VLABench: A Large-Scale Benchmark for Language-Conditioned Robotics Manipulation with Long-Horizon Reasoning Tasks

Supplementary Material

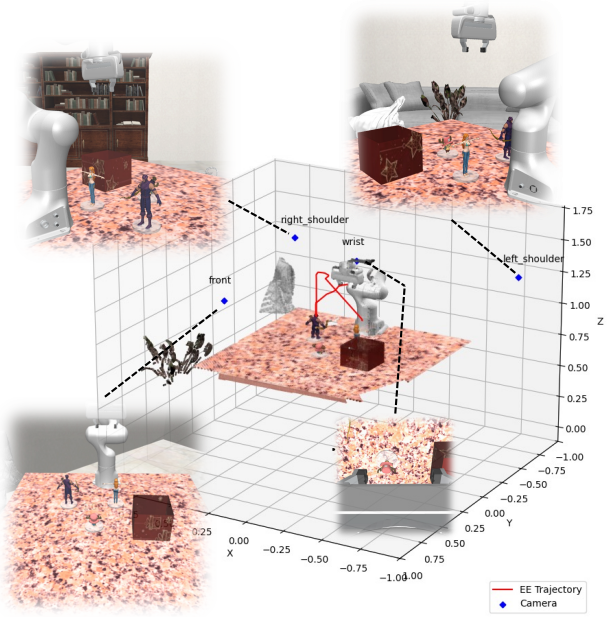


Figure 6. **Task observations.** The figure illustrates an example data instance from the *Select Toy* task, including multi-camera positions, multi-view RGB images, 3D point clouds, and the expert trajectory.

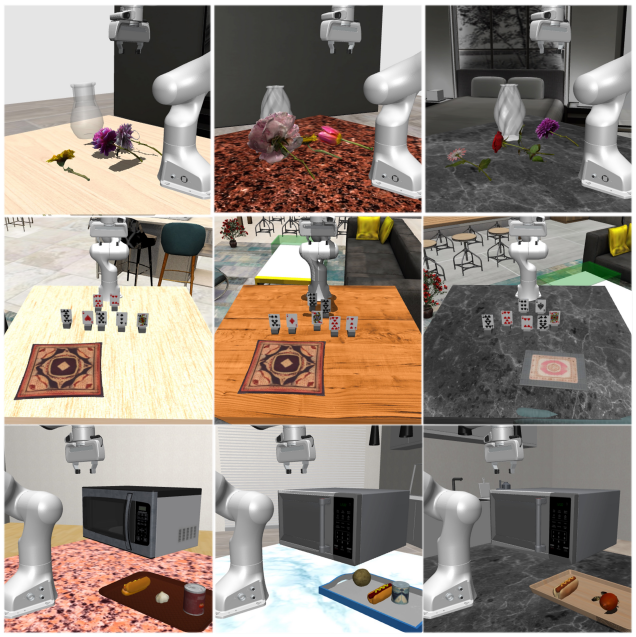


Figure 7. **Examples of task variants in different episodes. Row 1**: *Insert Flower* task from the left shoulder view. **Row 2**: *Play Texas Hold'em* task from the front view. **Row 3**: *Heat Food with Microwave* task from the right shoulder view. Examples in the same row originate from the same task but differ in task objectives, distracting objects, spatial configurations, spatial poses, etc.

6. Benchmark Implementation

6.1. Task Descriptions

All Tasks. VLABench includes both 60 primitive and 40 composite tasks. These tasks encompass a wide range of manipulation skills and involve many high-level capabilities. The skills include 1) Pick&place, 2) Open&close door, 3) Open&close drawer, 4) Hang objects on the wall, 5) Use tool e.g. Hammer nail, 6) Press button, 7) Insert, 8) Pour, 9) Twist, and 10) Explore. For higher-level intelligence, VLABench's evaluation dimensions encompass complex scene understanding, implicit semantic analysis, world knowledge transfer, understanding of physical laws, relative spatial perception, long-term task planning, and even multi-step logical reasoning. Table 11 provides a detailed introduction to the 100 tasks involved in VLABench, including the type of each task, the manipulation skills involved, the scope of high-level intelligence examined, the average episode length at a control frequency of 10Hz, as well as a detailed description of the task and an explanation of its challenges. For the sake of clarity in the table, we will use abbreviations to represent the various intelligence dimensions. **M&T** corresponds to *Mesh & Texture Understanding*, **SP** corresponds to *Spatial Understanding*, **C&W** corresponds to *Common Sense & World Knowledge*, **SEM** corresponds to *Semantic Conversation Understanding*, **PHY** corresponds to *Physical Laws Understanding*, and **L&R** corresponds to *(Logistic) Reasoning*.

Long-horizon Design with Multistep Reasoning. Compared to previous benchmarks, VLAbench places more emphasis on comprehensive long-term reasoning. The reasoning defined here includes associating world knowledge with visual mesh or texture information to solve tasks, understanding latent task requirements through emotional language interpretation, mapping spatial descriptions to target states, subtask planning for multi-step operations, logical understanding, calculations, and result derivation, among others. Figure 8 presents a detailed comparison of the average episode length of overall tasks. VLABench exhibits the longest horizon among both Primitive and Composite tasks, surpassing RoboCasa Atomic and RoboCasa Composite by 27.0% and 35.1%, respectively. Further-

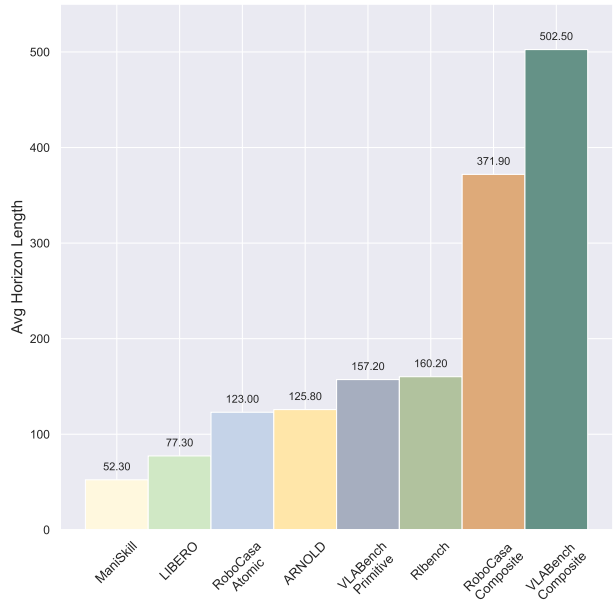


Figure 8. **Comparison of nomalized horizon length.** VLABench significantly surpasses other benchmarks in terms of average task length.

more, VLABench demonstrates significantly greater multi-step reasoning depth compared to other task sets, including sub-task numbers, inferring users' hidden semantics, integrating visual and commonsense information, spatial reasoning, and even logical reasoning, as exemplified by solving math problems.

Episode Diversity with Task Variants. In VLABench, a task represents a broad category of activities designed around specific assets and the actions an agent performs. These tasks are centered on object-related themes and require diverse visual information, relevant common-sense knowledge, and rich semantic input from the user. To ensure variability, each rollout introduces different target objects and receptacles, creating unique task instances.
To more clearly illustrate the diversity of episodes within the same task, we introduce task variants. More specifically, these task variants include different instances of the same category of objects(O), different object categories (C), task-irrelevant distractions (D), diverse language instructions (L), different spatial layout (S), and diverse object textures and backgrounds (V). Unlike previous benchmarks [38] where episodes with different task variants were treated as separate tasks, VLABench unifies such variations under a single task category. Figure 7 illustrates examples of task variants and episodes.

Grasp Obj	N-Cate	N-Obj	Recep	N-Cate	N-Obj
Billiards	2	24	Billiards Table	1	1
Books	8	52	Shelf	1	10
Baked Goods	5	60	Microwave	1	5
Condiment	5	50	Cabinet	1	5
Dessert	4	58	Tray	1	15
Drink	9	130	Fridge	1	5
Flower	9	25	Vase	1	10
Fruit	11	227	Box Container	2	10
Ingredient	16	181	Cutting Board	1	15
Mahjong	1	38	Counter	1	20
Number Cube	1	10	Safe	1	5
Painting	1	286	Stove	1	5
Poker	1	54	Table	1	20
Snack	8	97	Juicer	1	3
Flatware	4	80	Crockery	7	136
Tool	9	49	Coffee Machine	1	5
Toy	35	140	Placemat	1	10
Chemistry Solution	1	30	Tube Container	1	1
Name Tag	1	30	Flask	1	5

Table 3. **Assets statistics**. N-cate denotes the total number of object categories, while N-obj represents the total count of object instances. This table lists most of the assets.

6.2. Task Observation

Each task in VLABench supports multi-view RGB-D images, semantic segmentation images, and point cloud inputs. Figure 6 illustrates an example, showcasing the visualized point cloud data along with images from multiple viewpoints. Similar to general standard RLDS format datasets [50], each demonstration in VLABench not only includes the aforementioned multi-view RGB-D images and point clouds but also comprises: a list of language instructions, episode terminal, sparse reward, actions, full observations including joint positions, joint velocities, end effector position and orientation, grasping state, etc.

6.3. Task Difficulty Level

We further quantify the task difficulty level in Table 4 based on the task variant number(N_{tv}), subtask number(N_{sub}), and reasoning step number(N_{rs}).
Task Variant Score. The task variant is introduced in section 6.1. The difficulty score introduced by quantifying task variants is as follows:

$$Score_{tv} = N_{tv}/TV_{max} \tag{3}$$

where TV_{max} is set to 6 after comparing with multiple benchmarks.
Sub-step Number Score. Although we computed the normalized horizon length as shown in Figure 8, it does not fully reflect the long-term nature of the task due to differences in data collection. Therefore, we introduce sub-steps

Benchmark	TV↑	N_{sub}↑	N_{rs}↑	DL↑
RLBench	*D,O,S*	2.31	0	28.86
Calvin	*S, V*	5.0	0	40.0
SimplerEnv	*D,S,V*	2.11	0	27.66
Libero	*D,S*	2.16	0	22.96
VLABench	***C,D,L,O,S,V***	**3.66**	**1.78**	**75.96**

Table 4. **Benchmark difficulty level comparison.** VLABench significantly outperforms other benchmarks in difficulty level.

as a quantification metric, where each sub-step represents an atomic skill mentioned in section 6.1, such as pick, open, and pour. We computed the average number of sub-steps across all tasks in the language-conditioned manipulation benchmarks and defined the difficulty score as follows:

$$Score_{sub} = N_{sub}/SUB_{max} \tag{4}$$

where the SUB_{max} is set to 5 by default.

Reasoning Step Score. Similar to concepts in the LLM field, we define any behavior that requires intelligent reasoning as a reasoning step. This includes visual information with common sense, logical reasoning, semantic extraction, etc. However, VLABench is the first benchmark to introduce the concept of reasoning into the manipulation domain. Previous benchmark tasks mostly directly describe the task goals and requirements, resulting in a reasoning step count of zero. The difficulty score of reasoning steps can be represented as:

$$Score_{rs} = N_{rs}/RS_{max} \tag{5}$$

where RS_{max} is set to the current maximum number of reasoning steps, which is 3.

Difficulty Level. The difficulty level is computed as the sum of the three normalized scores:

$$DL = \alpha Score_{tv} + \beta Score_{sub} + \gamma Score_{rs} \tag{6}$$

where $\alpha = 0.3$, $\beta = 0.3$, $\gamma = 0.4$ in default setting, considering the importance of the intelligence of VLAs.

