitive skills. The placeholders for objects and attributes in the templates are annotated based on the video. We then use the teleoperation records (7-DoF parameters) in RH20T dataset to generate the various forms of primitive-level spatial information (*i.e.*, destination, direction, trajectory) for motion-based skills. Besides, we use GPT-4V [13] to caption each episode, providing

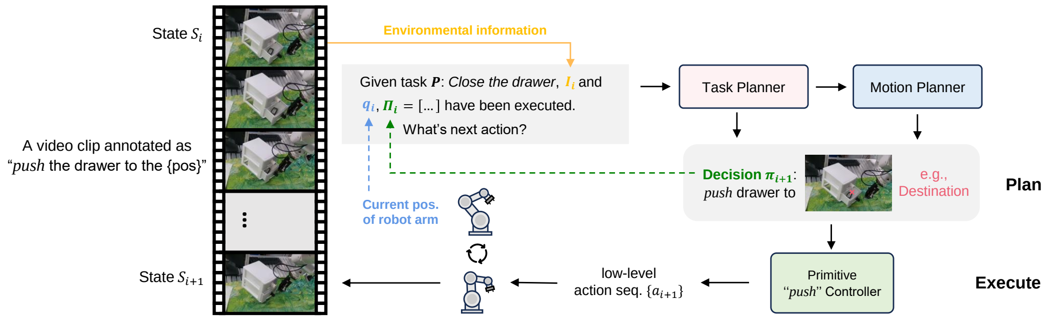

State S_i
Environmental information
Given task P: Close the drawer, I_i and q_i, Π_i = [...] have been executed. What's next action?
Task Planner
Motion Planner
A video clip annotated as "push the drawer to the {pos}"
Decision π_{i+1}: push drawer to
e.g., Destination
Plan
Current pos. of robot arm
State S_{i+1}
low-level action seq. $\{a_{i+1}\}$
Primitive "push" Controller
Execute

Figure 4: **Plan-execute CGA paradigm.**

detailed descriptions for each scene. Most of the hallucinations in these captions are removed after human inspection. We also include these captions in RH20T-P.

4 Plan-execute CGA Paradigm

We standardize a plan-execute CGA paradigm, with two planners for task decomposition and motion planning, as well as primitive-level controllers for subsequent execution.

4.1 Overall Pipeline

As shown in Figure 4, given a language description $\boldsymbol{P}$, we first define a state $\boldsymbol{S}_i$, where the agent has completed a part of task from initial state $\boldsymbol{S}_0$. Based on historical decisions $\boldsymbol{\Pi}_i = \{\boldsymbol{\pi}_0, \cdots, \boldsymbol{\pi}_i\}$ and observations under $\boldsymbol{S}_i$, the task planner in RA-P predict the next primitive-level decision $\boldsymbol{\pi}_{i+1} = \text{TaskPlanner}(\boldsymbol{P}, \boldsymbol{\Pi}_i, \boldsymbol{I}_i, \boldsymbol{q}_i)$. Here, we collect visual features $\boldsymbol{I}_i$ from cameras and the position of the robot arm $\boldsymbol{q}_i$ as the input observation. If output decision $\boldsymbol{\pi}_{i+1}$ belongs to motion-based skills, we further utilize motion planner to predict the corresponding primitive-level spatial information $\boldsymbol{m}_{i+1} = \text{MotionPlanner}(\boldsymbol{\pi}_{i+1}, \boldsymbol{I}_i)$. Next, the low-level controller maps the decision $\boldsymbol{\pi}_{i+1}$ and spatial information $\boldsymbol{m}_{i+1}$ to action sequences $\{\boldsymbol{a}_{i+1}\} = \text{Controller}(\boldsymbol{\pi}_i, \boldsymbol{m}_{i+1}, \boldsymbol{I}_i)$. These action sequences are performed by the robot arm to transition from $\boldsymbol{S}_i$ to the next state $\boldsymbol{S}_{i+1}$. Then a new round of planning is conducted under $\boldsymbol{S}_{i+1}$. In this manner, planner and controller continue to work alternately until planner gives a *done* decision, indicating the task has been completed.

In practice, we associate each transition $\boldsymbol{S}_i \rightarrow \boldsymbol{S}_{i+1}$ with a corresponding video clip from RH20T-P during training, based on the assumption that the part of task before $\boldsymbol{S}_i$ has been successfully executed. In each clip, we collect RGB images and position of the robot arm from the initial frame as the input observations $(\boldsymbol{I}_i, \boldsymbol{q}_i)$, expecting the task planner to make decisions $\boldsymbol{\pi}_{i+1}$ that are consistent with the primitive skill annotated in the clip. The 7-DoF control information recorded in the clips can be regarded as the action sequences $\{\boldsymbol{a}_{i+1}\}$ predicted by the low-level controller. During the inference stage, we follow the above paradigm to sequentially conduct planning and execution step by step.

4.2 RA-P: A Baseline Implementation

We implement a baseline CGA, *i.e.*, **RA-P**, on RH20T-P.

Task Planner. We employ LLaVA[4] [15] as task planner in RA-P. To fine-tune the language model, we also generate an instruction-following dataset with robotic manipulation knowledge based on RH20T-P. Note that using other VLMs as task planner is feasible, *e.g.*, applying GPT-4V via ICL.

Motion Planner. Various forms of spatial information in RH20T-P offer a broader range of options for motion planner. We use destination (x, y, d) of the trajectory as spatial information $\boldsymbol{m}_i$ for simplicity, where x, y denote the pixel coordinates in the image, and d denotes the depth relative to the camera. Here, we employ a Deformable DETR[5] [41] as a simple motion planner to localize next destination (x, y, d) that the robot arm should move to. Inspired by [42–44], we introduce a special

[4]https://github.com/haotian-liu/LLaVA, Apache-2.0 license.
[5]https://github.com/fundamentalvision/Deformable-DETR, Apache-2.0 license.

