Robotic Visual Instruction Supplementary Material

Yanbang Li [1†] Ziyang Gong [2] Haoyang Li [3] Xiaoqi Huang [4] Haolan Kang [5]
Guangping Bai [1] Xianzheng Ma [1]
[1] Independent Researcher [2] Shanghai AI Laboratory [3] UCSD [4] VIVO
[5] South China University of Technology
[†] Corresponding Author

1. Table of Content

This supplementary material contains:

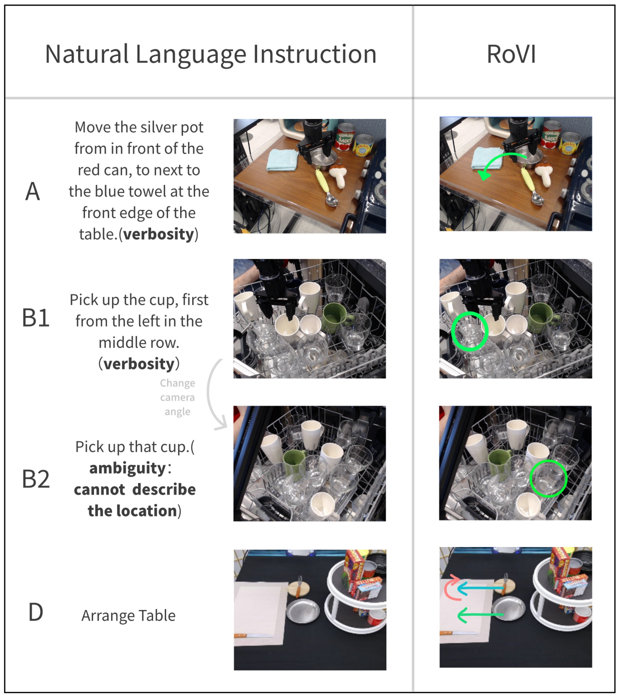

Natural Language Instruction
RoVI
A
Move the silver pot from in front of the red can, to next to the blue towel at the front edge of the table.(**verbosity**)
B1
Pick up the cup, first from the left in the middle row. (**verbosity**)
Change camera angle
B2
Pick up that cup.(**ambiguity**: **cannot describe the location**)
D
Arrange Table

Figure 1. Limitations of Natural Language Instruction.

2. Limitations of Natural Language Instruction

Natural language instructions are common in human-robot interaction but often pose challenges in tasks requiring precise spatial and contextual understanding. We present examples illustrating these challenges and demonstrate how RoVI effectively addresses them.

Spatial Ambiguities. Natural language often lacks precision in conveying spatial relationships. For example: **'Move the silver pot from in front of the red can to next to the blue towel at the front edge of the table.'** As shown in Figure 1 instance (A), such descriptions are ambiguous; phrases like 'next to the blue towel' and 'front edge' can be

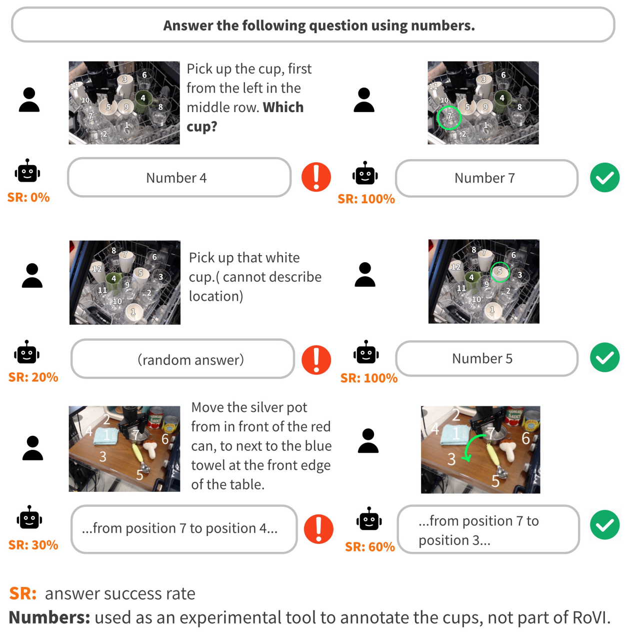

Answer the following question using numbers.
Pick up the cup, first from the left in the middle row. **Which cup?**
Number 4
SR: 0%
Number 7
SR: 100%
Pick up that white cup.(cannot describe location)
(random answer)
SR: 20%
Number 5
SR: 100%
Move the silver pot from in front of the red can, to next to the blue towel at the front edge of the table.
...from position 7 to position 4...
SR: 30%
...from position 7 to position 3...
SR: 60%
SR: answer success rate
Numbers: used as an experimental tool to annotate the cups, not part of RoVI.

Figure 2. Spatial VQA with GPT-4o: The experiment aims to evaluate VLM's spatial reasoning capabilities via simple VQA tests under RoVI compared to traditional language instructions. Results show that RoVI achieves a significantly higher success rate than language-based instructions.

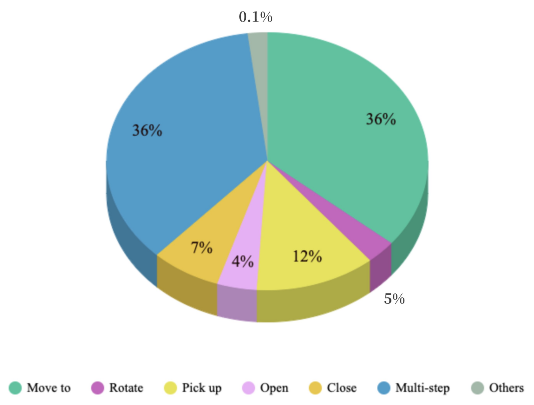


Figure 3. Distribution of skills represented in the RoVI Book dataset.