6.4. Domain Randomization

To ensure task diversity and broad data distribution, each task in VLABench incorporates multiple domain randomization techniques. These diversifications include:

- **Mesh&Texture Randomization.** This refers to the random variation of different instances within the same object category. For example, if a task scene requires an apple, the apple's mesh is randomly selected from a pool of 20 distinct instances.
- **Position&Orientation Randomization.** The default values for this randomized attribute are set as follows: the position offset is a random value within the range $[-0.05, 0.05]$ along the x and y directions, and the orientation is randomized with the yaw angle in the range $[-\pi/10, \pi/10]$.
  In certain tasks, including *SelectFruit*, grid sampling is employed for the random distribution of scene objects. Objects are constrained to be distributed within a grid space based on a maximum distance limit and are further subjected to the aforementioned basic pose offset.
- **Mesh Scale Randomization.** For the same mesh, VLABench scales the size of objects within a reasonable range, with the default scaling range set to $[0.95, 1.05]$.
- **Visual Disturbance.** VLABench employs random transformations of scenes and their relative positions, along with texture randomization of elements such as desks, floors, and walls, to achieve robust visual perturbations. In addition to the aforementioned color space transformations, the lighting intensity is randomly augmented within the range of $[0.8, 1.2]$.
- **Random Distractors.** VLABench requires different approaches to interpret scenes and extract key visual information accurately. To further enhance the robustness of task settings, we introduced the option to add irrelevant distractor objects to the tasks. For example, in the *SelectToy* task, 1–2 fruits can be included as visual distractors.

Figure 9. **Diverse scenes.** VLABench supports a wide range of different scenes.

7. Simulation and Framework

7.1. Simulator

VLABench is built based on Mujoco[58] and its control suite dm_control[59]. We selected Mujoco as the core simulation platform for our benchmark due to its lightweight design, high performance, and exceptional physical realism. These advances enable convenient, rapid evaluation of diverse algorithms. The VLABench framework is highly modular, meaning various object entities can be flexibly combined to create large-scale and diverse tasks and scenarios.

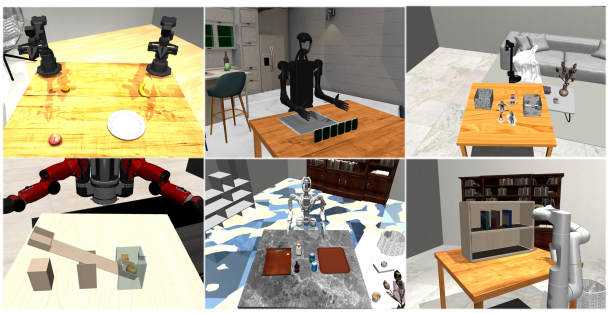


Figure 10. **Cross embodiment.** VLABench supports a wide range of different embodiments.

7.2. Scenes

To ensure diverse task environments and rich visual inputs, we curated over 20 distinct scene types, drawing inspiration from real-life contexts and task-specific backgrounds. These scenes span everyday household settings, such as kitchens, living rooms, and dining areas, and dynamic social scenarios, including shopping malls, supermarkets, chemistry laboratories, and medical rooms. Figure 9 highlights a small part of these carefully designed scenes. Beyond the variety of scene types and structures, we incorporated over 20 unique material textures for floors and walls, further enriching the visual complexity and enhancing the overall data diversity.

7.3. Cross Embodiment

In the standard evaluation process, VLABench employs a 7-DoF Franka Emika Panda manipulator equipped with a parallel gripper. We represent the position and orientation of the robot's end-effector in Euclidean space $\mathbb{R}^3$ using 3D coordinates for position and quaternions for orientation. Using inverse kinematics, we then resolve these end-effector poses into the corresponding rotational angles for the seven joints.

To ensure versatility and broad applicability, VLABench supports various embodiments, including multiple models of single-arm and dual-arm robots, humanoid robots, quadrupedal robots equipped with end-effectors, and mobile robots. Figure 10 illustrates the performance of these different embodiments within VLABench.

7.4. Assets

To meet the requirements of diverse tasks and capability assessments, we built an asset library centered around multiple task themes. We inherited some annotated assets from Robocasa [47] and retrieved numerous 3D models from Objaverse[12]. For novel tasks, such as the series of tasks we created around the toy theme, we carefully gathered a variety of high-quality character models from online 3D model sites. These models were then converted to MJCF format using the obj2mjcf [64] tool. Similarly to previous work [32, 47], we expanded the dataset of common simple objects using generative AI models. Specifically, we utilized Tripo.aI's text-to-3D and image-to-3D features to construct additional 3D objects, and Runaway.ai to generate multiple material textures.

Assets are divided into two main categories: objects-to-grasp and receptacles. For objects-to-grasp, recommended grasping points need to be annotated and are represented in the XML file using sites with *class=grasppoint*. For receptacles, both bounding boxes and recommended placement points are required: the former is annotated using sites with *class=keypoint*, while the latter is represented with *class=placepoint*. In the coarse and large-scale pre-annotation process, we annotated grasp points on all objects-to-grasp with Graspnet [16, 17] and manually refined them as needed. For receptacles, we used SAM [31] to assist in annotating bounding boxes and assigning the placement point default above the bottom of the receptacles. Subsequent manual refinement and post-processing were applied after pre-annotation. Table 3 provides an overview of the rough categories and the corresponding number of assets.

7.5. Extensibility

VLABench benefits from its modular design, offering strong flexibility and scalability in task composition. In terms of the codebase design, VLABench primarily includes:

- `Entity` class: `Entity` is the base class of all the objects to interact within VLABench. It is one of the core components of the object-centric data collection framework. All `Entity` objects are divided into two categories: `GraspedObject` and `Receptacle`. If a user wishes to add new assets, they simply need to inherit from the base class and register them in the code. For new functionalities, users are required to develop them according to their specific needs.
- `Robot` class: `Robot` is the base class of all the robots mentioned in Section 7.3. The `Robot` class primarily includes controllers for various robot models and interfaces for retrieving robot states. The implemented base classes currently cover single-arm, dual-arm, mobile arm, and humanoid robots. Thanks to the solid base class design, VLABench supports the rapid integration of new robot models, in addition to the commonly used ones.
- `Condition` class: The `Condition` class is derived to handle all the conditions used to determine whether a task is complete, such as `Contain`, `On`, `Pour`, and so on. If users wish to extend a new task, the task's completion conditions can be defined by combining existing `Condition` classes.
- `Task` class: The `Task` defines various tasks, includ-

ing different combinations of `Entitys`, task completion conditions, the robot executing the task, and other necessary interfaces. Thanks to the modularity and plug-and-play nature of the system, new tasks can be easily extended.

Built on these core components and a vast assets library, VLABench offers strong scalability, laying the foundation for future large-scale simulations and datasets. A more detailed tutorial will be updated in the open-source code repository.

7.6. Compuation Resources

VLABench, built on Mujoco, is a lightweight simulation benchmark with very low computational resource requirements. As shown in Table 5, VLABench's computational resource consumption is very similar to that of previously widely used benchmarks, and its running speed is also within a comparable range. We took all the simulation evaluation experiments on a single Nvidia RTX 3090 GPU (CUDA 12.2), with 256GB RAM and 48 CPU cores. Notably, compared to Libero, VLABench requires the rendering of multiple higher-resolution images, which leads to longer evaluation times. This is primarily limited by the rendering efficiency of the underlying simulator.

Benchmark	VRAM(MB)	RAM(MB)	Resolution	FPS	Time(s)
RLBench[27]	93.68	2076.67	$5 \times 128 \times 128$	13.79	63.8
Simpler[35]	1531.22	1597.44	$1 \times 480/512 \times 640$	20.72	53.6
Libero[38]	1175.08	6193.15	$2 \times 256 \times 256$	15.22	36.3
VLABench	863.04	4408.93	$4 \times 480 \times 480$	12.97	73.1

Table 5. **Benchmark computational resources comparison.** VLABench requires low computational resources while providing higher-quality observations, comparable to those of previous benchmarks.

8. Dataset Building

8.1. Skill Library as Domain Specific Language

To facilitate task description and execution in robotic manipulation, we design a domain-specific language (DSL) tailored for our system. The DSL provides a structured and human-readable way to define manipulation skills, their parameters, and execution sequences. By abstracting low-level commands into high-level instructions, the DSL ensures clarity, modularity, and ease of interpretation for various tasks. The DSL consists of three primary components:

- **Skills.** Atomic manipulation operations such as *Pick*, *Place*, *Lift*, etc.
- **Parameters.** Arguments specify each skill's details, such as the target object, orientation, and gripper state.
- **Task Execute Sequence.** Sequential or hierarchical combination of skills to define a complete manipulation task.

8.2. Data Collection Progress

The automatic data collection process in VLABench is built upon the aforementioned DSL-encapsulated code. For each designed task, a corresponding task sequence is defined to represent the order of operations required to complete the task. For example, a case in *Select Fruit* requiring the robot to pick up an apple and place it in a basket can be expressed as a DSL sequence as follows. The parameters, such as grasp pose and target position, are dynamically generated based on the simulation environment, and prior annotation information.

```
Pick("Apple",
    {"gripper_state": "close",
    "orientation": [np.pi, 0, 0]})
Place("Basket",
    {"pose": [0.6, 0.4, 0.15],
     "gripper_state": "open"})
```

All tasks involve the execution of Skills using motion planning algorithms for trajectory generation. Notably, the execution of the Pick Skill requires the robotic arm to first move to a preparation position. During this process, we compute the overlap between the gripper's point cloud and the environment's point cloud along the trajectory from the preparation position to the grasping position. This overlap is used as a rejection sampling condition to determine an appropriate grasping direction.

8.3. Data Collection Method Analysis

Efficiency Comparison. To validate the reliability of our data collection framework, we conducted a set of comparative experiments with human experts performing teleoperations. We invited three data collectors to gather 100 samples for each of three tasks, using the same OSC controller as Libero[38]. Among them, the *Select Fruit* task represents a low-difficulty pick/place task, *Insert Flower* involves tasks with complex rotations, and *Take Chemistry Experiment* represents a complex long-horizon task, where any operational error results in task failure. For our data collection framework, we conducted three trials using a single process on the same Nvidia 4080 workstation, collecting 100 samples in each trial.

As shown in Table 6, our heuristic data automation method achieve a great tradeoff both on success rate and speed, and multiprocessing allows easy dataset scaling.

Method	Select Fruit		Insert Flower		Chemistry Experiment	
	SR	Time(s)	SR	Time(s)	SR	Time(s)
teleoperation	0.96±0.01	53.3±4.2	0.82±0.02	47.1±3.8	0.62±0.04	160.0±18.9
ours	0.76±0.02	36.5±1.2	0.39±0.02	27.4±0.9	0.47±0.03	84.0±3.2

Table 6. The result is the average of 3 trials, teleoperations are conducted by three operators on the same 4080 workstation

Failure Cases Analysis. Due to the high level of randomness in each episode, the success rate of the object-centric data collection approach is not 100%. We measured the success rate from primitive to composite tasks, which fluctuates between 25% and 95%. Moreover, as the tasks become longer and the number of sub-tasks increases, the success rate tends to decrease due to cumulative errors. This is particularly evident in the construction of composite task data. After visualizing and analyzing the failure cases, we categorized them into the following types:

- Episode initialization error. Due to the random placement of objects or different object instances during environment reset, collisions may occur in the initial state, preventing the task from being completed correctly.
- Slippage during the grasping process. This is due to the use of irregularly shaped objects and the gripper's tendency to clamp too tightly during grasping, which causes non-planar objects to potentially slip.
- Incorrect solution by inverse kinematics. Due to the randomness of the target object and the robot's state, the computed pose may result in collisions with the environment.

We further analyzed the success rates across different tasks and the factors correlated with task success, as shown in Figure 11. Through categorization of task characteristics, we found that tasks involving manipulation of objects with complex geometric structures typically exhibit the lowest success rates, as these objects are highly sensitive to collisions and inherently unstable. Additionally, as the number of subtasks increases, the overall success rates show a declining trend due to cumulative error propagation.

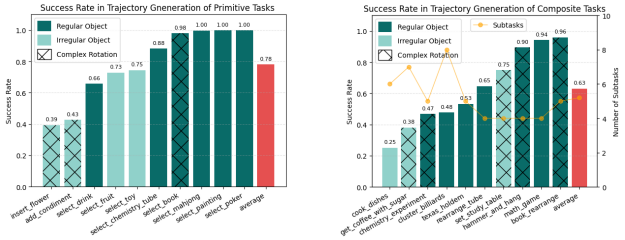


Figure 11. Success rates of data collection process.