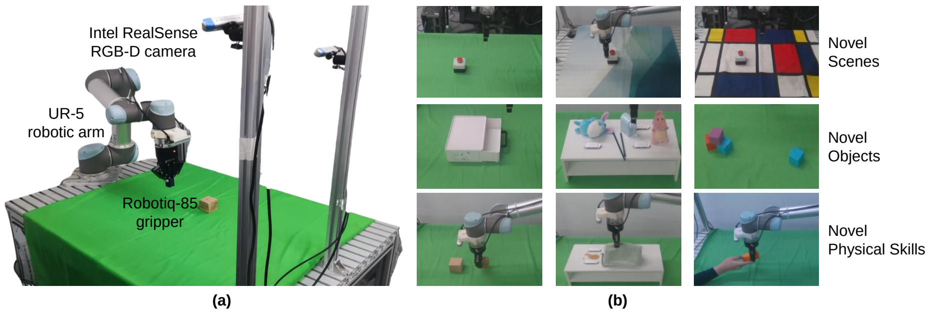


Figure 5: **(a)** Evaluation platform. **(b)** Evaluation on generalization of three levels in robotic manipulation tasks.

token <pos> to VLM vocabulary. Once the <pos> is included in the prediction of task planner (*e.g.*, "*move* on top of the block <pos>"), the motion planner will be activated. We then add the hidden features of the token <pos> with semantics related to objects and its spatial information to the object queries in DETR so that the DETR can localize the relevant destination. Finally, we convert (x, y, d) into a 3D point in real world with camera calibration for subsequent execution.

Low-level Controller. We apply two types of controllers, *i.e.*, hard-code and policy-based controller. Primitive skills like *move* and *open* can be executed directly or based on motion planning. We develop a set of hard codes to generate 7-DoF control parameters. For motion-based skills, we interpolate a trajectory from current position of robot arm to the predicted 3D point from motion planner, and use hard code to move robot arm along this trajectory. And for primitive skills that interacting with diverse objects (*i.e.*, *pick*, *push*, *pull* and *press*), we individually train primitive-level ACTs [29] as policy-based controllers. We use 7-DoF control sequences of corresponding clips to train each policy-based controller.

Training and Inference Details. All parameters in LLaVA, except for vision encoder CLIP [40], are fine-tuned. Motion planner, *i.e.*, DETR, in RA-P is jointly trained with LLaVA. We train the RA-P on 8 NVIDIA A100 GPU. Besides, to adapt DETR to new environment characterized by different sensors during inference, we collect an small dataset from evaluation environment to fine-tuning DETR alone after joint-training while keep VLM frozen. During inference, we deploy task planner, motion planner and controllers in RA-P on a NVIDIA A100 GPU, and develop a communication module between RA-P and robot arms. The whole inference pipeline operates as depicted in Section 4.1.

5 Experiments

5.1 Experimental Setup

Evaluation Platform. As shown in Figure 5 (a), we employ a UR-5 robotic arm with a parallel Robotiq-85 gripper for interaction. An Intel RealSense RGB-D camera is positioned in front of robot to capture environmental information.

Evaluation setup. We select 8 out-of-distribution novel tasks to evaluate the generalizability of RA-P on three levels:

1. **Novel scenes**: tasks are evaluated in unseen scenes, building with different background and tableclothes, including *Pick Object* and *Press Button*.
2. **Novel objects & compositions**: tasks with unseen objects or unseen compositions (consisting of seen skills and seen objects) are tested, including *Take Down Object* and *Close Drawer*.
3. **Novel skills**: tasks with unseen skills are tested, including *Wipe Table*, *Throw Garbage*, *Stack Blocks* and *Receive Object*.

Note that these levels are incremental, suggesting that the test for level 3 also includes the test for levels 1 and 2. Besides, ***the 8 tasks in the evaluation and the novel scenes/objects/skills appeared in the evaluation are not present in the training distribution of RA-P.***

Table 2: **Evaluation of novel tasks on three levels (10 trials).** "Plan" denotes planning accuracy and "Exec." denotes the execution success rate of whole system (including task planning and motion planning if exists). "w/ FT." and "wo/ FT" denote ACT with/without fine-tuning on evaluation tasks. Note that our RA-P have not seen evaluation tasks during training.

	Novel Scenes		Novel Objects & Compositions		Novel Skills	
	Plan (%)	Exec. (%)	Plan (%)	Exec. (%)	Plan (%)	Exec. (%)
ACT (wo/ FT.)	-	10	-	5	-	5
ACT (w/ FT.)	-	40	-	25	-	15
GPT-4V	**100**	35	**95**	17.5	**95**	12.5
RA-P (ours)	**100**	**80**	85	**70**	87.5	**67.5**

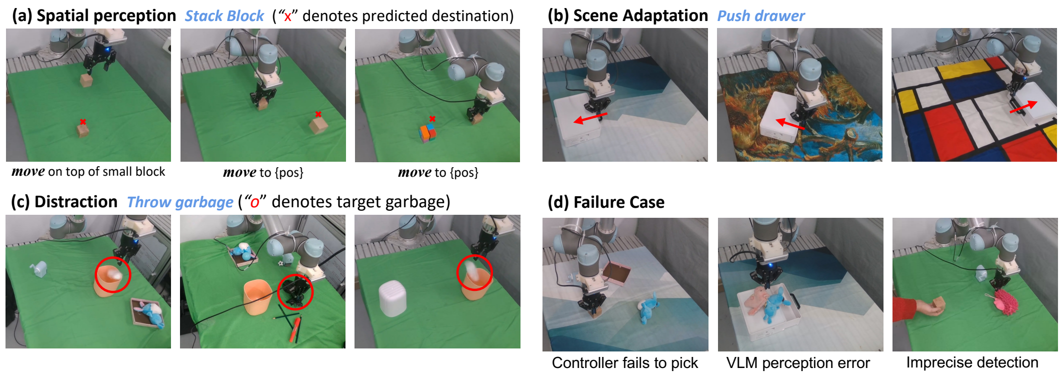

(a) Spatial perception Stack Block ("x" denotes predicted destination)
move on top of small block
move to {pos}
move to {pos}
(b) Scene Adaptation Push drawer
(c) Distraction Throw garbage ("o" denotes target garbage)
(d) Failure Case
Controller fails to pick
VLM perception error
Imprecise detection