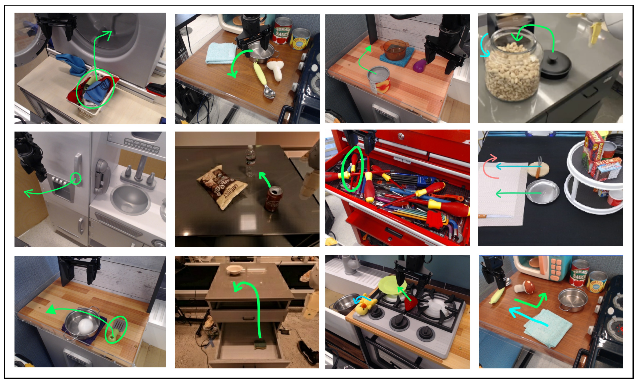


Figure 4. Dataset modified from Open-X Embodiment [2].

interpreted in multiple ways, leading to errors. In contrast, RoVI (A, right) provides explicit visual cues, such as arrows indicating movement and placement, eliminating ambiguity and ensuring accurate execution.

Object Identification Amid Similar Objects. Identifying specific items among similar ones via natural language is challenging. For example, **'Pick up the cup'** in a dishwasher filled with multiple glasses shown in Figure 1 instance B2. This instruction is also vague, making it difficult to select the correct cup. RoVI addresses this by overlaying a circle symbol directly on the target object (B2, right), providing an unambiguous reference and improving identification accuracy.

Complex Tasks and Task Decomposition. Conveying complex tasks requiring domain knowledge is difficult with language alone. For example: **'Arrange the table.'** Without knowledge of the necessary steps (Figure 1 instance D), the robot may perform the task incorrectly. RoVI mitigates this by offering sequential visual cues for each subtask (D, right), guiding the robot through the process, and reducing reliance on prior knowledge.

2.1. Spatial VQA Experiment

This experiment shown in Figure 2 — Robotic Spatial VQA — aims to evaluate VLM's spatial reasoning capabilities via simple VQA tests under RoVI compared to traditional language instructions. Compared to real-world experiments, spatial VQA significantly reduces testing costs. To test our hypothesis that in cluttered environments, real-life scenarios, using only language as a medium to communicate with robots is less efficient (verbose) and accurate (ambiguous) compared to adding visual instructions, we design two types of tasks: 'choose' and 'move to' under cluttered environment. Our conclusion is that visual instructions allow the VLM to better understand the precise spatial locations in tasks.

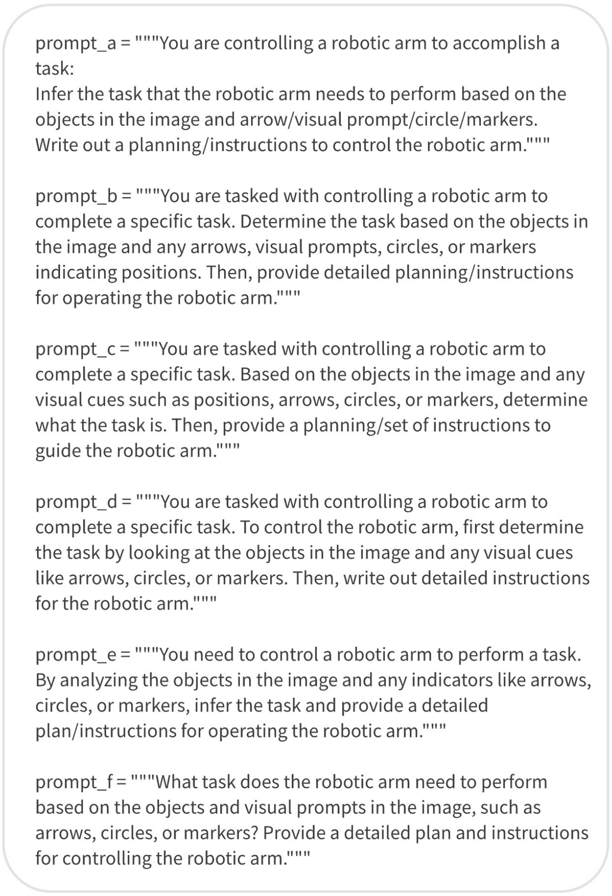
3. Datasets

To train a visual-language model, we curate a new dataset and assign it as D_v. The corresponding equation can be expressed as:

$$D_v = \left\{ \left(o_i, \{p_i^{(k,1)}, p_i^{(k,2)}, \dots, p_i^{(k,n)}\}, \{d_i^{(1)}, d_i^{(2)}\}, \{a_i^{(1)}, a_i^{(2)}, \dots, a_i^{(m)}\} \right) \right\}_{i=1}^{N}, \quad (1)$$

