RoboEXP: Action-Conditioned Scene Graph via Interactive Exploration for Robotic Manipulation

Anonymous CVPR submission

Paper ID *****

Contents

1. Additional Related Works

Neuro-symbolic representations integrates neural networks' perceptual abilities with the symbolic reasoning for robots in complex and dynamic environments. Prior works explored understanding scenes and describing robotic skills in symbolic texts to interpret demonstrations [1, 2], ground abstract actions for robotic primitives [3] and generate action plans [4–7]. Our proposed framework also constructs symbolic representations of the environment, but in the form of action-conditioned scene graphs for robotic manipulation.

Active perception aims to select specific actions for an agent to improve its ability to perceive and understand the environment [8, 9]. Unlike passive perception, actions offer more flexibility, such as control over better viewpoints [10–12], sensor configurations [13, 14], or adjustments to environmental configurations [15]. It can also reveal certain scene properties that cannot be perceived in a passive manner, such as dynamic parameters [16, 17] or articulation [18–20]. Previous studies have explored active perception in 3D reconstruction [21–25], object recognition [26–28], camera localization [29], and robotic manipulation [30, 31]. Our work falls into the category of actively exploring the environment to reveal what's inside or underneath objects. Differing from most previous active perception efforts, which are driven by handcrafted rules [32], information gain [33, 34], or reinforcement learning [16, 35], our approach to active perception is guided by grounding the rich commonsense knowledge encoded in a large language model into an explicit scene graph representation.

Language models for robotics. Large language models (LLMs) [36–38] and large multimodality models (LMMs) [39, 40] are bringing overwhelming influence into the robotics field, for their strong capacity in common-sense knowledge and long-horizon reasoning. Previous studies have harnessed the common-sense knowledge of such large models to generate action candidates [41] and action sequences for task planning [38, 42–44], and generate code for robotic control and manipulation [45–47]. More recently, VILA [48] utilized GPT-4V [39, 40] for vision-language planning. In our RoboEXP system, we leverage GPT-4V for decision-making in two crucial roles. First, as the *action proposer*, it ensures both effectiveness and efficiency in proposing appropriate strategies to expand potential nodes in our action-conditioned 3D scene graph. Second, as the *action verifier*, it ensures the plausibility and smoothness of actions and operations in our system. Moreover, instead of memorizing everything using large models in a brute force way, our system employs explicit memory to enhance the decision-making process.

2. Additional Details of Problem Statement

Due to space constraints, we did not include a comprehensive explanation of the algorithm proposed in the problem statement, but include more details here for clarity. We formulate the interactive scene exploration task into an active perception and exploration problem to construct the action-conditioned 3D scene graph (ACSG).

The algorithm shown in the main paper simply mentions "add spatial relations" and "add action preconditions" as part of the function of the memory module, but without detailed explanation. In the algorithm, we have demonstrated how to construct the edges from objects to actions $\mathbf{e}_{\mathbf{o}\to\mathbf{a}}$ and from actions to objects $\mathbf{e}_{\mathbf{o}\to\mathbf{a}}$; however, there is a lack of description for the other two types of edges.

Add Spatial Relations. The logic involves analyzing the spatial relationships among objects using spatial heuristics and incorporating the resulting spatial relation edges between objects $\mathbf{e}_{\mathbf{o}\to\mathbf{o}}$ (see Algorithm 1).

Algorithm 1 Add Spatial Relations

```
input: G^{t-1} = (V^{t-1}, E^{t-1})
E^t = E^{t-1}
for o ∈ V^{t-1} do                          % check relations
    if relation from o to o_i then          % memory
        E^t = E^t ∪ {e_{o→o_i}}             % add edge
    end if
    if relation from o_i to o then
        E^t = E^t ∪ {e_{o_i→o}}             % add edge
    end if
end for
output: G^t                                 % new scene graph
```

Add Action Preconditions. The approach is to assess the feasibility of implementing the actions. We utilize the decision-making module to verify whether there are any prerequisite actions that need to be completed beforehand, and then adjust the plan accordingly (see Algorithm 2).