Figure 6: **Visualization of RA-P executing the tasks.**

Comparison Counterparts. We choose ACT [29] as comparison for imitation learning. We provide two ACT baselines, *i.e.*, the first is pre-trained on the entire RH20T-P dataset, and another additionally collects datasets of each evaluation tasks and individually trains 8 ACTs based on pre-trained weights following RH20T [30]. We also introduce an agent that uses GPT-4V for both task planning and motion planning, representing agents [27, 28] solely relying on VLMs for composable generalization due to the lack of primitive-level spatial knowledge. We use ICL to guide GPT-4V in selecting primitive skills in RH20T-P and predicting destination for each motion-based skill. Due to difficulties in injecting external knowledge, GPT-4V predicts the 2D coordinate as a compromise and constructs the (x, y, d) triplet through directly using the depth information of that coordinate in the image.

Metric. We conduct 10 trials for each task to measure ***execution success rate***, assessing whether the whole RA-P system can accomplish the given tasks. To evaluate the performance of task planner, we record planning outputs during execution and manually examine whether task planner correctly generates primitives and objects (***planning accuracy***). Note that execution success rates contain evaluations of both task planning and motion planning. And we leave the assessment of motion planning in execution phase for potential discrepancies, *i.e.*, a seemingly feasible destination in planning stage may lead to failure in low-level execution.

5.2 Experimental Results

The results are shown in Table 2 and Figure 5 (b).

Comparison to Imitation-based Methods. ACT exhibits consistently poor performance, even in the basic tasks like *Pick Object*. We find the success rate of ACT decreases when the initial position of the robot arm is far from the target objects. In contrast, by decoupling motion planning from subsequent execution, the primitive-level controller in RA-P can perform picking operation near the target object, resulting in a significant performance improvement. Besides, these basic tasks often serve as components of more complex tasks such as *Stack Block*. The potential of agents that delegate motion planning to low-level controllers will be limited by the poor outcome of low-level controller.

Comparison to Agents relying on VLMs for motion planning. With larger-scale VLM, agents built by GPT-4V achieves a higher planning accuracy. However, obtaining reliable spatial information

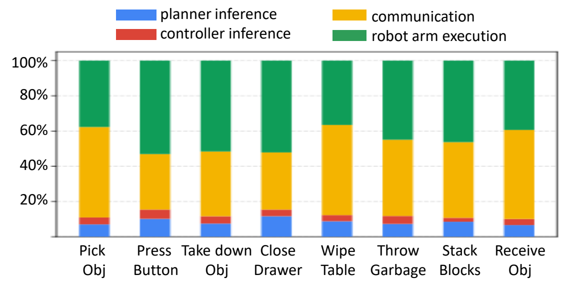


Figure 7: **Inference time of deployed RA-P.**

Table 3: **Data scaling during fine-tuning.** "Plan" and "Dest." denote planning accuracy and mean recall of predicted destination (threshold $\leq$ 5cm/10cm), respectively.

VLMs	Data	Plan	Dest. (5cm/10cm)
RA-P (LLaVA-7B + DETR)	-	0.45	0.10/0.31
	20%	0.71	0.26/0.60
	50%	0.79	0.38/0.72
	100%	0.86	**0.48/0.78**
GPT-4V	-	**0.94**	0.02/0.30

from GPT-4V through ICL poses great challenges, resulting in a huge disparity in the execution success rate of GPT-4V. In contrast, our RA-P achieves a higher execution success rate across all three levels through composable generalization, especially for novel skills, which can be attributed to the well-designed primitive skills and corresponding spatial information in RH20T-P.

5.3 Qualitative Analysis and Discussion

Visualization. As shown in Figure 6, we provide an illustration of our RA-P executing some of tasks in Table 2. More video demonstrations of RA-P are provided on an anonymous webpage (https://sites.google.com/view/rh20t-p/main).

Robustness on Object Distractions. As shown in Figure 6 (c), we place an object classified as garbage in a pile of unrelated objects and ask RA-P to conduct the "Throw garbage" task. RA-P can distinguish the target object from surroundings based on the observations and successfully execute the task, validating the robustness to object distractions.

Failure Case. As shown in Figure 6 (d), failures primarily include in low-level controllers (such as failing to pick a target object), DETR localization deviation (especially in scenarios with distractions), and perception error of VLM, which leads to subsequent incorrect positioning.

Data Scaling. To explore the impact of data scaling during fine-tuning, we construct an simple online benchmark without execution. The results are shown in Table 3. Both task planning and motion planning are significantly improved compared to baseline. They are far from saturated, leaving potential room for method design and data accumulation.

Inference Time. Inference time of deployed RA-P is shown in Figure 7. Using a 7B language model as a decision-making backend for inference is acceptable in terms of time ($\sim$ 8%). There still leaves room for larger-scale language models. We will continue to optimize the efficiency of the entire pipeline through asynchronous communication.

6 Conclusion

In this work, we introduce RH20T-P, a dataset designed for primitive-level robotic manipulation that features meticulously defined primitive skills and diverse primitive-level spatial knowledge of multiple forms. We believe RH20T-P can facilitate the CGA applications in robotics, especially in acquiring novel skills. We also present experimental demonstrations based on proposed plan-execute CGA paradigm that the agent built on RH20T-P showcases feasibility and robust generalization in real-world robotic manipulation tasks.