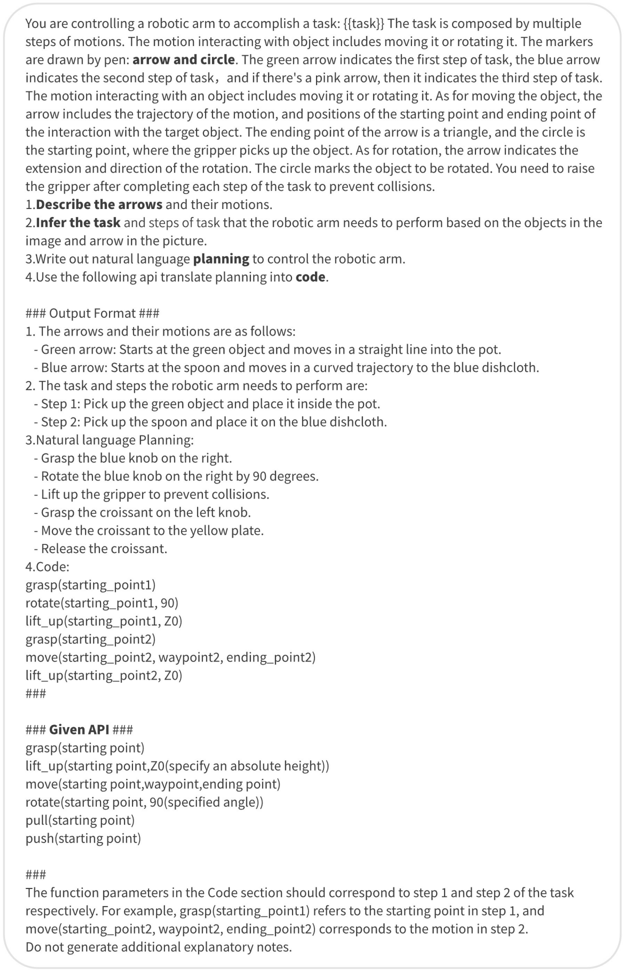
where:

- o_i represents the i-th task observation, where $i = 1, 2, \dots, N$, and N is the total number of task observations.
- $\{p_i^{(k,1)}, p_i^{(k,2)}, \dots, p_i^{(1,n)}\}$ is the set of n RoVI paths us-



```
prompt_a = """You are controlling a robotic arm to accomplish a task:
Infer the task that the robotic arm needs to perform based on the objects in the image and arrow/visual prompt/circle/markers.
Write out a planning/instructions to control the robotic arm."""

prompt_b = """You are tasked with controlling a robotic arm to complete a specific task. Determine the task based on the objects in the image and any arrows, visual prompts, circles, or markers indicating positions. Then, provide detailed planning/instructions for operating the robotic arm."""

prompt_c = """You are tasked with controlling a robotic arm to complete a specific task. Based on the objects in the image and any visual cues such as positions, arrows, circles, or markers, determine what the task is. Then, provide a planning/set of instructions to guide the robotic arm."""

prompt_d = """You are tasked with controlling a robotic arm to complete a specific task. To control the robotic arm, first determine the task by looking at the objects in the image and any visual cues like arrows, circles, or markers. Then, write out detailed instructions for the robotic arm."""

prompt_e = """You need to control a robotic arm to perform a task. By analyzing the objects in the image and any indicators like arrows, circles, or markers, infer the task and provide a detailed plan/instructions for operating the robotic arm."""

prompt_f = """What task does the robotic arm need to perform based on the objects and visual prompts in the image, such as arrows, circles, or markers? Provide a detailed plan and instructions for controlling the robotic arm."""
```

Figure 5. Default prompt in RoVI Book. The prompts in the RoVI Book dataset are specifically designed as system-provided default prompts. They are randomly integrated across samples within the dataset.



You are controlling a robotic arm to accomplish a task: {{task}} The task is composed by multiple steps of motions. The motion interacting with object includes moving it or rotating it. The markers are drawn by pen: **arrow and circle**. The green arrow indicates the first step of task, the blue arrow indicates the second step of task， and if there's a pink arrow, then it indicates the third step of task. The motion interacting with an object includes moving it or rotating it. As for moving the object, the arrow includes the trajectory of the motion, and positions of the starting point and ending point of the interaction with the target object. The ending point of the arrow is a triangle, and the circle is the starting point, where the gripper picks up the object. As for rotation, the arrow indicates the extension and direction of the rotation. The circle marks the object to be rotated. You need to raise the gripper after completing each step of the task to prevent collisions.
1.**Describe the arrows** and their motions.
2.**Infer the task** and steps of task that the robotic arm needs to perform based on the objects in the image and arrow in the picture.
3.Write out natural language **planning** to control the robotic arm.
4.Use the following api translate planning into **code**.

```
### Output Format ###
1. The arrows and their motions are as follows:
  - Green arrow: Starts at the green object and moves in a straight line into the pot.
  - Blue arrow: Starts at the spoon and moves in a curved trajectory to the blue dishcloth.
2. The task and steps the robotic arm needs to perform are:
  - Step 1: Pick up the green object and place it inside the pot.
  - Step 2: Pick up the spoon and place it on the blue dishcloth.
3.Natural language Planning:
  - Grasp the blue knob on the right.
  - Rotate the blue knob on the right by 90 degrees.
  - Lift up the gripper to prevent collisions.
  - Grasp the croissant on the left knob.
  - Move the croissant to the yellow plate.
  - Release the croissant.
4.Code:
grasp(starting_point1)
rotate(starting_point1, 90)
lift_up(starting_point1, Z0)
grasp(starting_point2)
move(starting_point2, waypoint2, ending_point2)
lift_up(starting_point2, Z0)
###

### Given API ###
grasp(starting point)
lift_up(starting point,Z0(specify an absolute height))
move(starting point,waypoint,ending point)
rotate(starting point, 90(specified angle))
pull(starting point)
push(starting point)

###
```

The function parameters in the Code section should correspond to step 1 and step 2 of the task respectively. For example, grasp(starting_point1) refers to the starting point in step 1, and move(starting_point2, waypoint2, ending_point2) corresponds to the motion in step 2.
Do not generate additional explanatory notes.