8.4. Prompt for Interactive Instruction

We have generated a diverse set of instructions for VLABench's dataset and evaluation tasks. These linguistically rich instructions effectively assess the ability of different models to achieve a comprehensive understanding of task scenarios. All task types, including the five categories of Primitive tasks and Composite tasks, share the following system prompt. In the system prompt, **{object list}** and **{target objects}** should be replaced with the actual objects and target objects involved in each task's scenario. For example, in the *Insert Flower* task, the object list might be ["rose", "tulip", "sunflower"], while the target objects would be ["rose"]. For each data point or evaluation task, we require the generation of ten distinct instructions, all referring to the same target object but expressed in completely different ways.

System Prompt Template

I am going to make some task instructions for a robot arm. Here are some objects:**{object list}**. And the target entity is **{target objects}**.
The target entity is the object that the robotic arm is supposed to grasp, move, or perform other operations on. Our task requirements are related to the characteristics of the target object and should also reflect everyday needs for a specific item.

For tasks involving common sense and world knowledge, the prompt should additionally include the following description, emphasizing the unique characteristics of the target objects. An example for **{Task-Specific Descriptions and Emphases}** in *Select Toy with Common Sense* task is: "The target entity is the one that the robotic arm is supposed to grasp, move, or perform other operations on. Our task requirements are related to the characteristics of the target object. The instruction should focus on IP, rather than directly saying which toy to choose.". While the few-shot examples are:["target_object: Donald, instruction: 'I want a toy in the Disney series.'", "target_object: Goku, instruction: 'Pick a toy which belongs to the dragon ball.'"].

Common Sense Template

{Task-Specific Descriptions and Emphases.}
Please find the target entity's specific character which is different from other target objects and combine it into the instruction.
{Task-Specific Few-shot Examples.}
Please provide the task following the format of the above example. Please provide ten tasks that meet the above requirements and format.

For tasks requiring linguistically rich instructions, the prompt extends the system prompt by incorporating the following semantic prompt. The few-shot examples in *Select Toy Semantic* may be like: ["target_object: batman, instruction: 'I'm a big fan of DC series, please help me choose a suitable toy.'", "target_object: Luffy, instruction: 'Today is my friend's birthday, and I want to buy a Luffy figure for him. Could you help me wrap it? Thank you!'"].

Semantic Template

Please find the target entity's specific character which is different from other target objects and combine it into the instruction. Do not directly mention the target entity by name and avoid explicitly stating the need for the object. Instead, create tasks that reflect real-life scenarios where the need for the object is implied through casual, everyday observations. The task should suggest a need without saying it directly, focusing on natural, implied requests.
{Task-Specific Few-shot Examples.}
Please provide the task following the format of the above example.
The target_entity must be the target entity. Please provide ten tasks that meet the above requirements and format.

Composite tasks integrate the abilities and skills involved in primitive tasks, with each composite task featuring its unique scenario and context. In this setup, while the system prompt remains shared, each task is accompanied by a specific prompt. Here, we present the specific prompt for the *Cluster Book* task.

Composite Task Example: Cluster Book

The task now is to classify the books. Please design real-life scenarios where there is a need to categorize books and generate instruction based on the classification requirement.
You cannot specify the exact classification method; just create a realistic scenario that requires classification and instruct it to categorize the books in front of it.
Please make the generated instructions more diverse in terms of conversational language, tone, and scenarios. Avoid sticking to a single-sentence structure.
{Task-Specific Few-shot Examples.}
Please provide the task following the format of the above example.
Please provide ten tasks that meet the above requirements and format.

8.5. Data Quality and Diversity Analysis

Qualitative Analysis. To ensure collection stability, human experts carefully adjusted the operational sequence and key prior information such as grasp orientation, and bounding box. Then human experts reviewed the quality and diversity of visualized episodes rollouts by visualizing the rendered video and trajectory points in 3D space. Figure 12 visualized diverse trajectories from 10 different episodes. To ensure sufficient task information and clear resolution, we input images from multiple perspectives into GPT-4 and perform VQA tasks. We verified that all visual information can be recognized by GPT-4. We use the same data format in RoboCasa and Libero, both built upon Mujoco, which have demonstrated the usefulness of the collected data both in sim and real.

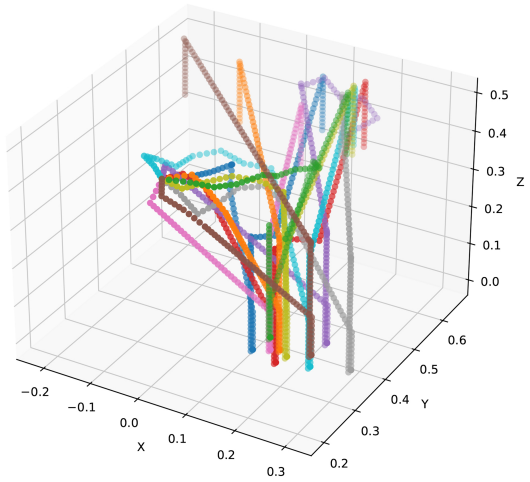


Figure 12. Visulization of 10 trajectories of same task.

Quantitative Analysis. Although our task setup is challenging, the model is still able to learn from this data. In Section 4.1, π_0 demonstrates a good success rate on the Select Painting task, and even on tasks with lower success rates, the progress score is still relatively promising. We also reviewed the videos of these policies interacting with the environment and found that they tend to learn the correct behavior patterns. However, due to insufficient accuracy and limited generalization ability, they fail to handle out-of-domain situations properly, leading to cumulative errors.

9. Experiment Implementation

9.1. VLA Training Details

To assess the generalization ability of various VLAs, we primarily fine-tune OpenVLA, Octo, RDT-1B and π_0 using our dataset. We utilize the original open-source code and adhere to the default hyperparameters set by the authors. For training OpenVLA and Octo, We converted the dataset into the same format as Libero and then transformed it into RLDS format for training. We use the hdf5 format dataset to train RDT-1B and the Lerobot format to train π_0 for 30k iterations, as same as the original settings. Given that OpenVLA has 7B parameters, we apply the recommended LoRA strategy in all experiments, rather than performing full parameter fine-tuning. In contrast, the other three models un-

dergo full parameter fine-tuning. We train all models until convergence is achieved. In particular, Octo exhibits a certain reluctance to converge, which might be attributed to its relatively low level of generality. Note that we adhere to the default configurations of these models: OpenVLA and Octo process a single-view image as input, whereas π_0 gets two views and RDT-1B utilizes three different views. All experiments are conducted on NVIDIA A800 with 80GB of memory.

We choose five primitive tasks as basic training tasks: {Select Fruit, Select Toy, Insert Flower, Add Condiment, Select Painting} for tracks 1-5, considering the difficulty of these tasks and the coverage of skills. In track 5, we evaluate the tasks on unseen but similar five tasks: {Select Poker, Select Mahjong, Select Billiards, Select Ingredient, Friction QA}. In track 6, we choose five composite tasks as training and evaluation tasks: {Texas Holdem, Play Math Game, Find Unseen Object, Cluster Toy, Hammer Nail then Hang Picture }.

9.2. Evaluation of Worksflows

In evaluating the foundation model-based workflow algorithms, we adopt the same evaluation process and metrics used for assessing the VLAs. The procedure for evaluating each task individually is outlined as follows:

1. **Run the Base Model Workflow.** Execute the base workflow in the specified environment and record the corresponding outputs, with particular emphasis on data related to the model's target entity detection information.
2. **Task Evaluation.** Once the relevant information has been collected, the success of the task and the accuracy of target identification are assessed. Specifically, correct identification of a target contributes 20% of the total score, while task success will award full points.
3. **Final Score Calculation.** After evaluating individual tasks several times, the scores for each time task are aggregated to yield the final score for the model under each configuration.

By applying this evaluation framework, we ensure a consistent and comprehensive assessment of the model's performance across different tasks and settings.

9.3. Evaluation of VLMs

9.3.1. Evaluation Pipeline

Non-interactive Evaluation. Figure 3 illustrates the simplified evaluation process specifically designed for VLMs in VLABench. Firstly the evaluation dataset is generated by initializing a series of task scenarios, each associated with two four-view diagrams: one annotated with masks and labels to identify distinct entity segments, and the other serving as a reference image without annotations, as shown in Data Production module in Figure 3. A randomly selected linguistic instruction from GPT4 relevant to the task accompanies these diagrams, forming the input to the Vision-Language Model (VLM).

During inference time, we provide a detailed description of the skill library, the requirements of output format, and several few-shot examples in different settings. These elements collectively form the system prompt for querying the VLM. The VLM is required to generate DSL output consisting of a sequence of skills, where each skill includes a name and associated parameters, conforming to predefined patterns to enable systematic evaluation.

Then, the generated skill sequences are constructed into a directed graph based on their logical dependencies. Subsequently, these DAGs are matched with the reference ones and scored in four metrics. Finally the scores are combined using weighted aggregation to calculate a total score for each model. Please refer to Section 9.3 in the supplementary material for more detailed metric computation.

Interactive Evaluation. Similar to the VLA and workflow evaluation process mentioned in previous sections, interactive evaluation computes a task progress score based on the interaction with the environment. VLABench provides a controller that parses the DSL action sequences output by the VLM into executable actions, which are then applied in a simulation environment to interact with real-world objects. This approach is one of the key metrics for evaluating robotic manipulation tasks. However, it is more time-consuming compared to non-interactive approaches, and its evaluation dimension is relatively limited, as it cannot distinguish between errors in skill selection and those in parameter generation.

9.3.2. Metrics

As discussed above, the entire evaluation process of VLMs can be simplified to DSL generation and the score can be computed through direct graph matching. The assessment of the skill sequences output by the VLM is based on the following four metrics. **Skill Recall Rate (SR).** We use SR as the coarsest-grained metric to evaluate the model's capability to identify and invoke the correct skills.

$$SR = \frac{|SL_{\text{gt}} \cap SL_{\text{pred}}|}{|SL_{\text{gt}}|} \tag{7}$$

where SL_{gt} represents the list of skills manually labeled for completing tasks, and SL_{pred} refers to the list of skills predicted by the model. The denominator corresponds to the total number of relevant skills in the dataset, while the numerator counts the intersection of the relevant skills and those correctly identified by the model.

Parameter Recall Rate (PR). The PR quantifies the model's ability to correctly identify the parameters associated with each skill. In many cases, each skill is contingent upon specific parameters, which are often represented by

the labels of relevant objects within an image. The PR thus measures the model's accuracy in recognizing and interpreting these parameters, a crucial aspect for ensuring the correct execution of the task. Accurate parameter identification is fundamental not only for skill invocation but also for the model's overall performance in real-world applications. A higher PR indicates a higher accuracy of the parameters predicted by the model, thus ensuring that the model correctly identifies the entities that need to be valued in the figure.

$$PR = \frac{|Param_{\text{gt}} \cap Param_{\text{pred}}|}{|Param_{\text{gt}}|} \tag{8}$$

where $Param_{\text{gt}}$ refers to the list of parameters manually labeled for each skill, and $Param_{\text{pred}}$ denotes the list of parameters predicted by the model.

Skill&Parameter Recall Rate (SPR). Unlike the individual metrics SR and PR, SPR requires the model to identify both the correct skills and the exact parameters associated with each skill. It provides a more comprehensive and strict evaluation of the model's ability of scene understanding and task planning in a real-world context. This metric is particularly useful in evaluating scenarios where both skills and their contextual parameters are critical for task execution, such as in visual recognition tasks where precise associations between actions and objects are necessary.

$$SPR = \frac{|SP\text{-}Pair_{\text{gt}} \cap SP\text{-}Pair_{\text{pred}}|}{|SP\text{-}Pair_{\text{gt}}|} \tag{9}$$

where $SP\text{-}Pair_{\text{gt}}$ represents the set of all manually labeled skill-parameter combinations, and $SP\text{-}Pair_{\text{pred}}$ refers to the corresponding combinations predicted by the model.