Limitation. While RH20T-P serves as a pioneering primitive-level robotic manipulation dataset for real-world CGA applications, empirical studies indicate the great potential for further data accumulation. By scaling primitive-level dataset, we anticipate advancements in research on composable generalization, significantly expanding generalization capabilities in robotic learning. Additionally, task-planner (LLaVA-7B) and motion planner (DETR) currently used in RA-P face constraints due to computing resources. In the future, we will explore more sophisticated planning system like [28, 50] and robust motion planning strategies based on direction [25, 26] or trajectories [3, 5] in CGAs.

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,” *Robotics: Science and Systems (RSS)*, 2023.

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, “RT-2: Vision-language-action models transfer web knowledge to robotic control,” *arXiv preprint arXiv:2307.15818*, 2023.

[3] J. Gu, S. Kirmani, P. Wohlhart, Y. Lu, M. Arenas, K. Rao, W. Yu, C. Fu, K. Gopalakrishnan, Z. Xu *et al.*, “RT-Trajectory: Robotic task generalization via hindsight trajectory sketches,” *arXiv preprint arXiv:2311.01977*, 2023.

[4] M. Shridhar, L. Manuelli, and D. Fox, “Cliport: What and where pathways for robotic manipulation,” in *Conference on Robot Learning*. PMLR, 2022, pp. 894–906.

[5] W. Zhi, K. Liu, T. Zhang, and J. Matthew, “Learning Orbitally Stable Systems for Diagrammatic Teaching,” in *CoRL 2023 Workshop on Learning Effective Abstractions for Planning (LEAP)*.

[6] A. Escontrela, A. Adeniji, W. Yan, A. Jain, X. Peng, K. Goldberg, Y. Lee, D. Hafner, and P. Abbeel, “Video prediction models as rewards for reinforcement learning,” in *Advances in Neural Information Processing Systems*, 2024.

[7] J. Luo, P. Dong, J. Wu, A. Kumar, X. Geng, S. Levine, “Action-quantized offline reinforcement learning for robotic skill learning,” in *Conference on Robot Learning (CoRL)*, PMLR, 2023, pp. 1348–1361.

[8] N. Hansen, Y. Lin, H. Su, X. Wang, V. Kumar, A. Rajeswaran, “Modem: Accelerating visual model-based reinforcement learning with demonstrations,” *arXiv preprint arXiv:2212.05698*, 2022.

[9] A. Adeniji, A. Xie, C. Sferrazza, Y. Seo, S. James, P. Abbeel, “Language reward modulation for pretraining reinforcement learning,” *arXiv preprint arXiv:2308.12270*, 2023.

[10] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, W. Chen, “What Makes Good In-Context Examples for GPT-3?,” *arXiv preprint arXiv:2101.06804*, 2021.

[11] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.

[12] OpenAI, “GPT-4 technical report,” 2023.

[13] OpenAI, “GPT-4V(ision) System Card,” 2023 .

[14] H. Touvron, T. Lavril, G. Izacard, M. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, “Llama: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.

[15] H. Liu, C. Li, Q. Wu, Y. Lee, “Visual instruction tuning,” in *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[16] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence, “PaLM-E: An embodied multimodal language model,” 2023.

[17] R. Xu, X. Wang, T. Wang, Y. Chen, J. Pang, D. Lin, “Pointllm: Empowering large language models to understand point clouds,” *arXiv preprint arXiv:2308.16911*, 2023.

[18] Z. Chen, Z. Wang, Z. Wang, H. Liu, Z. Yin, S. Liu, L. Sheng, W. Ouyang, Y. Qiao, J. Shao, “Octavius: Mitigating task interference in mllms via moe,” in *International Conference on Learning Representations*, 2024.

[19] W. Dai, J. Li, D. Li, A. Meng, J. Zhao, W. Wang, B. Li, P. Fung *et al.*, “InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning,” *arXiv preprint arXiv:2305.06500*, 2023

[20] K. Chen, Z. Zhang, W. Zeng, R. Zhang, F. Zhu, R. Zhao, “Shikra: Unleashing Multimodal LLM’s Referential Dialogue Magic,” *arXiv preprint arXiv: 2306.15195*, 2023.

[21] Z. Peng, W. Wang, L. Dong, Y. Hao, S. Huang, S. Ma, F. Wei, “Kosmos-2: Grounding Multimodal Large Language Models to the World,” *arXiv preprint arXiv:2306.14824*, 2023.

[22] Z. Yin, J. Wang, J. Cao, Z. Shi, D. Liu, M. Li, X. Huang, Z. Wang, L. Sheng, L. Bai, J. Shao, W. Ouyang, “LAMM: Language-Assisted Multi-Modal Instruction-Tuning Dataset, Framework, and Benchmark,” in *Advances in Neural Information Processing Systems*, vol. 36, 2023, pp. 26650–26685.

[23] Y. Hong, H. Zhen, P. Chen, S. Zheng, Y. Du, Z. Chen, C. Gan, “3D-LLM: Injecting the 3d world into large language models,” in *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[24] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, “Do as i can, not as i say: Grounding language in robotic affordances,” *arXiv preprint arXiv:2204.01691*, 2022.

[25] 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.

[26] X. Li, M. Zhang, Y. Geng, H. Geng, Y. Long, Y. Shen, R. Zhang, J. Liu, H. Dong, “Manipllm: Embodied multimodal large language model for object- centric robotic manipulation,” *arXiv preprint arXiv:2312.16217*, 2023.

[27] Y. Hu, F. Lin, T. Zhang, L. Yi, Y.Gao, “Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning,” *arXiv preprint arXiv:2311.17842*, 2023.

[28] N. Wake, A. Kanehira, K. Sasabuchi, J. Takamatsu, K. Ikeuchi, “Gpt-4v (ision) for robotics: Multimodal task planning from human demonstration,” *arXiv preprint arXiv:2311.12015*, 2023.