Figure 6. System Default Prompt for off-the-shelf VLMs **Zero-Shot Learning** and RoVI Book **Data Generation.**

ing the k_{th} arrow style for task o_i. In this paper, we design 2 styles of RoVI (k=2).

- $\{d_i^{(1)}, d_i^{(2)}\}$ represents the two default prompts shown in Figure 5, randomly selected from a set of h possible prompts, i.e., $d_i^{(l)} \in \{d_1, d_2, \dots, d_h\}$ for $l = 1, 2$.
- $\{a_i^{(1)}, a_i^{(2)}, \dots, a_i^{(m)}\}$ represents the set of m planning and action sequences associated with the task observation o_i.

Thus, each task observation includes two arrow styles, each with n paths, along with two default prompts and multiple action sequences. We also show the ratios of each task in D_v in Figure 3. Notably, our dataset is modified from Open-X Embodiment [2] shown in Figure 4.

4. Zero-shot Learning

This section shows the structure of the system prompt shown in Figure 6 which is used to provide VLMs in View some initial knowledge. The design of the system prompt follows the principle of chain-of-thought (CoT) reasoning. It involves:

1. **Positional Analysis of RoVI Symbols**: Precisely interpreting the spatial positions and relationships of RoVI components (starting point, waypoints, endpoint, and center) to extract actionable insights.
2. **Task Language Instruction Prediction**: Generating high-level language-formed task definitions for robotic manipulation from visual cues.
3. **Fine-Grained Planning**: Decomposing tasks into detailed, sequential plans to ensure clarity and precision in execution.
4. **Code Function Inference and Output**: Translating planning steps into executable Python code functions, enabling direct control of the robotic system.

We also show the real outputs of VLMs in Figure 9 generated by LLaVA-13B (fine-tuned), which demonstrate that

RoVI is easy to understand and the RoVI Book is effective. The details of LLaVA in VIEW are shown in Figure 8.

5. Challenges and Optimizations in the Design Process

Challenges. During initial testing, we identify several key issues leading to task misidentification when using robotic visual instruction with arrow-based instructions for the VLM:

1. The model often fails to correctly distinguish the arrow's direction, starting point, and endpoint.
2. Curved arrows cause confusion between rotational tasks and curved 'move to' trajectories.
3. The VLM struggles with spatial orientation, frequently confusing left and right.
4. The accuracy in recognizing the start and end points of the arrow remains low, leading to incorrect semantic interpretation of the arrow.

To address these issues:

1. For zero-shot learning, we implement **Knowledge Augmentation** in prompt, and **Geometric Enhancement** within the RoVI input space, effectively resolving issues 1, 2, and 4.
2. We designed and trained a keypoint module to address issues 3 and 4 (see Section Keypoint Module).

Knowledge Augmentation. We refine the default prompt by enriching contextual information about arrows, enabling better spatial semantic understanding in robotic tasks. Adding detailed descriptions of the arrow's appearance and spatial semantics in the prompt:

The arrow includes the motion trajectory, with starting and ending points of interaction with the target object. For rotation tasks, it represents the extent and direction of rotation.

The arrowhead typically has a sharp angle or triangular shape, indicating the 'pointed' end, while the tail is wider and straight or slightly curved, representing the 'base'.

Geometric Enhancement. Decomposing the arrow into easily recognizable geometric components with clear semantic meanings. Simple shapes like triangles and circles are more readily recognized by visual encoders due to their distinct features and prevalence in pretraining datasets. Different visual designs are applied for 'move to' and 'rotate' tasks: in 'move to', a circle indicates the affordance location; in 'rotate', it represents the rotation center. Emphasizing the geometric features of the arrow's head and tail to improve recognition:

The arrow's endpoint is represented by a green triangle, and a green circle at the starting point indicates where the gripper picks up the object.

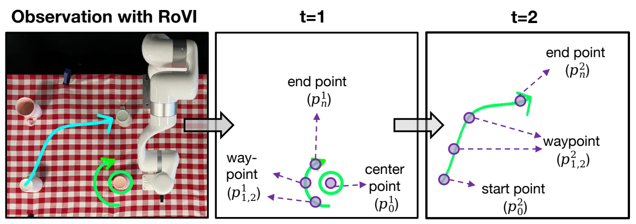


Figure 7. Visualization of post-process in our Keypoint Module.

6. Keypoint Post-process Details

In fact, the pure and trained YOLOv8 (Keypoint Module f_δ) in the main paper does not always exhibit stable performance. However, due to its low latency capabilities, we adopted it and devised an additional post-processing strategy to enhance the Keypoint Module. Our design draws inspiration from two observations.

The first observation is that Keypoint Module is susceptible to environmental disturbances. When the content of input images $X \in \mathbb{R}^{H\times W\times 3}$ is cluttered, the Keypoint Module's performance is adversely affected. The second observation is that Keypoint Module can only recognize green RoVI symbols and fails to detect other colors. The reason is that the color of most RoVI symbols is green in our keypoint training dataset. Thereby, Keypoint Module struggles to extract keypoints in long-horizon tasks that contain multiple RoVI symbols in different colors.