Precise Matching Rate (PM). In addition to evaluating the correctness of skill-parameter matching, PM places greater emphasis on assessing the logical dependencies of the skill sequence, particularly for tasks with strict temporal requirements. Instead of totally strict sequential order, this metric focuses on ensuring that the necessary dependencies are satisfied for successful task execution. For example, in *Make Juice* task, the model must ensure that the juicer is opened before adding fruit, but the order of adding apples versus oranges is irrelevant.

We begin by aggregating the skill sequence according to predefined operational patterns and constructing a directed acyclic graph (DAG) with a designated source node to represent the logical dependencies among operations. A match is defined as a node in the model-generated graph that shares the same skill name and parameters as a corresponding node in the ground-truth graph while also satisfying the logical dependency relationships, e.g. incoming and outgoing edges.The formula for this metric is as follows:

$$PM = \frac{|Node_{\text{matched}}|}{|Node_{\text{total}}|} \tag{10}$$

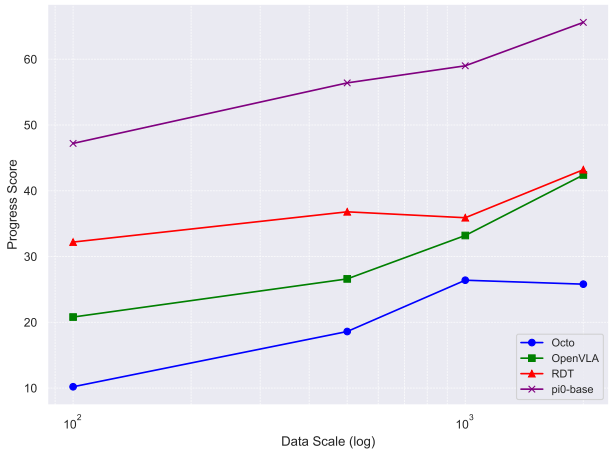


Figure 13. **Scaling trend.** This result is evaluated on *Select Painint* task with data scales of 100, 500, 1000, and 2000.

Model	From Scratch	From Pretrained
Octo	1.02	1.34
OpenVLA	3.02	11.74
RDT-1B	6.26	15.37

Table 7. Ablation of fine-tuning from scratch and pretrain. Evaluated on primitive tasks with seen objects.

Model	Avg PS
RDT-1B_open_step_64	15.37
RDT-1B_open_step_32	15.52
RDT-1B_cls	17.68

Table 8. Comparison of open-loop and closed-loop control for RDT-1B. Evaluated on primitive tasks with seen objects.

where the numerator $Node_{\text{matched}}$ represents the number of nodes in the model-generated DAG that match the corresponding nodes in the ground-truth DAG. $Node_{\text{total}}$ represents the total number of nodes in the ground-truth DAG.

Finally, these four scores are combined using predetermined weights to compute a total score for each model. The formula for this metric is as follows:

$$Score = w_1 \cdot SR + w_2 \cdot PR + w_3 \cdot SPR + w_4 \cdot PM \tag{11}$$

where w_1, w_2, w_3, w_4 are the weights of different metrics, with the constraint $w_1 + w_2 + w_3 + w_4 = 1$.

10. Detailed Analysis and Case Study

10.1. Ablations and Analysis for VLAs

Experimental results show that the current open-source VLAs perform poorly on our tasks. On one hand, this can be

attributed to the high difficulty of VLABench tasks, which impose stringent requirements on the generalization capabilities of the models. More importantly, the limitations and deficiencies in both the architecture and pretraining process of current VLAs make it challenging for them to adapt effectively to downstream tasks after large-scale pretraining, especially under fine-tuning scenarios with diverse data distributions. This stands in stark contrast to LLMs, which excel in adapting to downstream tasks with minimal fine-tuning on small datasets. To further illustrate the aforementioned issues, we conducted several ablation experiments:

- **Data Scaling.** More data implies a greater number of visual-language to trajectory mappings. For specific primitive tasks, we expanded the dataset to 2,000 samples and conducted separate evaluations on three models using datasets of varying scales: 100, 500, 1,000, and 2,000 samples. As illustrated in Figure 13, as the number of training samples increases, the performance of all models improves. In practice, this improvement is mainly reflected in higher action accuracy and more robust behavior. We also conducted scaling experiments on other tasks, but tasks that initially performed poorly, such as the Select Toy task, did not show significant improvements in success rate. This suggests that the performance differences brought by scaling vary across different tasks, and are related to task difficulty, skill requirements, and visual complexity. These findings are similar to those in [18, 47].
- **Pretrained Effect.** We also conducted an evaluation of models trained with the fine-tuning dataset from scratch, with the results summarized in Table 7. The findings indicate that models of this scale struggle to quickly adapt to downstream tasks with limited data. Moreover, models trained from scratch exhibit a significantly slower convergence rate compared to their pre-trained counterparts. It is reasonable to infer that pretraining on large-scale, domain-relevant data can significantly facilitate the faster transfer of VLAs to downstream tasks.
- **Closed-loop Effect.** Following the open-source RDT framework, the primary experiment employs a single-trajectory inference scheme with 64 trajectory points, implemented using open-loop control. However, open-loop control is prone to error accumulation. To address this, we conducted additional evaluations of RDT using closed-loop control. The results in Table 8 show that closed-loop control achieves slightly better performance compared to open-loop control. This suggests that the low success rate in task execution is primarily due to the inherent limitations of the model itself.

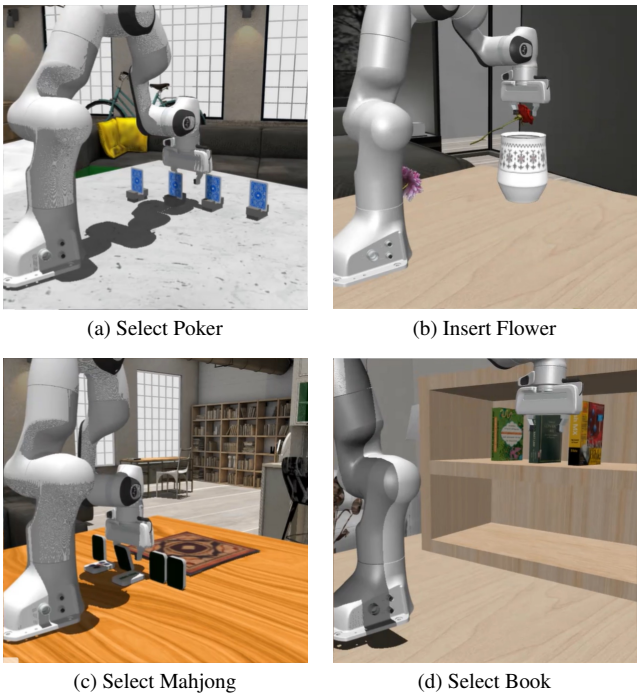
To analyze why these models perform poorly, we base our discussion on experimental results and observations from two key perspectives.

Limitation of Model Architecture.

- **Incomplete Information Intake**. Some shortcomings in the model architecture result in this issue. For instance, OpenVLA and Octo only process single images with a resolution of 224×224, which inherently puts them at a disadvantage when the input images contain occlusions or require finer texture details. Similarly, due to issues with perspective and low resolution, directly mapping visual information to precise spatial coordinate points becomes challenging. Comparing OpenVLA and π_0, both of which are based on VLM, we find that π_0 additionally takes the end effector state as input when decoding actions. This information and the output exhibit a strong linear relationship, which helps in more accurate behavior prediction.
- **Lack of Memory**. Current models only accept inputs representing the current state, lacking position embeddings to capture temporal sequences or tokens to represent historical actions. This limitation can cause the model's behavior to become stuck in certain states. This issue is particularly pronounced in long-horizon tasks, where the model may "forget" previous actions and repeatedly perform the same behavior.
- **Inherent Flaws of Different Architecture.** The VLAs we used primarily include two forms: transformer-based next-token prediction architectures and diffusion model-based architectures. The former, leveraging VLMs, benefits from pretraining on world knowledge but inherently suffers from precision loss due to the discretization required by action tokenization. On the other hand, diffusion policies are better suited for continuous spatial distributions, yet they lack visual and language pretraining. Additionally, diffusion models rely on multiple large encoders, such as T5, making it challenging to jointly fine-tune parameters during unified training. This limitation contributes to the poor performance of diffusion policies in VLABench tasks requiring common sense.

Shortcomings of Pretrain. VLA pretraining has been proven effective for efficient transfer to downstream tasks. However, the current pretraining approaches may have certain issues. For example, RT-2 [5] highlights a pretraining strategy that jointly trains on multiple text tasks and text-visual tasks to preserve the model's inherent language and reasoning capabilities, resulting in impressive generalization behaviors. Recent research [18] has confirmed that jointly training multimodal VQA tasks with trajectory data helps improve the model's generalization ability across different dimensions. In contrast, OpenVLA, which is also based on VLM, is pretrained solely on trajectory datasets. This likely leads to the degradation of VLM's original capabilities, such as commonsense knowledge and reasoning skills.

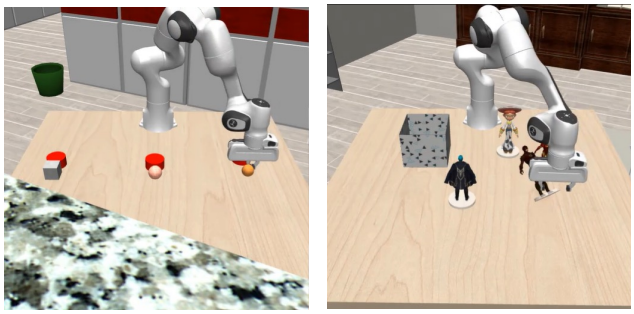
Additionally, constrained by the availability of datasets, the scale of current VLA pretraining data is far smaller



(a) Select Poker (b) Insert Flower

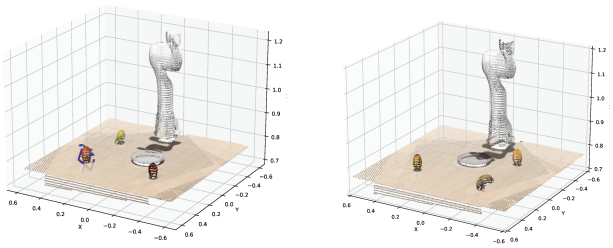
(c) Select Mahjong (d) Select Book

Figure 14. Failure case: failures caused by the inability of the algorithm to percept task scene and plan on rotation.



(a) Physics QA (b) Select Toy

Figure 15. Failure case: fails to perceive object interactions.



(a) Graspnet successes. (b) Graspnet fails.

Figure 16. Failure case: Graspnet fails in generating valid grasp points, resulting in task failing.

than that of language models. Drawing inspiration from the scaling laws and emergent behaviors observed in language models, there is likely a critical point and correlation between model parameter size and data volume. The scaling curve for VLA pretraining, however, remains an open topic for future research.

10.2. Simulation and Real-World Gap Analysis.

Real2sim gap. Since the models used in the experiments were pre-trained on real-world data, but fine-tuned on simulation data, the real-to-sim gap may have an impact on the experimental results. This could also be one of the reasons why these VLA models perform poorly on VLABench. Previous work [3, 30, 43] has shown that although these VLA models were pre-trained on real-world data, they still possess the capability to adapt to simulation tasks. Moreover, compared to previous SOTA methods, these fine-tuned VLAs achieve promising results on simulation tasks. However, Kim et al. [30] point out that the performance improvement of VLAs in simulation environments, compared to real-world environments, shows a reduction in the gap compared to other methods without robotics pretraining. A feasible approach is to mix a certain proportion of high-quality simulation data into the pre-training dataset, allowing the model to learn a broader range of data distributions. We leave the exploration of the performance effects of using the large-scale data generated by VLABench in the pre-training process for future work.