[29] T. Zhao, V. Kumar, S. Levine, C. Finn, “Learning fine-grained bimanual manipulation with low-cost hardware,” *arXiv preprint arXiv:2304. 13705*, 2023.

[30] H. Fang, H. Fang, Z. Tang, J. Liu, J. Wang, H. Zhu, C. Lu, “Rh20t: A robotic dataset for learning diverse skills in one-shot,” *arXiv preprint arXiv:2307.00595*, 2023.

[31] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay *et al.*, “Roboturk: A crowdsourcing platform for robotic skill learning through imitation,” in *Conference on Robot Learning (CoRL)*, PMLR, 2018, pp. 879–893.

[32] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, C. Finn, “Robonet: Large-scale multi-robot learning,” *arXiv preprint arXiv:1910.11215*, 2019.

[33] P. Sharma, L. Mohan, L. Pinto, A. Gupta, “Multiple interactions made easy (mime): Large scale demonstrations data for imitation,” in *Conference on Robot Learning (CoRL)*, PMLR, 2018, pp. 906–915.

[34] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, P. Florence, “Interactive language: Talking to robots in real time,” *IEEE Robotics and Automation Letters*, 2023.

[35] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, “BC-Z: Zero-shot task generalization with robotic imitation learning,” in *Conference on Robot Learning (CoRL)*, 2021, pp. 991–1002.

[36] S. Nair, E. Mitchell, K. Chen, S. Savarese, C. Finn *et al.*, “Learning language-conditioned robot behavior from offline data and crowd-sourced annotation,” in *Conference on Robot Learning*. PMLR, 2022, pp. 1303–1315.

[37] 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,” in *Robotics: Science and Systems (RSS) XVIII*, 2022.

[38] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan *et al.*, “Open x-embodiment: Robotic learning datasets and rt-x models,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[39] C. Lynch, P. Sermanet, “Language conditioned imitation learning over unstructured data,” arXiv preprint arXiv:2005.07648, 2020.

[40] A. Radford, J. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, “Learning transferable visual models from natural language supervision,” *International Conference on Machine Learning (ICML)*, 2021, pp. 8748–8763.

[41] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, J. Dai, “Deformable detr: Deformable transformers for end-to-end object detection,” *arXiv preprint arXiv: 2010.04159*, 2020.

[42] Z. Dai, B. Cai, Y. Lin, J. Chen, “Up-detr: Unsupervised pre-training for object detection with transformers,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 1601–1610.

[43] Z. Chen, G. Huang, W. Li, J. Teng, K. Wang, J. Shao, C. Loy, L. Sheng, “Siamese detr,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 15722–15731.

[44] Y. Zang, W. Li, J. Han, K. Zhou, C. Loy, “Contextual Object Detection with Multimodal Large Language Models,” *arXiv preprint arXiv:2305.18279*, 2023.

[45] L. Chen, J. Li, X. Dong, P. Zhang, C. He, J. Wang, F. Zhao, D. Lin, “Sharegpt4v: Improving large multi-modal models with better captions,” *arXiv preprint arXiv:2311.12793*, 2023.

[46] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. Zitnick, “Microsoft coco: Common objects in context,” in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 740–755.

[47] Y. Qin, E. Zhou, Q. Liu, Z. Yin, L. Sheng, R. Zhang, Y. Qiao, J. Shao, “Mp5: A multi-modal open-ended embodied system in minecraft via active perception,” *arXiv preprint arXiv:2312.07472*, 2023.

[48] Y. Du, S. Yang, B. Dai, H. Dai, O. Nachum, J. Tenenbaum, D. Schuurmans, P. Abbeel, “Learning universal policies via text-guided video generation,” in *Advances in Neural Information Processing Systems*, vol. 36, 2024.

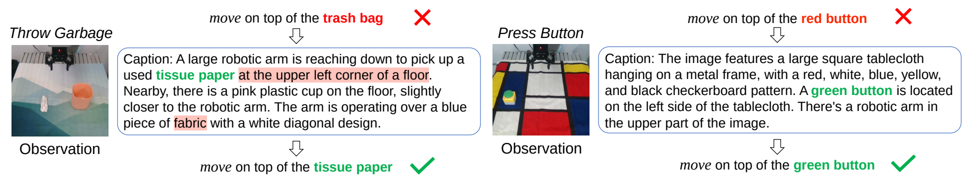
[49] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. Srirama, K.Mohan, Y. Chen, K. Ellis *et al.*, “Droid: A large-scale in-the-wild robot manipulation dataset,” *arXiv preprint arXiv:2403.12945*, 2024.

[50] A. Ajay, S. Han, Y. Du, S. Li, A. Gupta, T. Jaakkola, J. Tenenbaum, L. Kaelbling, A. Srivastava, P. Agrawal, “Compositional Foundation Models for Hierarchical Planning,” in *Advances in Neural Information Processing Systems*, 2024.

[51] J. Li, D. Li, S. Savarese, S. Hoi , “Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models,” *arXiv preprint arXiv:2301.12597*, 2023.

Appendix

A More Details about RA-P


Throw Garbage
move on top of the trash bag
Observation
move on top of the tissue paper
Press Button
move on top of the red button
Observation
move on top of the green button

Caption: A large robotic arm is reaching down to pick up a used tissue paper at the upper left corner of a floor. Nearby, there is a pink plastic cup on the floor, slightly closer to the robotic arm. The arm is operating over a blue piece of fabric with a white diagonal design.

Caption: The image features a large square tablecloth hanging on a metal frame, with a red, white, blue, yellow, and black checkerboard pattern. A green button is located on the left side of the tablecloth. There's a robotic arm in the upper part of the image.

Figure 8: **Chain-of-Thought inference.**

`messages` = `[{"role":"system", "content": f"""` A chat between a curious user and an artificial intelligence assistant. The assistant is required to guide a robotic arm to perform a complex and comprehensive robotic task. Based on the current observation, completed actions, and current position of the robotic arm, you should give the next action for the robotic arm.