To address these issues, we have designed two corresponding methods. For the first issue, we simply transform all pixels except for the RoVI color into white, akin to the green screen. This allows the Keypoint Module to focus solely on the pure RoVI symbols, thereby achieving more robust performance. The second method is based on the first. we separately transform the colors of different step symbols into the first step color green. We then repeat the first method to eliminate other irrelevant content colors. In this way, the vanilla Keypoint Module can be more robust when facing different environments and divide tough long-horizon tasks into multiple and simple single-step tasks. To formalize the above process, we define two functions: color-filter function $F_c(X', color_j)$, and color-transform function $T_c(X')$.

$$F_c(X'_i, color_j) = \begin{cases} X'_i, & if\, X'_i == (color_j) \\ (255,255,255), & else \end{cases} , \tag{2}$$

where $color_j$ means the RoVI color of jth step in the current task. X' means the combination of X and RoVI v, X'_i means the RGB value of each pixel in X', and $i \in \{1, ..., H \times W\}$.

$$T_c(X'_i) = \begin{cases} X'_i, & if X'_i == (0,255,94) \\ (0,255,94), & else \end{cases} . \tag{3}$$

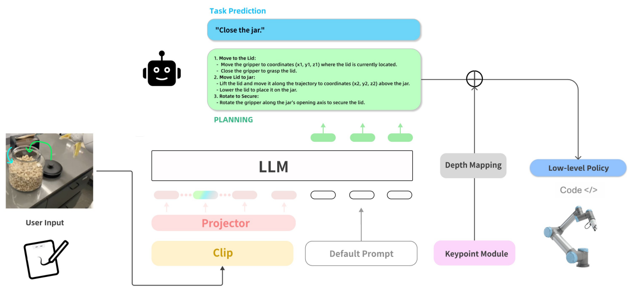

Task Prediction
"Close the jar."
1. Move to the Lid:
- Move the gripper to coordinates (x1, y1, z1) where the lid is currently located.
- Close the gripper to grasp the lid.
2. Move Lid to Jar:
- Lift the lid and move it along the trajectory to coordinates (x2, y2, z2) above the jar.
- Lower the lid to place it on the jar.
3. Rotate to Secure:
- Rotate the gripper along the jar's opening axis to secure the lid.
PLANNING
LLM
Projector
Clip
Default Prompt
Depth Mapping
Keypoint Module
Low-level Policy
Code </>
User Input

Figure 8. Overview of VIEW (LLava (RoVI Book)).

The **arrows** and their **motions**:
- **Green arrow**: The green arrow starts at the cup with a green circle and indicates a rotation. The rotation appears to be counterclockwise.
- **Blue arrow:** The blue arrow starts from the rotated cup (on the left) and moves in a curved trajectory towards a cup on the right.

The **task and steps** the robotic arm needs to perform are:
- Step 1: Rotate the cup marked with the green circle (on the left) counterclockwise.
- Step 2: Pick up the rotated cup and move it along the curved trajectory towards the cup on the right.

Subgoal Planning:
- Grasp the cup marked with the green circle.
- Rotate the cup counterclockwise.
- Move the cup along the blue arrow's trajectory towards the cup on the right.
- Release the cup in the new position.

Code
- grasp(starting_point1)
- rotate(starting_point1, degree, axis)
- move(starting_point2, waypoint2, endpoint2)

Figure 9. VLMs output example.

Thus, for each step of keypoint extraction, the process can be expressed as:

$$P^j_{\{1,\dots,n\}} = f_\delta(T_c(F_c(X', color_j))). \tag{4}$$

We take a two-step task as an example and visualize this process in Figure 7.

7. Skill Definition and Execution Function

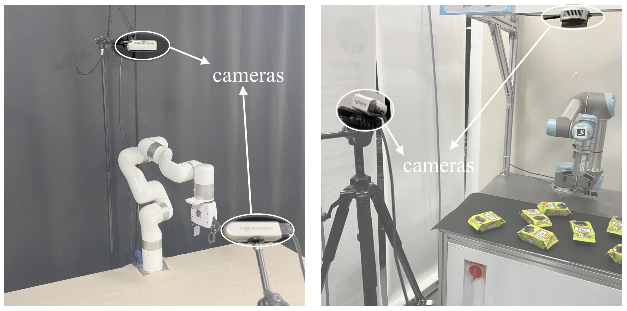
In our formulated policy, manipulation skills serve as the primitives for implementing tasks, primarily consisting of *grasp*, *move*, *rotate*, and *lift*. The execution function is a code-based action that maps skills to corresponding Python functions:

grasp (): The grasp module takes an RGB-D image as input and generates multiple grasp pose candidates. Based on the coordinates of the starting point, it selects the nearest grasp pose e_0.

move (): This function moves the end effector from the starting point to the endpoint, sequentially passing through defined waypoints. Motion planning is employed to compute the optimal collision-free path between keypoints.

rotate (): After grasping, the target rotation angle is calculated based on the angles between the vectors from the center to the starting and ending points. The end effector then performs the required rotation.

lift (): For pick-up tasks, after *grasp ()*, the end effector moves a certain distance along the z-axis to lift the object. In multi-step tasks, motion planning ensures collision avoidance during transitions between steps.