Algorithm 2 Add Action Preconditions

```
input: G^{t-1} = (V^{t-1}, E^{t-1}), U^{t-1}
if object o obstruct then                   % decision-making
    choose action a
    V^t = V^{t-1} ∪ {a}, U^{t-1} ∪ {a}      % add node
    E^t = E^{t-1} ∪ {e_{o→a}}               % add edge
    E^t = E^{t-1} ∪ {e_{a→a_k}}             % add edge
end if
output: G^t, U^t                            % new scene graph & plan
```

3. Additional Details of RoboEXP system

In this section, we provide additional details of the decision module and action module. We then discuss our system's design for the interactive scene exploration task and the usage of our system in following sections, focusing on its application in closed-loop exploration processes that may require multi-step or recursive reasoning and handle potential interventions.

3.1. Details of the Modules

Decision-Making Module. As illustrated in the main paper, the decision-making module fulfills two crucial functions within our system. The first function serves as an action proposer (Fig. 1a), proposing the appropriate skill for the query object node. The subsequent role functions as the action verifier (Fig. 1b), tasked with confirming the feasibility of implementing the action and determining the action preconditions. The complete prompts for both roles are detailed in Fig. 1.

Action Module. The action module focuses on providing useful action primitives to aid in constructing our ACSG. We have designed seven action primitives: "open the `[door]`", "open the `[drawer]`", "close the `[door]`", "close the `[drawer]`", "pick `[object]` to idle space", "pick back `[object]`", "move wrist camera to `[position]`". To fully support autonomous actions, we employ a heuristic-based algorithm leveraging geometric cues.

For the door and drawer relevant primitives, engagement with handles is required. In our implementation, we exploit the handle's position and geometry to discern its motion type (prismatic or revolute) and motion parameters (motion axis and motion origin). Executing this action involves utilizing the detected handle and its geometry to adeptly open doors or drawers. Upon identifying the specific handle to be operated, our system retrieves the point cloud converted from our voxel-based representation corresponding to that handle from our memory module. Subsequently, we employ Principal Component Analysis (PCA) to determine the principal direction of the handle, aiding in aligning the gripper for optimal engagement. Additionally, understanding the opening direction is pivotal for effectively handling doors or drawers. To ascertain this, we analyze neighboring points and deduce the most common normal as the opening direction. The combined information of the handle direction and the opening direction provides sufficient guidance for our robot arm to grasp the handle and open the prismatic part. However, in the case of a revolute joint, the motion becomes more intricate. Therefore, we further utilize the motion parameters inferred from the geometry to simulate the evolving opening direction based on the revolute joint's opening process. This well-designed heuristic empowers our system to reliably open drawers or doors in our tabletop setting.

For the pickup-related primitives, we simplify the pickup logic to exclusively consider a top-down direction. Consequently, our focus narrows down to acquiring essential information such as the object's height and xy location. We achieve this by extracting the object's point cloud from its associated voxel-based representation. Subsequently, we pinpoint the highest points within the cloud, calculating their mean to determine the optimal pickup point. This calculated point serves as a precise reference for our gripping mechanism, facilitating the successful grasping of objects in the specified direction.

Regarding viewpoint change, the primitive is parameterized with the expected pose. For example, after opening the door/drawer, to see inside, we develop the heuristic to

(a) Prompts of Proposer

System: You are an assistant tasked with aiding in the construction of a complete scene graph for a tabletop environment. The objective is to identify all objects hidden from the current observation in the tabletop setting. Your role involves selecting appropriate actions or opting not to take any action based on commonsense knowledge in response to queries with current observations. Your responses will guide a robot in efficiently exploring the environment. Approach each step thoughtfully, and analyze the fundamental problem deeply, considering the potential vagueness or inaccuracy in the queries. Adhere to the provided formats in your instructions.

User: Analyze and provide your final answer for each new query object/part category, considering the given surrounding objects and observations in the tabletop scene from different viewpoints. The query object/part will be enclosed in a green bounding box, though it may not always be fully accurate. Format your responses as follows: "[Analysis]: <your reasoning process>; \n\n [Final Answer]: <skill>". Be comprehensive and avoid repeating my question. Choose from three skills: 1. Open the doors or drawers. 2. Pick up / Open the top object. 3. No action. The primary goal is to select an action that has the potential to reveal hidden objects. The secondary goal is to act efficiently, performing only necessary actions to uncover hidden objects. For example, if an object contains doors or drawers and can potentially store something inside, opt for the first skill "Open the doors or drawers". If an object has no bottom side and can potentially cover something beneath it, choose the second skill " Pick up / Open the top object"; otherwise, select the third skill "No action" to ensure efficiency.