The given completed actions and the decision for the next command are composed of the following optional actions: `[`

- *move* to the `{pos}`
- *move* `{on top of | in front of | ...}` the `{object}`
- *push / pull* the `{object}` to the `{pos}`
- *pick / place* the `{object}`
- *open / close* the gripper
- *press* the `{object}` `{pos}`
- *rotate* `{clockwise | counterclockwise}` `{angle}` `{pos}`
- *done / reset*

Here `{object}` and `{pos}` are the templates for the value that needs to be predicted. The position of the robotic arm is represented as [x, y, d] with floating numbers, where x and y denote the coordinates of the robotic arm range from 0 to 1, and d denotes the depth of the robot arm in the observation. `"""}]`

Table 4: The system prompt for robotic tasks in RA-P.

Chain-of-Thought Inference. We find some perceptual errors in classifying object categories and attributes during decision-making, due to the limited distribution of objects in RH20T [30] and VLM scale (7B) we used in RA-P. Consequently, we propose to use the VLM to first describe the scene and then make decisions based on the generated descriptions. We refer it as to Chain-of-Thought inference (CoT inference). No extra adjustments are required during the training phase; instead, we can directly add the descriptions generated by VLM to the prompts for inference. Note that we only caption the scene before the initial decision, and all subsequent decisions within the same task rely on the same description, thereby increasing a minor overhead to the inference time of planning. As illustrated in the Figure 8, CoT inference can effectively reduce some perceptual errors thanks to scene descriptions. Given the scale of model we used in RA-P (7B), there is still room for improvement in captioning to assist VLM with subsequent decision-making, especially in terms of hallucinations included in the descriptions (texts marked with a red background in Figure 8) and scenarios with multiple object interferences. Besides, GPT-4V [13] can be used as an option to describe the scene for CoT inference.

Prompts and Instructions used in RA-P. System prompt in RA-P is listed in Table 4. Instructions for robotic tasks in RA-P is listed in Table 5. Here, `{task_desc}`, `{historical_decisions}` and `{robot_arm_pos}` denote language specifications like "Pick Blocks", historical decisions made by task planners before current state and the position of the robot arm, respectively. During training, we randomly select one of the instructions in the conversations.

- To accomplish the task `{task_desc}`, the following actions have been sequentially completed: `{historical_decisions}`. Based on the observation and scene depiction (`{scene_desc}`), the current position of the robotic arm is `{robot_arm_pos}`. What should be the next action of the robotic arm?
- Given the embodied task `{task_desc}`, the subsequent steps have been undertaken: `{historical_decisions}`. Referring to the observation and scene description (`{scene_desc}`), the present location of the robotic arm is indicated by `{robot_arm_pos}`. What would be the appropriate next action for the robotic arm?
- With the assigned task `{task_desc}`, the following actions have been accomplished: `{historical_decisions}`. As shown in the observation and the description (`{scene_desc}`), the current position of the robotic arm is `{robot_arm_pos}`. Considering this, what are the next advisable action for the robotic arm?
- In light of the assigned task `{task_desc}` and corresponding description (`{scene_desc}`), the ensuing steps were carried out: `{historical_decisions}`. Observing the image given, the robotic armś current position is marked as `{robot_arm_pos}`. In light of this, what is the recommended next step for the robotic arm?
- For the purpose of achieving the goal of `{task_desc}`, we have progressively executed these steps: `{historical_decisions}`. As indicated by the observation and scene depiction (`{scene_desc}`), the robotic armś current location is `{robot_arm_pos}`. What would be the advisable subsequent action for the robotic arm?
- To achieve the goal of `{task_desc}`, we have progressively executed these steps: `{historical_decisions}`. As indicated by the observation and scene depiction (`{scene_desc}`), the robotic armś current location is `{robot_arm_pos}`. What would be the advisable subsequent action for the robotic arm?
- To reach the objective of `{task_desc}`, we have methodically performed the following actions: `{historical_decisions}`. The observation shows that the robotic arm is currently positioned at `{robot_arm_pos}`. Given the scene description (`{scene_desc}`), what should be the next action for the robotic arm?
- Aimed at accomplishing the goal of `{task_desc}`, we have systematically undertaken these steps: `{historical_decisions}`. As depicted in the supplied image, the location of the robotic arm is currently `{robot_arm_pos}`. Considering this, what would be the prudent next step for the robotic arm?
- With the given task `{task_desc}`, a sequence of actions has been diligently executed: `{historical_decisions}`. The current position of the robotic arm, as shown in the image, is `{robot_arm_pos}`. Based on the scene depiction (`{scene_desc}`), what would be the logical next action for the robotic arm?
- Considering the ongoing `{task_desc}`, our approach has involved a series of progressive steps: `{historical_decisions}`. The image provided points out the current position of the robotic arm at `{robot_arm_pos}`. The scene description is `{scene_desc}`. What is the next recommended action for the robotic arm in this scenario?

Table 5: The list of instructions used for robotic tasks in RA-P.

Execution Deployment. As shown in Figure 9, we deploy LLaVA, Deformable DETR [41] and low-level ACT [29] controllers on a NVIDIA A100 GPU, and develop a communication module between RA-P and robot arm. We conduct the following procedures to perform the plan-execute paradigm during the evaluation stage:

1. **Collecting observation:** the sensor in the evaluation platform captures the observation information in the environment, and then transmits the RGB images along with the current position of the robot arm to the agent, which is deployed on an A100 GPU, through the communication module.
2. **Decision making:** the task planner will take images and position them as input and make decisions using predefined primitive skills. If any motion-based skill is chosen, the motion planner will be invoked to predict the precise coordinate of the destination (x, y, d) where the robot arm should move to. The coordinate will be transformed into a 3D coordinate with camera calibration.