(a) X-Arm6 Real-world setting (b) UR5 Real-world setting

Figure 10. Real-world environment setting.

8. Experiment Environment Setting

8.1. Real World Setting

To verify our framework could be applied to different settings, we use two robotic arms both with two-finger grippers for real-world experiments: UFACTORY `X-Arm 6` and `UR5`. The details are shown in Figure 10.

In the X-Arm 6 real-world setting Figure 10 (a), two calibrated RealSense D435 cameras are positioned for top-down and third-person views with a resolution of 640 $\times$ 480. Both robotic arms operate at a 20 Hz control frequency with an end-effector delta control mode.

In the UR-5 real-world setting Figure 10 (b), we investigate this method using a single-arm platform, which consists of a UR5 robotic arm mounted on a cabinet. At the end of the arm, an OnRobot RG2 parallel gripper is attached. An Intel RealSense L515 camera, with a resolution (color aligned to depth) of 1280$\times$720, is mounted on the ceiling above the workspace to capture an overhead view. Another D435i camera is positioned in the third-person view.

8.2. Simulation Setting

We use SAPIEN [9] as the simulator and SIMPLER [4] as the base environment. We replace the background referred to SIMPLER [4] with a 'green screening' approach to adapt to the Octo [8], thus simulating the real-world scenario. We choose the Google robot and Window-X for our agents. The

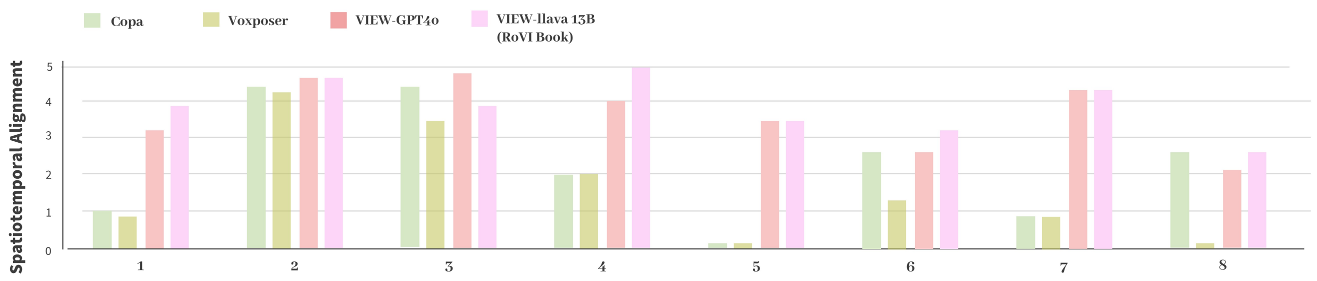


Figure 11. Spatio-temporal performance in in-the-wild experiments. The results of most samples show our methods' superiority.

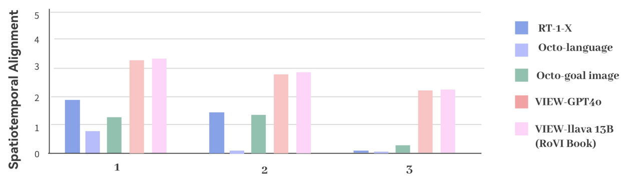


Figure 12. Spatio-temporal performance in SIMPLER. Results also demonstrate the effectiveness of VIEW and RoVI.

control process involves adjusting the arm's position and orientation with interpolation by a planner, combined with controlling the gripper's joint position.

9. Spatio-temporal Alignment

Metric. This explains the spatiotemporal scale used for evaluation. The action video results were assessed by 20 human evaluators, who reported their perceived spatial alignment using a 6-point Likert scale. The scale is defined as follows: 0 - Completely incorrect; 1 - Passed some parts of the trajectory but did not reach the correct end state; 3 - Successfully reached the end state with semantically correct but spatially inaccurate positioning; 4 - Successfully reached the end state, which was mostly accurate, though the trajectory was not precise; 5 - Successfully reached the end state with an accurate trajectory. We show the visualization of this process in Figure 13.

Result. We show the spatiotemporal alignment experiment results in Figure 11 and Figure 12. The VIEW method significantly outperforms language-instruction-based baselines in spatiotemporal performance, particularly in tasks requiring spatial precision and distinguishing disturbances. The largest performance gaps are observed in trajectory-following tasks (e.g., real-1, real-5) and tasks involving disturbances objects (e.g., real-4, sim-2). These results highlight the limitations of language-conditioned policies, which lack the capability to follow trajectories during motion and often fail to precisely satisfy spatial requirements at the end state. Comparisons between Octo-language and Octo-goal images show that goal images provide better spatial alignment at the end state. However, due to the absence of constraints during the motion process, their performance is inferior to VIEW.

10. User Study

During the process of using a goal image as input in all tasks, we faced significant challenges related to generating the goal image, which serves as a crucial input for the model. This process involve manually setting objects in the scene to the desired end state and capturing the input image, which was not only time-consuming but also added complexity to the user experience. When using language as an instruction in cluttered environments, it becomes challenging to accurately describe which target object to select amidst the distractions. Long and detailed descriptions are often required to specify the location of the target. As a result, in methods like RT-1 [1] and Octo [8], users spend more time formulating precise language descriptions for the target's location. In contrast, RoVI offers a more user-friendly and efficient experience; users simply need to circle the selected object with a stylus, significantly reducing the effort and time required for input.