Sim2real gap. The sim2real gap has been extensively discussed and analyzed in previous work [35, 44, 47]. We adopt similar approaches to minimize the gap between real and simulated environments, such as domain randomization and using more realistic environments and 3D objects. Following the setting in Robocasa [47], we conducted basic real-world experiments. We chose the simple Select Mahjong task as the test task and collected 50 real-world data samples using teleoperation. During the experiment, we trained two π_0-base models: one using only 50 real data samples and the other using 50 real data samples combined with 200 simulation data samples. The success rates of these two models are 3/20 and 5/20, respectively. This confirms that the quality of our simulation data is sufficient for real-world deployment.

10.3. Further Analysis for Workflows

From the experimental results, we observe that while the framework algorithm based on the foundation model demonstrates some degree of robustness in handling complex semantic settings, the overall success rate and PS score remain relatively low. A comprehensive analysis of the failure cases reveals that the underlying issues can be broadly categorized into the following groups.

Perception. One of the primary challenges lies in the model's image and spatial perception capabilities. As Vox-

poser is implemented as a purely text-based framework, its perception module relies directly on the ground-truth labels of all items, which are provided as input for selection. While this leverages the comprehension and generalization capabilities of large language models to understand tasks, it exposes significant limitations in scenarios that require spatial perception and image-based reasoning. Specifically, Voxposer demonstrates clear incompetence in handling tasks involving spatial awareness or detailed image descriptions.

To address this, we augmented our experimental setup by incorporating an image perception module into Voxposer. Although this adjustment improved success rates on spatial perception tasks, the overall performance deteriorated due to errors introduced by the visual perception module. A similar issue was observed in CoPA, where the SoM family of models exhibited high sensitivity to segmentation parameters, requiring extensive tuning to achieve accurate entity recognition. Even with optimization, a substantial number of incorrect object recognition cases persisted, highlighting fundamental challenges in the perception component.

Planning. Another significant limitation emerges in the model's planning capabilities. After selecting the target object, the model's lack of spatial perception often prevents it from recognizing the need to adjust its pose, such as rotating the robotic arm when grasping certain objects. This deficiency leads to frequent task failures, particularly in scenarios involving objects like cardboard sheets or books, as illustrated in Figure 14. Additionally, the simple point-cloud-based center-of-mass grasping strategy employed by the model exhibits a high probability of failure when interacting with objects of complex shapes, such as toys, as shown in Figure 15.

For CoPA, similar challenges were encountered in the graspnet module, where planning grasping actions was hindered by its instability. In many instances, the module failed to identify a valid grasping point, resulting in task failures, as depicted in Figure 16. These issues underscore the model's inability to effectively plan and execute tasks involving diverse and irregularly shaped objects.

Module Connections. As hierarchical systems, such algorithms rely on the integration of multiple independent modules, which inevitably introduces errors at the interfaces between components. For example, the large language model may generate incorrect outputs, such as failing to locate the corresponding object or the constraints generated by the system may not be successfully converted into waypoints by the solver. These errors significantly reduce the system's robustness when handling diverse task conditions. The inability to reliably bridge constraints and waypoints highlights a critical limitation in the framework's modular connectivity, further undermining its ability to adapt to varying operational scenarios.

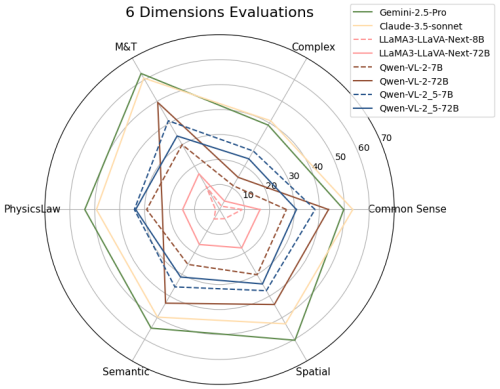


Figure 17. Performance of different sizes for the same model family.

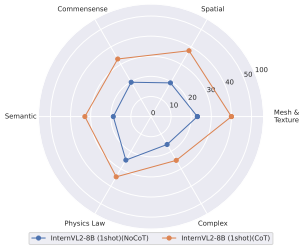
10.4. Ablations and Analysis for VLMs

In our evaluation of VLMs, we conducted two key experiments to explore the impact of Chain-of-Thought (CoT) prompting and few-shot learning on model performance.

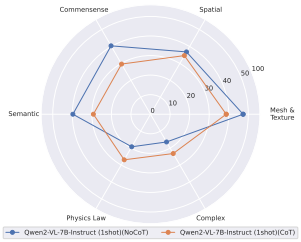
Effect of CoT Prompting. Our investigation into the use of CoT prompting revealed a notable improvement in overall performance for the InternVL2 model, as shown in Figure 18. Similarly, LLaVA-NeXT and Qwen2-VL demonstrated enhanced performance in challenging tasks, particularly those requiring reasoning about complex scenarios and physics laws. However, their performance on semantically common-sense tasks remained stagnant or experienced minor degradation. In contrast, the MiniCPM model exhibited significant limitations: it failed to output answers at the conclusion of the reasoning process when CoT was applied, resulting in all scores dropping to 0.0.

Effect of Few-Shot Learning. As shown in Figure 19 our exploration of few-shot learning with the Qwen2-VL model indicated that increasing the number of few-shot examples (0 to 7) enhances the model's multimodal reasoning capabilities, particularly under CoT prompting. This enhancement was observed across both basic and complex scenarios. However, we found diminishing returns beyond two or three shots for tasks involving diverse semantic requirements or spatial reasoning. This suggests that the utility of additional examples is context-dependent and saturates relatively quickly in certain domains.

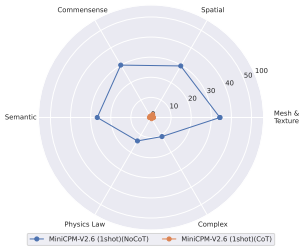
Effect of Model Size. To verify whether scaling in the VLM domain applies to embodied reasoning tasks, we selected models from the same model family but with different sizes for evaluation. The results are shown in Figure 17. While larger VLMs from the same model family typically exhibit better performance, this trend does not always hold, where Qwen2.5-VL serves as a counterexample.



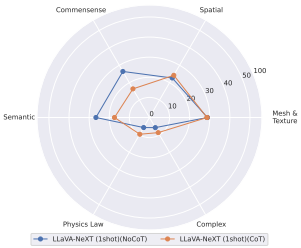
(a) Result of InternVL2-8B.



(b) Result of Qwen2-VL-7B.



(c) Result of MiniCPM-V2.6.



(d) Result of LLaVA-NeXT.

Figure 18. Variation of the six-dimensional scores of the different models in the CoT case, where the orange line represents the case with CoT, and the blue line represents the case without CoT.

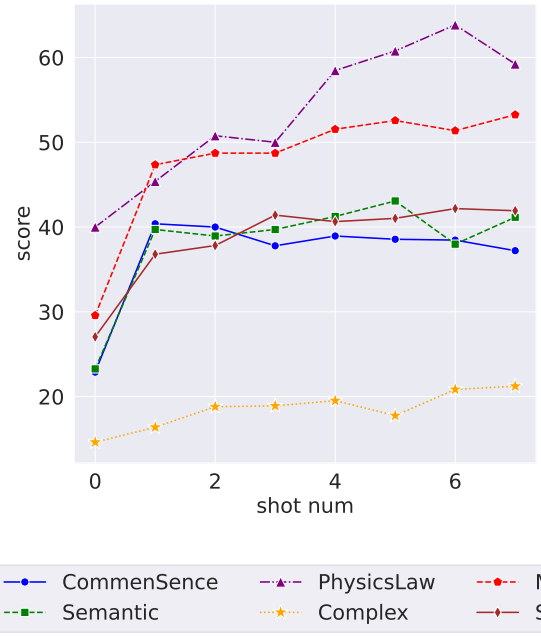


Figure 19. The impact of different few-shot settings on the performance of VLMs. As the number of few-shot examples increases, the generation quality of the model improves progressively.

Model	Task	Track 1			Track 2			Track 3			Track 4			Track 5			Track 6		
		IS	PS	SR	IS	PS	SR	IS	PS	SR	IS	PS	SR	IS	PS	SR	IS	PS	SR
Octo	Task 1	0.10	0.0	0.0	0.12	0.0	0.0	0.06	0.00	0.00	0.14	0.00	0.00	0.20	0.00	0.00	0.20	0.00	0.00
	Task 2	0.14	0.02	0.00	0.08	0.00	0.00	0.04	0.00	0.00	0.04	0.00	0.00	0.14	0.00	0.00	0.14	0.04	0.00
	Task 3	0.68	0.04	0.00	0.46	0.02	0.00	0.52	0.04	0.00	0.04	0.00	0.00	0.14	0.00	0.00	0.14	0.04	0.00
	Task 4	0.46	0.12	0.00	0.46	0.02	0.00	0.52	0.04	0.00	0.60	0.04	0.00	0.10	0.00	0.00	0.20	0.06	0.00
	Task 5	0.22	0.10	0.10	0.16	0.04	0.00	0.20	0.04	0.04	0.22	0.06	0.06	0.22	0.00	0.00	0.20	0.00	0.00
OpenVLA	Task 1	0.38	0.08	0.00	0.30	0.02	0.00	0.32	0.04	0.00	0.28	0.04	0.00	0.26	0.00	0.00	0.26	0.16	0.02
	Task 2	0.16	0.00	0.00	0.10	0.02	0.00	0.12	0.00	0.00	0.16	0.02	0.00	0.14	0.00	0.00	0.34	0.12	0.00
	Task 3	0.84	0.06	0.00	0.72	0.04	0.00	0.72	0.06	0.00	0.68	0.06	0.00	0.24	0.00	0.00	0.50	0.16	0.00
	Task 4	0.74	0.26	0.08	0.36	0.13	0.02	0.40	0.12	0.04	0.44	0.18	0.00	0.04	0.00	0.00	0.22	0.04	0.00
	Task 5	0.38	0.24	0.24	0.30	0.14	0.14	0.34	0.12	0.12	0.34	0.16	0.14	0.12	0.00	0.00	0.12	0.00	0.00
RDT-1B	Task 1	0.38	0.12	0.00	0.30	0.07	0.00	0.28	0.06	0.00	0.30	0.04	0.00	0.30	0.00	0.00	0.20	0.08	0.00
	Task 2	0.24	0.06	0.00	0.10	0.02	0.00	0.10	0.00	0.00	0.12	0.02	0.00	0.22	0.00	0.00	0.26	0.00	0.00
	Task 3	1.0	0.15	0.0	0.84	0.11	0.00	0.84	0.08	0.00	0.76	0.08	0.00	0.22	0.00	0.00	0.30	0.20	0.00
	Task 4	0.42	0.09	0.00	0.38	0.05	0.00	0.26	0.02	0.00	0.20	0.04	0.00	0.10	0.00	0.00	0.12	0.02	0.00
	Task 5	0.46	0.30	0.30	0.34	0.16	0.16	0.32	0.20	0.20	0.40	0.22	0.22	0.18	0.00	0.00	0.20	0.04	0.00
π_0-Base	Task 1	0.34	0.04	0.00	0.38	0.02	0.0	0.32	0.00	0.00	0.38	0.03	0.00	0.22	0.00	0.00	0.46	0.22	0.04
	Task 2	0.10	0.02	0.00	0.14	0.03	0.00	0.40	0.07	0.00	0.44	0.09	0.00	0.32	0.00	0.00	0.34	0.16	0.06
	Task 3	1.00	0.48	0.00	0.94	0.42	0.00	1.00	0.48	0.00	1.00	0.45	0.00	0.24	0.00	0.00	0.44	0.10	0.00
	Task 4	0.88	0.39	0.12	0.78	0.2	0.06	0.86	0.23	0.04	0.84	0.21	0.00	0.08	0.00	0.00	0.20	0.10	0.00
	Task 5	0.66	0.54	0.54	0.4	0.24	0.24	0.44	0.22	0.22	0.36	0.18	0.18	0.24	0.00	0.00	0.28	0.06	0.00
π_0-Fast	Task 1	0.42	0.16	0.04	0.42	0.11	0.00	0.48	0.14	0.00	0.46	0.13	0.02	0.40	0.00	0.00	0.26	0.12	0.04
	Task 2	0.40	0.22	0.06	0.30	0.10	0.00	0.56	0.09	0.00	0.46	0.07	0.00	0.24	0.00	0.00	0.40	0.14	0.04
	Task 3	0.96	0.37	0.00	0.86	0.36	0.00	0.80	0.32	0.00	0.76	0.31	0.00	0.18	0.00	0.00	0.56	0.30	0.00
	Task 4	0.69	0.18	0.02	0.48	0.13	0.0	0.30	0.04	0.00	0.38	0.03	0.00	0.12	0.00	0.00	0.12	0.06	0.00
	Task 5	0.52	0.34	0.34	0.32	0.12	0.12	0.36	0.18	0.18	0.54	0.32	0.32	0.34	0.00	0.00	0.36	0.10	0.00