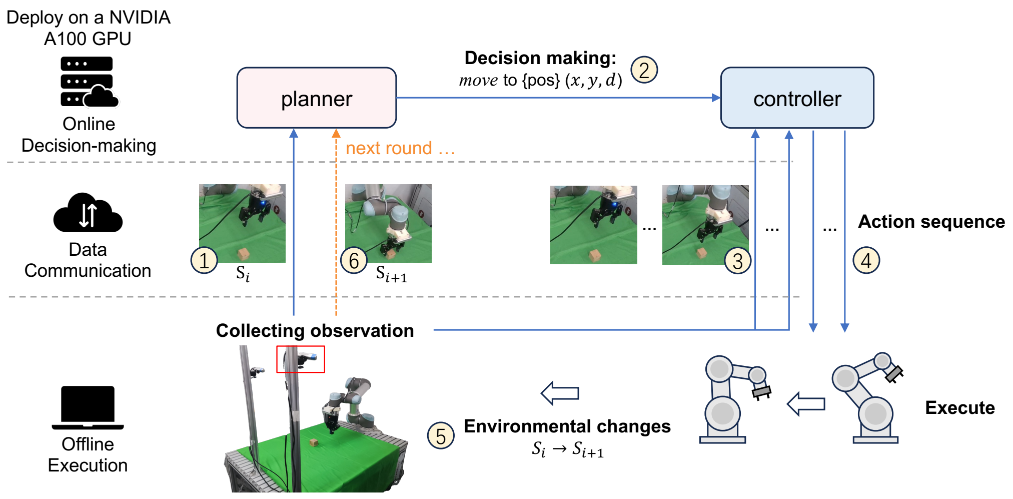


Figure 9: **Deployment pipeline of RA-P.**

3. **Mapping to action sequence and then executing:** based on the type of primitive skills, a specific low-level controller will be called to map the decision and coordinate to the action sequence. For policy-based controllers, we predict action sequences based on current observations for the next 5 steps. After receiving and executing the 5-step action sequence, the robot arm collects information again and transmits it to controllers, repeating until the controllers give a terminate signal. For hard-code controllers, we interpolate a straight line between the starting position of the robot arm and the predicted destination to obtain a movement trajectory, then use hard code to move the robot arm along the trajectory as the action sequence. The robot arm then executes the action sequence, omitting the multiple data transfers and communications like those in policy-based controllers.
4. **Iterative plan-execute process:** Once the controllers complete the decision made by the planners, we will return to step 1 to start a new round of the plan-execute process until the planner ultimately gives a *done* decision.

We also provide a analysis on inference time in Section 5.3.

B GPT-4V Execution Setup

System Prompts for agents with GPT-4V during Execution Phase. We evaluate agents with GPT-4V through in-context learning. Detailed system prompt is shown in Table 6.

C More Results

Cumulative Success Rates. We show the cumulative success rates for different steps of several tasks in Figure 10. It is observed that the majority of task failures in agents with GPT-4V are attributed to insufficient localization abilities. The positions predicted by GPT-4V, which are far from the target object, result in the failure of the low-level controller's execution. In contrast, our RA-P trained on RH20T-P can provide more reasonable spatial priors, resulting in a higher execution success rate.

More Qualitative Results. More video demonstrations of RA-P are provided on a webpage (https://sites.google.com/view/rh20t-p/main). We also give an illustration to display the complete execution process in Figure 11.

D All Tasks in RH20T-P

We provide the list of tasks in RH20T-P in Table 7.

`messages = [{"role":"system", "content":  f"""` You are placed inside an embodiment environment with a robot arm and a gripper with it. Given a goal (task description) in language and multi-frame RGB images depicting the scene and objects, you are required to decompose the goal into sub-tasks step by step to ensure it can be accomplished eventually.

At each round of conversation, I will give you:

- multi-frame RGB images, time from past to present
- task description in language
- historical decisions already made
- The relative coordinates (x, y, d) in the current image where the gripper is located. The x-axis is from left to right, and the y-axis is from top to bottom, with a range of values between 0 and 1. For example, the top left corner of the image is (0,0), and the top right corner is (1,0).

Your reply should be one of the primitive skills, as follows:

- *move* to the `{pos}`
- *move* `{on top of | in front of | ...}` the `{object}`
- *push / pull* the `{object}` to the `{pos}`
- *pick / place* the `{object}`
- *open / close* the gripper
- *press* the `{object}` `{pos}`
- *rotate* `{clockwise | counterclockwise}` `{angle}` `{pos}`
- *done / reset*

Remember to replace the `{pos}` in primitive skills with relative coordinates (x, y) as the position in the image.

I will then give an example to help you follow the context: ...
`"""}]`

Table 6: The system prompt for Agents with GPT-4V in Execution Phase.

E Potential Social Impact

The proposed RH20T-P dataset and RA-P model demonstrate effectiveness and generalization in robotic tasks, especially in novel physical skills, which can benefit the future development of CGAs. The violent elements (*e.g.*, using a knife) in the dataset and related knowledge learned by the robot may have some potential negative social impacts. However, considering that our data source, RH20T, has already been publicly released, these impacts are controllable.