11. Keypoint Module Ablation Experiments

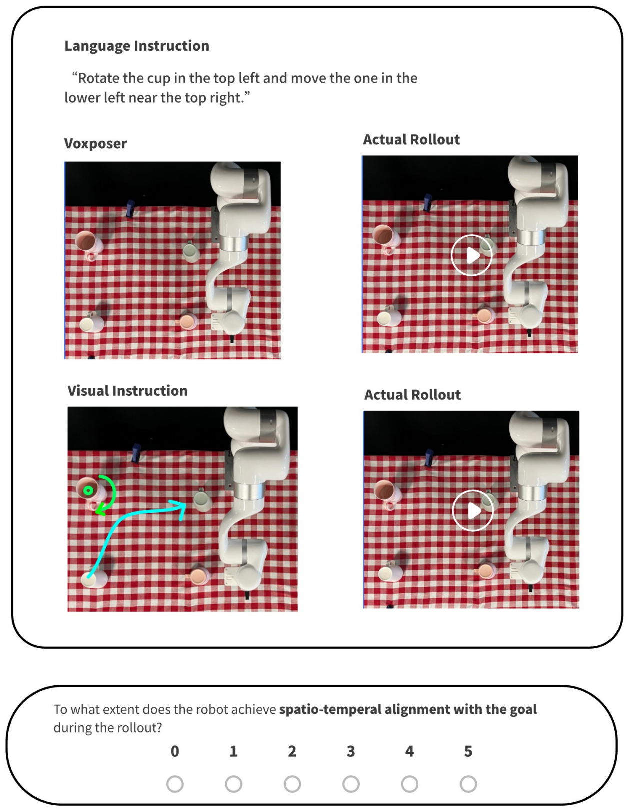
11.1. Experimental Setup

Test Samples with RoVI. The experiment utilizes four test images **1-4**. The RoVI (arrows and circles) contains multiple keypoints. We show the visualizations of test image details in Figure 14.

Model Configuration. Several representative grounding models were evaluated:

- **Grounding DINO [5]**: SwinT backbone with OGC configuration.
- **OWL-ViT [6]**: Base model with patch32 configuration.
- **OWL-V2 [7]**: Base model with patch16-ensemble configuration.
- **YOLOv8 (ours) [3]**: Specialized detection model.

Text Prompts Multiple text prompts are designed for the above open-vocabulary detection models (Grounding DINO [5], OWL-ViT [6], and OWL-V2 [7]). Text prompts include these aspects:


Language Instruction

"Rotate the cup in the top left and move the one in the lower left near the top right."

To what extent does the robot achieve **spatio-temperal alignment with the goal** during the rollout?

0	1	2	3	4	5
○	○	○	○	○	○

Figure 13. Evaluation Questionnaire: This interface is used to evaluate the spatial-temporal alignment of the robot's execution compared to the provided visual instruction and reference goal. Participants are tasked with rating the robot's performance, where they are randomly assigned different skills and environments. The results of these evaluations are presented in this section 9.

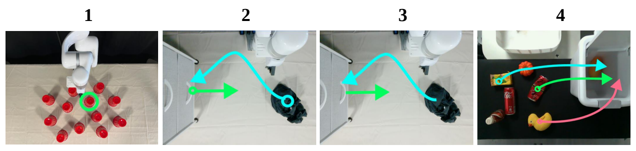


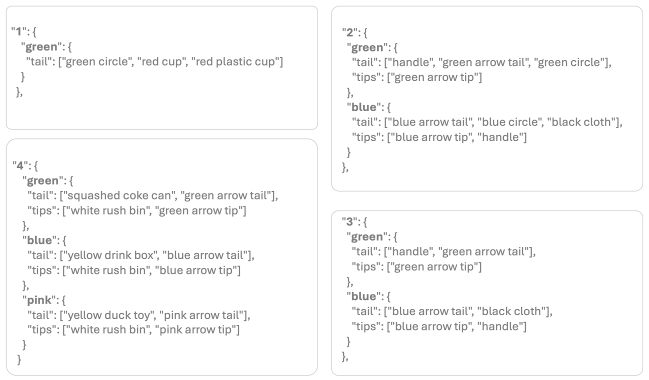
Figure 14. Test samples adopted in the keypoint module ablation study.

- RoVI-component-centric descriptions (e.g., 'green arrow tail')
- Object-centric descriptions (e.g., 'handle', 'squashed coke can', 'red cup', 'yellow duck toy')

We provide all the text prompts for each image in the JSON format. The details are shown in Figure 15.

11.2. Evaluation Metrics

The experiment is conducted with 10 evaluation runs for each model. The results are calculated by two metrics, Euclidean Distance and mAP score. For the distance metric, we calculate the predicted keypoint coordinates with the ground-truth coordinates. For the bounding box output, we adopt the center point coordinates. For the evaluation of mAP scores, we defined that if the distance is less than 50, the mAP will be 1. Otherwise, mAP will be 0. Finally, we calculate the mean and standard deviation to assess performance variability.


```
"1": {
  "green": {
    "tail": ["green circle", "red cup", "red plastic cup"]
  }
},
```

```
"2": {
  "green": {
    "tail": ["handle", "green arrow tail", "green circle"],
    "tips": ["green arrow tip"]
  },
  "blue": {
    "tail": ["blue arrow tail", "blue circle", "black cloth"],
    "tips": ["blue arrow tip", "handle"]
  }
},
```

```
"3": {
  "green": {
    "tail": ["handle", "green arrow tail"],
    "tips": ["green arrow tip"]
  },
  "blue": {
    "tail": ["blue arrow tail", "black cloth"],
    "tips": ["blue arrow tip", "handle"]
  }
},
```

```
"4": {
  "green": {
    "tail": ["squashed coke can", "green arrow tail"],
    "tips": ["white rush bin", "green arrow tip"]
  },
  "blue": {
    "tail": ["yellow drink box", "blue arrow tail"],
    "tips": ["white rush bin", "blue arrow tip"]
  },
  "pink": {
    "tail": ["yellow duck toy", "pink arrow tail"],
    "tips": ["white rush bin", "pink arrow tip"]
  }
}
```