Table 9. **Detailed result of the evaluation of VLAs.** In track 1-4, tasks 1-5 are Select Fruit, Select Toy, Insert Flower, Add Condiment, Select Painting; In track 5, tasks 1-5 are Select Poker, Select Mahjong, Select Billiards, Select Ingredient, Friction QA; In track 6, tasks 1-5 are Texas Holdem, Play Math Game, Find Unseen Object, Cluster Toy, Hammer Nail then Hang Picture

Model	SR	PR	SPR	MS	Score	FER
Llava_NeXT	42.80	15.89	12.92	4.10	23.48	33.97
InternVL2	33.51	15.07	11.39	2.98	19.43	40.56
GPT_4v	59.41	27.22	24.70	7.35	34.65	14.07
GLM4v	68.97	3.15	11.91	0.53	28.85	0
MiniCPM_V2_6	47.14	21.86	16.96	4.39	27.60	6.50
GPT_4o	51.41	32.43	28.79	11.54	33.54	0.31
Qwen2_VL	56.27	37.53	26.80	9.06	37.52	2.67

Table 10. Full evaluation results of VLMs, taking the average of all the evaluation episodes. FER: format error rate.

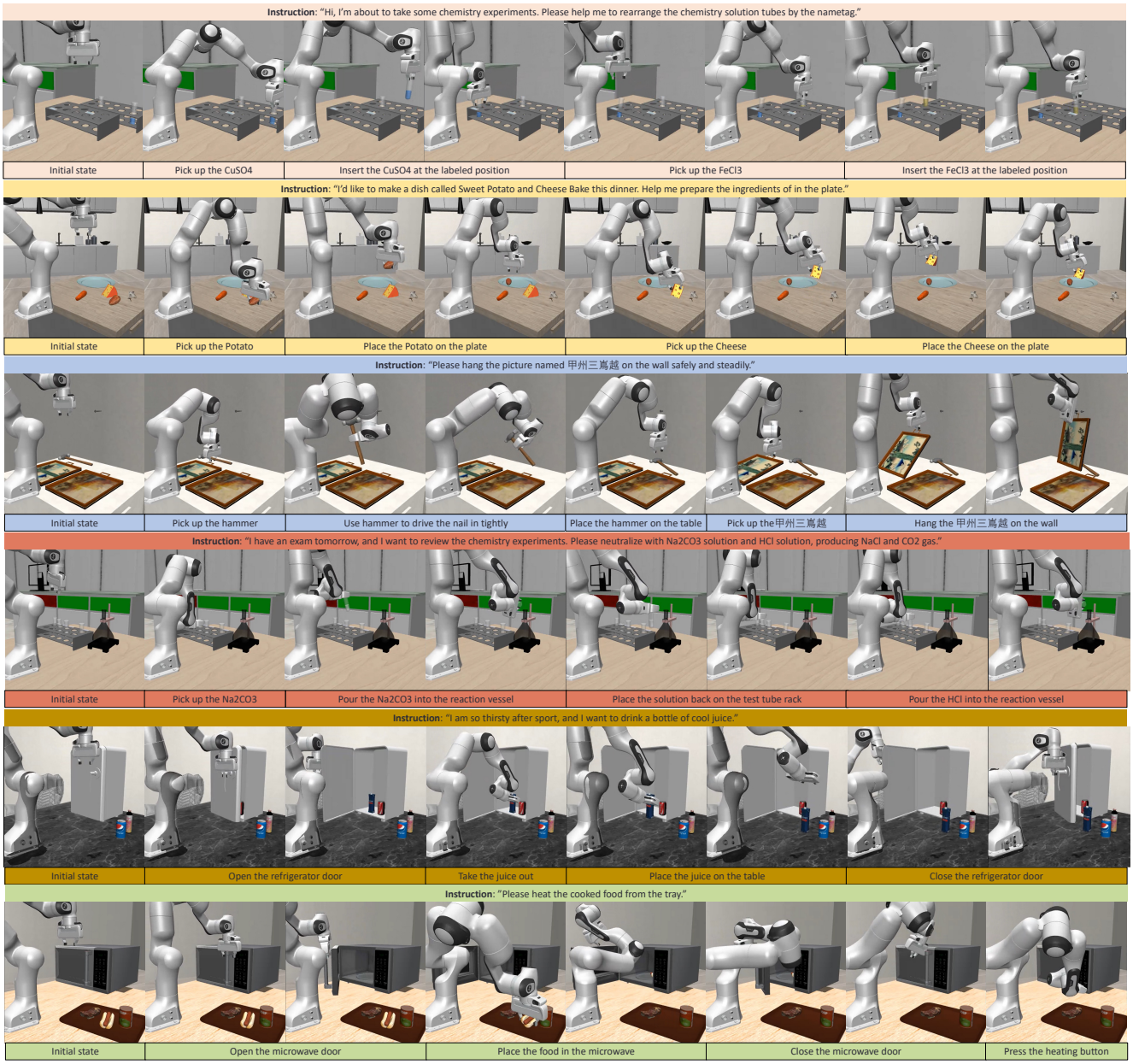


Figure 20. **Long horizon task requiring reasoning.** These tasks all involve multiple skills and require reasoning.

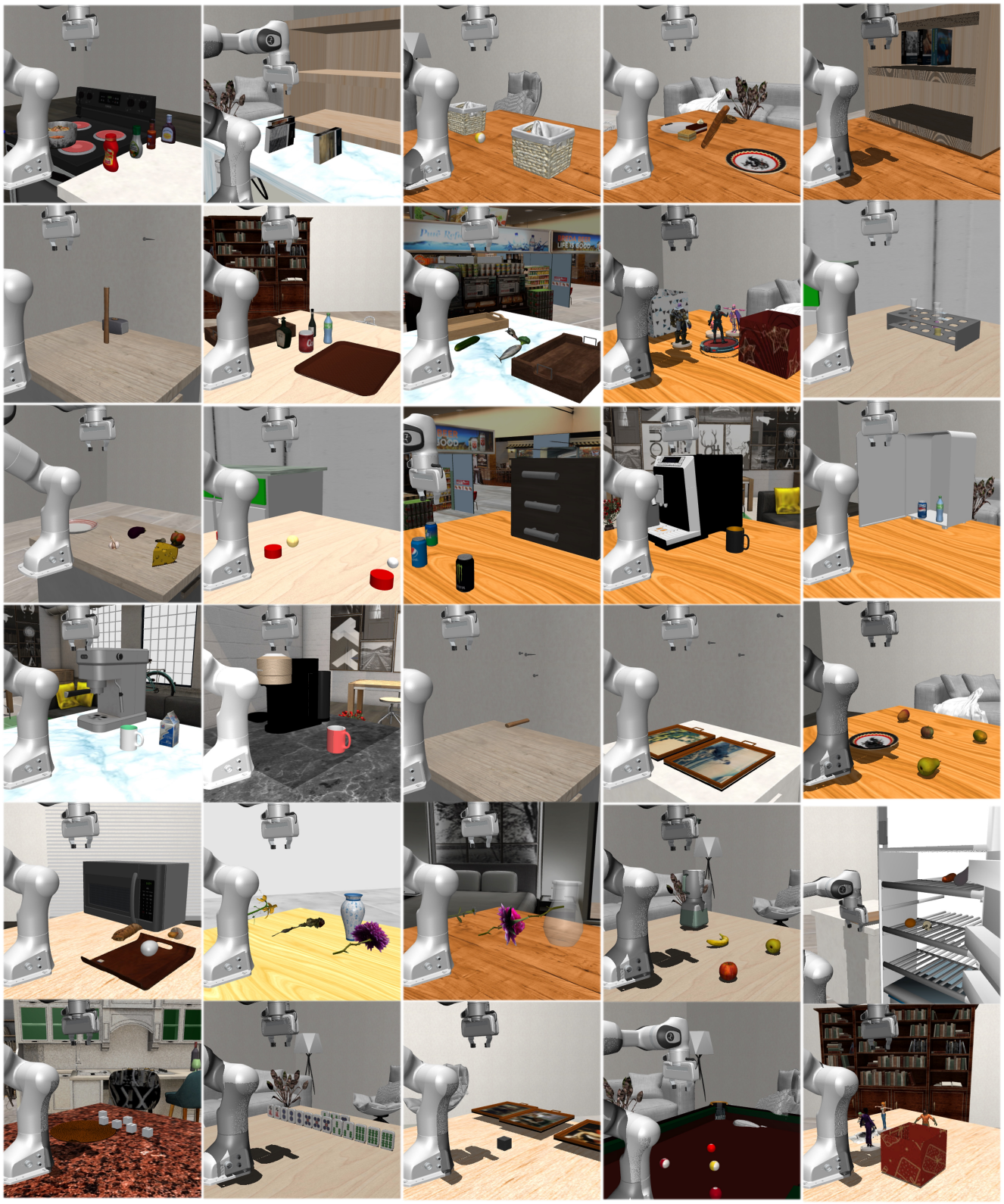


Figure 21. Part of tasks in VLABench.

Task	Type	Ability Dimension	Skill Involved	Description
Select Fruit	Primitive	M&T	Pick&place	Pick the specific fruit into a specific receptacle, such as "put the strawberry into the basket".
Select Drink	Primitive	M&T	Pick&place, Pull	Get the specific drink from a particular receptacle, such as "pick the cola out of the fridge".
Select Toy	Primitive	M&T	Pick&place	Put the specific toy into a specific receptacle, such as "Select Ironman from the toys and wrap it in the gift box".
Select Book	Primitive	M&T	Pick&place, Pull	Take a particular book from the receptacle, such as "Take *Pride and Prejudice* from the bookshelf".
Select Ingredient	Primitive	M&T	Pick&place	Get the specific ingredient from the particular receptacle, such as "Take the bell pepper from the fridge and place it on the tray".
Insert Flower	Primitive	M&T	Pick&place, Insert	Insert the specific flower into the container, such as "Insert the rose into a vase."
Add Condiment	Primitive	M&T	Pick&place, Pour	Add the specific condiment into the dish, such as "Add some salt into the dish in the pot."
Put Box on Famous Painting	Primitive	M&T	Pick&place	Place the geometric shape on the specified famous painting., such as "Press the button before the painting *The Starry Night*"
Pick ChemistryTube	Primitive	M&T	Pick&place	Select specific solution tube based on the nametag, such as "pick up the tube of CuCl2".
Select Poker	Primitive	M&T	Pick&place	Select specific poker, such as "Pick jack of red heart".
Select Mahjong	Primitive	M&T	Pick&place	Select specific mahjong, such as "Pick mahjong: 2 of Man and put it on the placemat".
Select Billiards	Primitive	M&T	Pick&place	Select specific Billiards, such as "Pick Black 8 and place it in any hole."
Hammer Loose Nail	Primitive	M&T	Pick&place, Tool use	By comparing the lengths of different nails, use a hammer to tighten the loose nail, such as "Hammer the loose nail on the wall".
Select Fruit-Spatial	Primitive	M&T, SP	Pick&place	Pick the fruit in a specific place or a certain spatial relationship, such as "put the nearest strawberry into the plate".
Select Drink-Spatial	Primitive	M&T, SP	Pick&place	Get the drink from a specific place or a certain spatial relationship, such as "pick the monster outside", while there is also a can of monster inside the fridge.