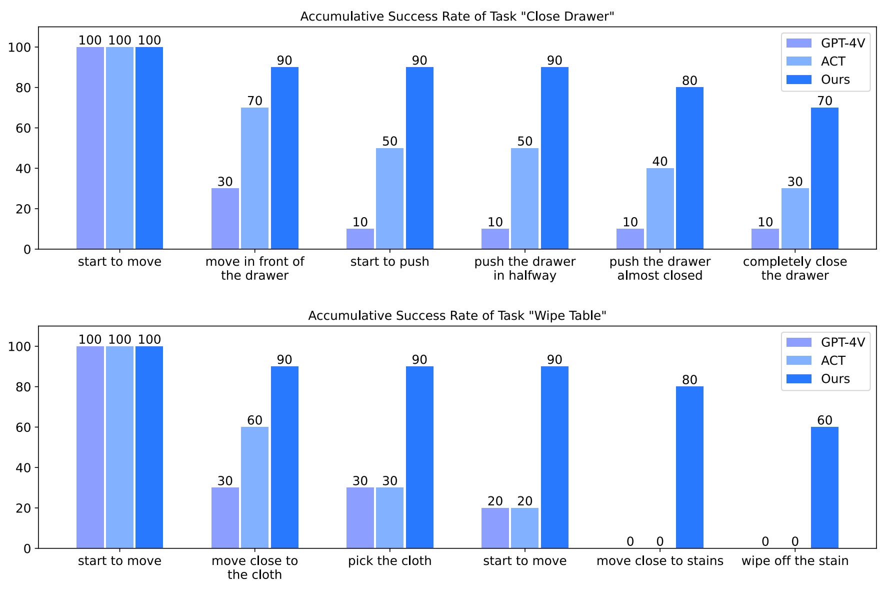


Figure 10: **Cumulative success rates for different stages of several tasks.**

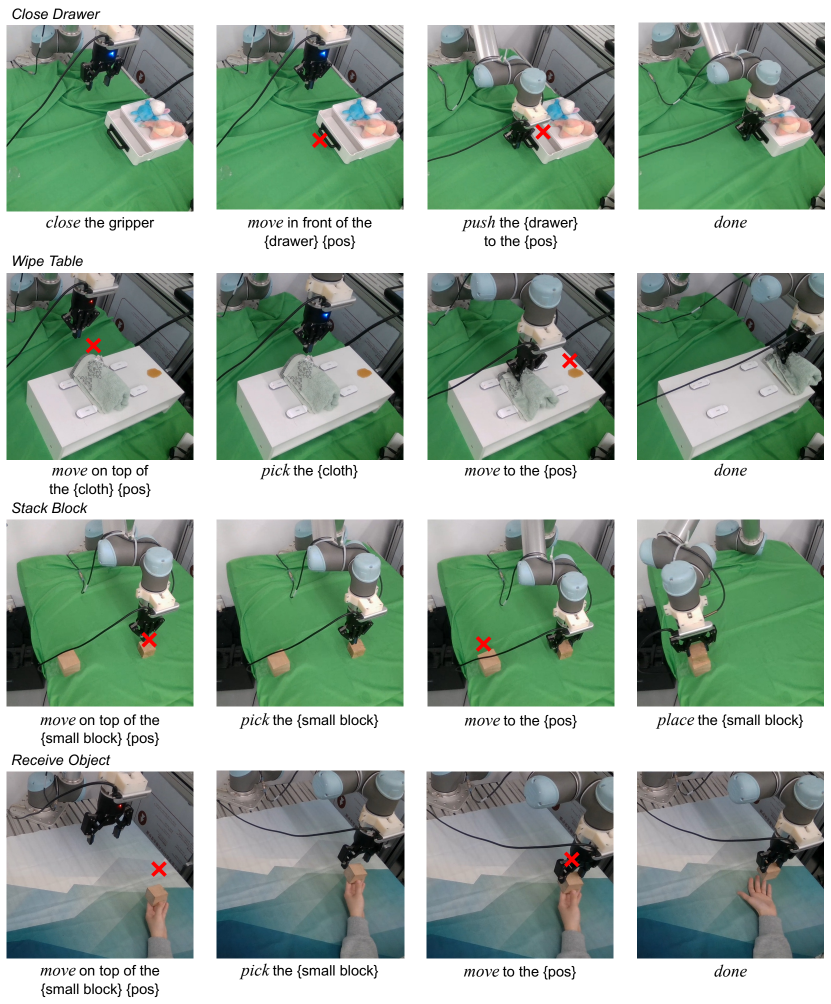

Close Drawer
close the gripper
move in front of the {drawer} {pos}
push the {drawer} to the {pos}
done
Wipe Table
move on top of the {cloth} {pos}
pick the {cloth}
move to the {pos}
done
Stack Block
move on top of the {small block} {pos}
pick the {small block}
move to the {pos}
place the {small block}
Receive Object
move on top of the {small block} {pos}
pick the {small block}
move to the {pos}
done

Figure 11: **Complete execution process of RA-P during evaluation.**

- Turn off the desk lamp
- Take the dish off the dish rack
- Turn the knob to increase the volume of the speaker
- Placing a piece on the chessboard to complete the setup.
- Clean the table with a cloth
- Throw the garbage
- Take the pencil out from the pencil sharpener
- Pull out a napkin
- Plug in the power cord of the desk lamp, turn on the socket, and light up the desk lamp
- Press three buttons from left to right in sequence
- Grab the block and place it at the designated location
- Turn the hands of a clock
- Remove the object from the scale
- Play the first move as black on the 3-4 points in the upper right corner of the Go board
- Grasp the handle and open the drawer
- Remove the bubble ring from the assembled bubble ring and ball
- Pick up one small block
- Take the photo frame down from the bracket
- Put the object on the shelf
- Pick up and place an object with obstacles
- Put the knife on the cutting board
- Press the button
- Approach and touch the side of the small block
- Hold a block with the gripper and sweep it from left to right
- Close the drawer
- Take everything out of the gift box
- Press a button from top to bottom with obstacles
- Open the microwave door
- Turn on the water tap
- Place the handset of the telephone on the corresponding phone cradle
- Play the drum
- Turn on the desk lamp by pressing the button
- Turn on the power strip by pressing the button
- Put the cup on the cup rack
- Open a sliding window
- Move an object from one box to another
- Cover the pot with the lid
- Push an object with obstacles
- Use the gripper to push the small block from left to right
- Pick up the cup
- Wipe the tabletop with a sponge
- Take the object down from the shelf
- Push down the lever
- Turn off the water tap

Table 7: The list of tasks in RH20T-P.