Figure 15. Pesudo code of text prompts for each image. These texts will be used to prompt open-vocabulary detection models.

$$\text{Euclidean Distance} = \sqrt{(x_{\text{pred}} - x_{\text{gt}})^2 + (y_{\text{pred}} - y_{\text{gt}})^2} \tag{5}$$

$$\text{mAP Score} = \begin{cases} 1.0 & \text{if distance} < 50 \text{ pixels} \\ 0.0 & \text{otherwise} \end{cases} \tag{6}$$

11.3. Limitations of open-vocabulary object detectors

1. Applicable only to object-to-object tasks, as they rely solely on object detection and cannot ground non-object regions.
2. Unable to detect waypoints of trajectory.
3. Environmental objects and RoVI annotations interfere with each other, reducing recognition accuracy.

12. Showcase

More detailed demonstrations of robotic manipulation samples are shown in Figure 16 and Figure 17. We also attach video demonstrations in our supplementary zip file.

References

[1] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine,

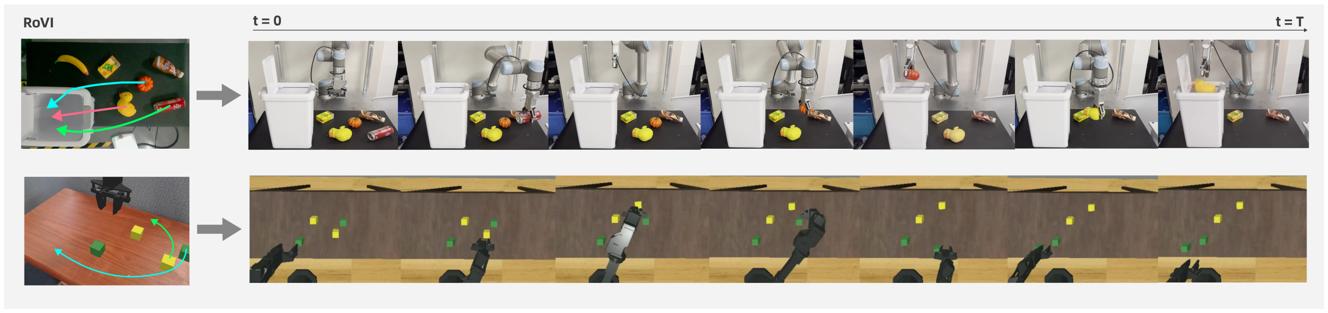


Figure 16. More long-horizon demonstrations of VIEW inputs and outputs in real-world and simulation environments.

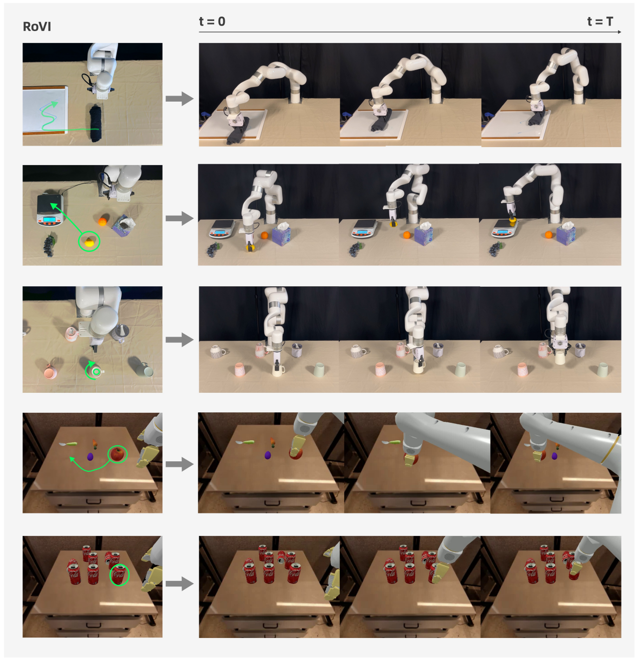


Figure 17. More single-step demonstrations of VIEW inputs and outputs in real-world simulation environments.

Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2212.06817*, 2022. 6

[2] Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-

Mart'in, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. `https://arxiv.org/abs/2310.08864`, 2023. 2, 3

[3] Glenn Jocher, Jing Qiu, and Ayush Chaurasia. Ultralytics YOLO, 2023. 1, 6

[4] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024. 5

[5] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. 6

[6] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. Simple open-vocabulary object detection. In *European Conference on Computer Vision*, pages 728–755. Springer, 2022. 6

[7] Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. Scaling open-vocabulary object detection. *Advances in Neural Information Processing Systems*, 36, 2024. 6

[8] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024. 5, 6

[9] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 5