Select Toy-Spatial	Primitive	M&T, SP	Pick&place	Take the toy from a specific place or a certain spatial relationship, such as "Place the toy on Luffy's right-hand side into the box".
Select Book-Spatial	Primitive	M&T, SP	Pick&place	Take the book on the specific position or a certain spatial relationship, such as "Take out the book on the most left in the top layer".
Select Ingredient-Spatial	Primitive	M&T, SP	Pick&place	Get the ingredient on the specific position or in a certain spatial relationship, such as "Place the ingredient on the bottom layer in the fridge onto the tray".
Insert Flower-Spatial	Primitive	M&T, SP	Pick&place	Insert the flower on the specific position or in a certain spatial relationship, "Place the flower on the far left into the vase.".
Add Condiment-Spatial	Primitive	M&T, SP	Pick&place	Add the condiment on the specific position or in a certain spatial relationship, such as "Add the furthest spice to the dish".
Hang Picture	Primitive	M&T, SP	Pick&place, Hang	Hang the picture on the nail in the specified location, such as "Hang the picture on the highest nail".
Pick ChemistryTube-Spatial.	Primitive	M&T, SP	Pick&Place	Take out the chemistry solution tube on the specific position or in a certain spatial relationship, such as "Take the tube in the first row, the second column before you".
Select Poker-Spatial	Primitive	M&T, SP	Pick&Place	Select the poker on the specific position or in a certain spatial relationship, such as "Pick the second poker from left to right".
Select Mahjong-Spatial	Primitive	M&T, SP	Pick&Place	Select the mahjong on the specific position or in a certain spatial relationship, such as "Pick the mahjong on the right of six of sou".
Put Billiards in Pocket	Primitive	M&T, SP	Pick&Place	Place the billiard ball into the specified pocket, such as "Place the 8-ball into the pocket in the right front".
Select Fruit with Common Sense	Primitive	M&T, C&W	Pick&Place	Select fruit with specific characteristics, including nutritional characteristics, common uses, whether they grow in clusters, easy to peel, etc. Example: "Put the fruit with the most vitamin C into the basket" from among **orange**, banana, and apple.
Select Drink with Common Sense	Primitive	M&T, C&W	Pick&Place	Select a drink with some specific characteristics including types of beverages, functions of the beverages, flavors of the beverages, etc. Example: "Get a can of energy drink from the fridge" from among cola, apple juice, and **redbull**.

Select Toy with Common Sense	Primitive	M&T, C&W	Pick&Place	Select the toy with some specific characteristics including the associated IP, character personality, character background, etc. Example: "Put the toy from the Marvel series to the giftbox" from among **Hulk**, Batman, and Mickey.
Select Book with Common Sense	Primitive	M&T, C&W	Pick&Place	Select the book with some specific characteristics including the type of the book, the content of the book, the main message it conveys, etc. Example: "Get the book about computer science" from among *Steve Jobs*, ***3D Computer Vision*** and *War and Peace*.
Select Ingredient with Common Sense	Primitive	M&T, C&W	Pick&Place	Select the ingredient with some specific characteristics, such as "Pick an ingredient full of protein from the fridge and put it on the tray" from among **egg**, tomato and bell pepper.
Insert Flower with Common Sense	Primitive	M&T, C&W	Pick&Place	Insert the flower with some specific characteristics into vase including the flower language, the symbolic qualities of the flower, appropriate occasions for giving flowers, etc. Example:"Insert the flower suitable for Valentine's Day into the vase" from **rose**, sunflower, and tulip.
Insert Bloomed Flower	Primitive	M&T, C&W	Pick&Place	An intelligent agent should possess awareness: flowers should be arranged with blooming ones, not with those that are already withered. Example: "Insert a proper flower into the vase" from among wilted rose, wilted daisy, and **sunflower**.
Add Condiments with Common Sense	Primitive	M&T, C&W	Pick&Place	Add the condiment with some specific characteristics including distinctive flavor, seasoning role, suitability for various dishes, and etc. Example: "Add the condiment that makes the dish taste more salty" from among **salt**, ketchup, and salad dressing.
Select Painting with Common Sense	Primitive	M&T, C&W	Pick&Place	Press the button before the painting with specific styles or contents. Example: "Choose the painting in the style of rococo" among from paintings of ***La Liseuse***, *The Stary Night*, and *Golden Autumn*.
Pick Chemistry Tube with Common Sense	Primitive	M&T, C&W	Pick&Place	Pick up the specific solution without the nametag and distinguish by the solution color. Example: "Pick up the solution of CuSO4" from among the solution of **blue**, green, and yellow.

Select nth Largest Poker	Primitive	M&T, C&W	Pick&Place	Choose the largest poker under the rule of the specific poker games. Example: "Pick the largest poker in single under the rule of Texas Holdem" from among **Ace of Spades**, Three of Hearts, and Queen of Clubs.
Select Unique Mahjong	Primitive	M&T, C&W	Pick&Place	Choose the mahjong with the unique type. Example:"Pick the unique type of mahjong" among from **East**, One of Man, Nine of Man.
Select Billiards with Common Sense	Primitive	M&T, C&W	Pick&Place	Select a specific billiard game under particular rules with a specific score. Example: "Place the two-point ball from a snooker match into any pocket" from among green ball, **yellow ball**, and red ball.
Select Fruit-Semantic	Primitive	M&T, SEM	Pick&Place	The user expresses implicit needs for a certain fruit during a semantically rich conversation or context, such as: "Today, I suddenly feel like doing some baking and plan to make a strawberry cake! Could you help me prepare the fruits I'll need?"
Select Drink-Semantic	Primitive	M&T, SEM	Pick&Place, Pull	The user expresses implicit needs for a certain drink during a semantically rich conversation or context, such as: "I just worked out at the gym for a long time, and now I'm a bit dehydrated. Could you help me grab a bottle of electrolyte drink from the fridge?"
Select Toy-Semantic	Primitive	M&T, SEM	Pick&Place	The user expresses implicit needs for a specific toy during a semantically rich conversation or context, such as "I've loved Disney since I was a kid, especially the Toy Story series! I want to place Buzz Lightyear on the top layer of the shelf, but I can't reach it! "
Select Book-Semantic	Primitive	M&T, SEM	Pick&Place, Pull	The user expresses implicit needs for a specific book during a semantically rich conversation or context, such as "I'm getting ready to review for my final Python exam. Could you help me prepare the textbook?"
Select Ingredient-Semantic	Primitive	M&T, SEM	Pick&Place	The user expresses implicit needs for a specific ingredient during a semantically rich conversation or context, such as "I'm keeping fit so I want to eat something full of protein. I want a steak as my lunch and could you get one for me?"

Insert Flower-Semantic	Primitive	M&T, SEM	Pick&Place	The user expresses implicit needs for a specific flower during a semantically rich conversation or context, such as "Today is Teacher's Day, and Ms. Lisa has always been kind to me. I want to give her a bouquet of carnations. Could you help me place them in the vase on her desk?"
Add Condiment-Semantic	Primitive	M&T, SEM	Pick&Place, Pour	The user expresses implicit needs for a specific condiment during a semantically rich conversation or context, such as "I'm making tomato-braised beef brisket, but the tomato flavor doesn't seem strong enough. Could you help me add some tomato paste? Thanks!"
Select Painting-Semantic	Primitive	M&T, SEM	Press	The user expresses implicit needs for a specific solution during a semantically rich conversation or context, such as: "I am a student who has just started learning painting, and I'm not very good at distinguishing between different styles of painting. Could you help me identify which of these three paintings is in the realist style?"
Select ChemistryTube-Semantic	Primitive	M&T, SEM	Pick&Place	The user expresses implicit needs for a specific solution during a semantically rich conversation or context, such as: "I'm going to demonstrate an acid-base neutralization experiment today, but I'm missing an acid-base indicator. Could you help me grab the phenolphthalein solution?"
Simple Poker Play	Primitive	M&T, SEM	Pick&Place	The agent plays the poker that should be played on behalf of the player during the semantically rich interaction. Example: "We're playing Landlord, and the player before me just played a 10. Now it's our turn. Please play a 2 for me."
Simple Mahjong Play	Primitive	M&T, SEM	Pick&Place	The agent plays the Mahjong that should be played on behalf of the player during the semantically rich interaction. Example: "It's hard to win with the 'Wan' character tiles left. Go ahead and discard the '1 Wan'."
Simple Snooker Play	Primitive	M&T, SEM	Pick&Place	The agent picks the billiard that should be played on behalf of the player during the semantically rich interaction. "We're playing a simple game of snooker. Now, let's pot the yellow ball into the pocket."

Friction QA	Primitive	M&T, PHY	Press	Using the relevant physics knowledge of sliding friction and rolling friction, determine the rolling speed of different shaped and material objects on a slope. Example: "Press the button before the object that falls the fastest down the slope."
Density QA	Primitive	M&T, PHY	Press	Visually judge the material of an object and determine the relative density of objects made from different materials. Example: "Press the button before the object can float on water."
Magnetism QA	Primitive	M&T, PHY	Press	Visually identify the material of an object and determine whether objects made from different materials are magnetic. Example: "Press the button before the object that is not magnetic."
Weight QA	Primitive	M&T, PHY	Press	Visually identify the material of an object, and combine the material density and shape (in the actual setup, this includes cubes of different shapes, along with their corresponding inscribed spheres and circumscribed spheres) to make a comprehensive judgment of the object's mass. Example: "Press the button before the object with the smallest weight."
Thermal Expansion QA	Primitive	M&T, PHY	Press	Visually identify the material of an object and determine the thermal expansion properties of objects made from different materials. Example: "Press the button before the object with a medium thermal expansion coefficient."
Speed of Sound QA	Primitive	M&T, PHY	Press	Visually identify the material of an object and determine the sound propagation speed in objects made from different materials. Example: "Press the button before the object that sound propagates fastest in."
Specular Reflection QA	Primitive	M&T, PHY	Press	Judge based on visual information whether different objects exhibit specular reflection and make a selection. Example: "Press the button before the object that can reflect the image of others."
Drag Force QA	Primitive	M&T, PHY	Press	Determine the object's free fall speed based on its shape, texture, and material. This involves physical theories such as air resistance, the Kármán vortex street effect, and others. Example: "Press the button before the object falls slowest in the air" from among **golf**, basketball, football.

Basic Seesaw Usage	Primitive	M&T, PHY	Pick&place, Tool use	Using the principle of leverage, place a heavy object on one side of the seesaw to lift the other side. Example: "Make the other side of the seesaw lift".
Strike Billiards	Primitive	M&T, PHY	Pick&place, Tool use	Use the laws of collision to perform a simple strike. Example: "Use the cue stick to strike the white ball, aiming to make it hit other colored balls".
Take Chemistry Experiment	Composite	M&T, SP, SEM, C&W, L&R	Pick&place, Insert, Pour	The agent should first use the user's request for the desired chemical product, **combine it with visual observation and common knowledge for logical reasoning**, and determine the chemical **solutions involved in the reaction**. After identifying the appropriate solutions using the name tag, the agent should select the solutions and mix them into the flask. Example: "I would like to obtain AgCl precipitation in the flask. Please carry out this experiment."
Find Unseen Object	Composite	M&T, SP, L&R	Open&close drawer, Pick&Place, Explore	The target object is **not directly visible**, requiring the agent to open multiple drawers and eventually find the target object. Example: "Find a snack in the drawer for me".
Find Unseen Object without Telling Find	Composite	M&T, SP, SEM, C&W, L&R	Open&close drawer, Pick&Place, Explore	The other settings are the same as for Find Unseen Object, but the requirements are implicitly conveyed through semantically rich dialogue. The agent needs to be aware of the need for **exploration and search on its own**. Example: "I'm a bit hungry, could you get me something to eat?"
Make Juice with Juicer	Composite	M&T, SEM, L&R	Pick&place, Tool use, Press	Select the appropriate fruits based on semantically rich user instructions, place them into a container, and correctly use the juicer. Example: "It's so hot today! I feel like having a freshly squeezed kiwi and strawberry juice right now."
Find Fruit to Make Juice	Composite	M&T, C&W, SEM, L&R	Pick&place, Tool use, Press, Explore	The fruits are **not directly available and visible** to the agent because the fruits are stored in a closed fridge or a cabinet. The agents should find the proper fruit first. The example is the same as above.
Plug-in Power Cord to Make Juice	Composite	M&T, C&W, SEM, L&R	Pick&place, Tool use, Press, Explore, Insert	The other basic settings remain the same as above. However, the juicer's **power cord is not plugged in**. The agent needs to first observe this and, using common sense, plug in the power cord to supply power. The example is the same as above.

Take out Cool Drink	Composite	M&T, SP, C&W, SEM, L&R	Open&close door, Pick&place	Obtain user requirements through semantically rich interaction: the user wants a cold drink. Given the observation of the same target **drink on the desk as disturbance**, the agent should use **common sense to determine that the drink from the fridge** should be chosen. Example: "The weather is so hot! I feel like having a cold soda."
No Drink in Fridge & Refrigerate Drink	Composite	M&T, SP, C&W, SEM, L&R	Open&close door, Pick&place, Explore	The task is set the same as above. However, after the agent opens the fridge door, it finds that the target object is not there. The agent needs to realize that it should first **refrigerate the room-temperature target drink**.
Wrap Proper Toy as Gift	Composite	M&T, C&W, SEM, L&R	Open&close door, Pick&place	Choose a suitable toy for kids as a gift from product shelf during the semantic interaction with the user. Then wrap in as a gift. Example: "My son is a superhero fun, but I don't know that much. Could you wrap a gift for him?"
Rearrange Books by Year	Composite	M&T, C&W, SP, L&R	Pick&place	Identify the book title and use **world knowledge to determine the publication period**. Then rearrange them. Example: "Rearrange the book by published year order in the top layer of the shelf, the far left is the earliest one."
Rearrange Books by Author Name	Composite	M&T, C&W, SP, L&R	Pick&place	Identify the book title and use **world knowledge to determine the author name**. Then rearrange them. Example: "Rearrange the book by their author names, the far right starts with the largest word."
Classify the Books	Composite	M&T, C&W, SP, L&R	Pick&place	Identify the book titles and categorize the books based on their genre or content. The agent needs to **infer the classification criteria** on its own and correctly divide the books into two layers. Example: "Divide the books into two classes, one class on the top layer while another on the bottom."
Cook Dishes Following Menu	Composite	M&T, C&W, SEM, L&R	Pick&place	**Multi-turn pick and place the correct ingredients** for a dish whose menu is offered by semantic instructions. Example: "I'm about to cook a dish of tomato-fried eggs, prepare ingredients in the tray."
Store Proper Food	Composite	M&T, C&W, SEM, L&R	Open&close door, Pick&place	Store the ingredients or fruits into the fridge and do not put the disturbance including snacks into the fridge. Example: "I left some food on the table in the last meal, store them properly please."

Heat Food with Microwave	Composite	M&T, C&W, SEM, L&R	Open&close door, Pick&place, Press	**Extract implicit goals** from semantically rich interactions: heating food. Use common sense to choose the proper food, such as a hot dog, with the microwave, **while avoiding heating canned food or raw ingredients**. Finally, correctly use the microwave. Example: "I just finished class, and now my stomach is growling. Could you heat up some food for me to have a quick bite?"
Plug-in Power Cord to Heat Food	Composite	M&T, C&W, SEM, L&R	Open&close door, Pick&place, Press, Insert	The other experimental settings remain the same as above. The agent must first have the **common sense to plug in the power source** for the device to operate. The example is the same as above.
Replace Wilted Flower and Drop	Composite	M&T, C&W, SEM, L&R	Pick&place, Insert	Based on the semantically rich user request and using common sense, determine the target flower. Discard the wilted flower in the vase, and then insert the new flower. Example: "It's Valentine's Day today, replace the flower in the vase."
Find Condiment and Add to Dish	Composite	M&T, C&W, SEM, SP, L&R	Open&close drawer, Pick&Place, Explore, Pour	All the condiments are stored in the cabinet and the agent should **proactively find them first** and add proper condiment into the dish. Example: "The spiciness of this dish isn't quite enough. Could you add some more seasoning to make it tastier?"
Hammer Nail &Hang Picture	Composite	M&T, C&W, SEM, L&R	Pick&place, Tool use, Hang	The agent needs to observe and determine if the nail is loose, and then use a hammer to tighten the nail. After that, the agent should hang the appropriate picture on the wall. Example: "Hang 'the Stary Night' on the wall steadily."
Assemble Hammer &Hammer Nail	Composite	M&T, C&W, L&R	Pick&place, Insert, Tool Use	The agent needs to observe and **reason that the task cannot be completed with the current conditions**. It must first assemble the hammer handle and the hammerhead precisely before proceeding. Example: "Hammer the loose nail."
Rearrange Chemistry Tube	Composite	M&T, SP, C&W, L&R	Pick&place, Insert	Rearrange the multiple tubes by the **corresponding relationships between color and name tag**, the result of utilizing common sense and reasoning ability. Example: "Rearrange the solution tubes."
Texas Holdem Play	Composite	M&T, C&W, SEM, L&R	Pick&Place	**Deduce the strongest Texas Hold'em hand based on the common game rules and visual information**. Then take multi-step pick&place. Example: "We are playing Texas Hold'em, place your strongest hand combination on the placemat."

Flip Facing-downs &Play Texas Holdem	Composite	M&T, C&W, SEM, L&R	Pick&Place, Twist, Explore	Based on the previous task, some of the cards are face down. The agent needs to have an **exploration mindset and actively retrieve all the observational information**, then make the correct judgment. The example is the same as above.
Play Mahjong	Composite	M&T, C&W, SEM, L&R	Pick&Place	The agent **makes decisions based on world knowledge of Mahjong rules combined with visual information**. It discards an unnecessary tile and draws a necessary tile to win. Example: "We seem to be close to winning the game. Take the right actions to help us win this round."
Flip Facing-downs &Play Mahjong	Composite	M&T, C&W, SEM, L&R	Pick&Place, Twist, Explore	The agent needs to have an **exploration mindset and actively retrieve all the observational information**, then make the correct judgment. The example is the same as above.
Leverage SeeSaw to Grasp Target	Composite	M&T, C&W, SEM, PHY, L&R	Pick&place, Explore, Tool use	Using the lever principle, place one or more heavy objects on one side of the seesaw to lift the target object on the other side, initially unreachable. The challenge lies in the fact that **if the placed weights are insufficient, the agent will need to add additional weights**. Example:"I want to eat the pear in the glass container, but I can't get it out. Can you help me?"
Find Weights to Leverage SeeSaw	Composite	M&T, C&W, SEM, SP, PHY, L&R	Open&close drawer, Pick&place, Explore, Tool use	All the weights are stored in the cabinet and are not visible. The agent needs to **explore multiple drawers to find enough weights** before being able to properly use the seesaw. The example is the same as above.
Get Black Coffee	Composite	M&T, C&W, SEM, L&R	Pick&place, Tool use, Press	The agent needs to derive the task goal from semantically rich interactions: to prepare a cup of **coffee without milk and sugar**. Then, it should correctly place the cup and operate the coffee machine. Example: "I'm feeling sleepy right now. Could you get me a cup of coffee? An Americano will do."
Get Sweet Coffee	Composite	M&T, C&W, SEM, L&R	Pick&place, Tool use, Press, Pour	The agent needs to additionally infer firstly: the user prefers sweet coffee -¿ **the coffee needs to be prepared with sugar**. Example: "Get me a cup of sweet coffee to clear my mind, thx!"
Get Latte Coffee	Composite	M&T, C&W, SEM, L&R	Pick&place, Tool use, Press, Pour	The agent needs to additionally infer firstly: **latte coffee is composed of black coffee and milk**. Example: "A cup of latte please. Nice to meet you here, thank god!"

Set Dining Table by Menu	Composite	M&T, C&W, SEM, SP, L&R	Pick&place	The agent needs to **infer the appropriate utensils based on semantic interactions and the type of cuisine**. For example, chopsticks for Chinese cuisine, a knife and fork for Western cuisine, and a spoon if soup is being served. Example: "Today's main course is steak! Please help me set up the table."
Set Dining Table Left-Handed	Composite	M&T, C&W, SEM, SP, L&R	Pick&place	The agent needs to first extract the key information from user interactions that the user is **left-handed**. Then, using common sense, it should **adjust the placement of the utensils**, such as switching from the original left-knife-right-fork arrangement to a left-fork-right-knife setup. Example: "Tonight, we're having fried rice! Remember to get me a spoon. Oh, and don't forget that I'm left-handed."
Play Snooker	Composite	M&T, C&W, SEM, L&R	Pick&place	Put the ball into the hole by snooker order: yellow, green, brown, blue, pink, black. The agent needs to make the **correct sequence of decisions based on snooker rules in world knowledge**. Example: "Put the colored billiards into holes by score order in a snooker match."
Cluster Toy	Composite	M&T, C&W, SP, L&R	Pick&place	Based on common sense, world knowledge, and visual information, cluster the toys according to their **associated IPs, character types**, and other attributes. Example: "Cluster the toys into two classes." These toys are Spiderman, Hawk Eye, Nami, Chopper.
Classify Desserts	Composite	M&T, C&W, SP, L&R	Pick&place	Based on common sense, world knowledge, and visual information, categorize the desserts according to their types. Example: "Classify the different desserts." These desserts are strawberry donut, banana donut, coco cupcake, common cupcake.
Setup Study Table	Composite	M&T, C&W, SP, L&R	Pick&place Open laptop	Determine the task goal from semantically rich interactions: the user needs a specific book and to use the computer. The agent needs to use common sense to **infer the correct book** and place it on the desk, while also **turning on the computer**. Example: "I have a Python practical exam the day after tomorrow, and I'm planning to review later. Could you help me set up my desk?"

Organize Study Table	Composite	M&T, C&W, SP, L&R	Pick&place Close laptop	Determine the task goal by observing the desk and combining user interactions: organize the desk. This requires completing subtasks in sequence, including **arranging the books** and closing the laptop. Example: "That's all for today. Please help me tidy up the desk. Thanks!"
Math Game	Composite	M&T, C&W, SEM, L&R	Pick&place	Based on the **math problem provided by the user**, use **logical reasoning to find the answer** and display it by arranging number blocks to form the solution. Example: "Let's play a math game, show me the answer by number blocks. The question is: 'Toulouse has twice as many sheep as Charleston. Charleston has 4 times as many sheep as Seattle. How many sheep do Toulouse, Charleston, and Seattle have together if Seattle has 20 sheep?'" This math question is from the GSM8K dataset [10].
Art Game	Composite	M&T, C&W, SEM, SP, PHY, L&R	Pick&place	Place the geometric object with a **specific physical property** onto the painting that aligns with the user's **hinted content or style**. Example: "Let's play a game of 'Simon Says'! Place the geometric object with a specific physical property onto the painting that aligns with the user's hinted content or style.".
Cluster Beverage	Composite	M&T, C&W, SP, L&R	Pick&place	Based on common sense, world knowledge, and visual information, cluster the drinks according to their types. Example: "Cluster the beverages into two types." These beverages are mango juice, milk, Vodka, Champagne.

Table 11. Task List. Include the name, type, ability required, and detailed description of all the tasks.