Assistant: Got it. I will output the reasoning process step-by-step, explain why I choose the skill but not others and follow the output format.

User: [Query Object] + [Query Images]

Assistant: [Reply from GPT-4V]

(b) Prompts of Verifier

System: You are an assistant tasked with evaluating the feasibility of actions within a tabletop environment. Your role is to select suitable objects that could obstruct open actions based on queries and current observations. Provide guidance for a robot's planning process. Approach each step thoughtfully, analyzing the underlying problem thoroughly while considering potential vagueness or inaccuracy in the queries. Follow the provided formats in your instructions.

User: Provide an analysis and your final answer each time I present a new query object/part category, the list of surrounding objects you need to consider and observations of the corresponding in the tabletop scene from different viewpoints. The query object/part is enclosed in a green bounding box, which may not always be fully accurate. Present your reasoning process and final answer in the format "[Analysis]: <your reasoning process>; \n\n [Final Answer]: <list of objects>". Be comprehensive and avoid repeating my question. Use the given list of surrounding objects, maintaining the provided names. Only consider the surrounding objects in the given list. The objective is to identify all objects that could potentially block open actions. If an object obstructs the door or drawer from opening, include it in the final list of objects. Analyze the action movement and identify the blocking objects.

Assistant: Got it. I will output the reasoning process step-by-step, explain why I choose the object but not others and follow the output format.

User: [Query Object] + [Query Images]

Assistant: [Reply from GPT-4V]

Figure 1. **Prompts of the Decision-Making module.** We present the full prompts for the two pivotal roles of our decision-making module, **proposer** in (a), **verifier** in (b). The prompts are used for all our experiments without modification and extra examples.

System: You are an assistant tasked with aiding in the construction of a complete scene graph for a tabletop environment. The objective is to identify all objects hidden from the current observation in the tabletop setting. Your role involves selecting appropriate actions or opting not to take any action based on commonsense knowledge in response to queries with current observations. Your responses will guide a robot in efficiently exploring the environment. Approach each step thoughtfully, and analyze the fundamental problem deeply, considering the potential vagueness or inaccuracy in the queries. Adhere to the provided formats in your instructions.

User: Analyze and provide the current scene graph and your final answer for the next action given the latest observations in the tabletop scene from different viewpoints. Each time you need to pick an action to do or choose "Done" to terminate. The action you can choose should be composed of (<object/part>, <skill>). Be specific on which object or part you refer to. The skills you can choose: [1. Open the door. 2. Close the door. 3. Open the drawer. 4. Close the drawer. 5. Pick up the object to idle space. 6. Pick back the object from the idle space]. Each time after you choose an action, you will receive the new observations after the action. Format your responses as follows: "[Analysis]: <your reasoning process>; \n\n [Scene Graph]: <current scene graph> \n\n [Final Answer]: <skill>". Be comprehensive and avoid repeating my question. The primary goal is to select an action that has the potential to reveal hidden objects. The secondary goal is to act efficiently, performing only necessary actions to uncover hidden objects. The third goal is to make the object go back to the initial state after exploration. For the output scene graph, you need to output all the objects in the scene, including those found during the exploration process.

Assistant: Got it. I will output the reasoning process step-by-step, explain why I choose the skill but not others and follow the output format.

User: [Query Images]

Assistant: [Reply from GPT-4V]

User: [Query Images]

Assistant: [Reply from GPT-4V]

...

Figure 2. **Prompts of the GPT-4V baseline.** To ensure fairness in comparison to this baseline, we choose to use similar prompts, employing the chain-of-thoughts technique to enhance its performance.

choose the proper viewpoint from the open direction as the parameter for the primitive, allowing for the implementation of the action primitive.

3.2. Other Design in Interactive Exploration

One desiderata for robot exploration is the ability to handle scenarios that necessitate multi-step or recursive reasoning. An example of this is the Matryoshka doll case, which cannot be addressed using previous one-step LLM-based code generation approaches [46, 48]. In contrast, our modular design allows agents to dynamically plan and adapt in a closed-loop manner, enabling continuous LLM-based exploration based on environmental feedback.

To manage multi-step reasoning, our system incorporates an action stack as a simple but effective "planning" module. Guided by decisions from the decision module, the stack structure adeptly organizes the order of actions. For instance, upon picking up the top Matryoshka doll, if the perception

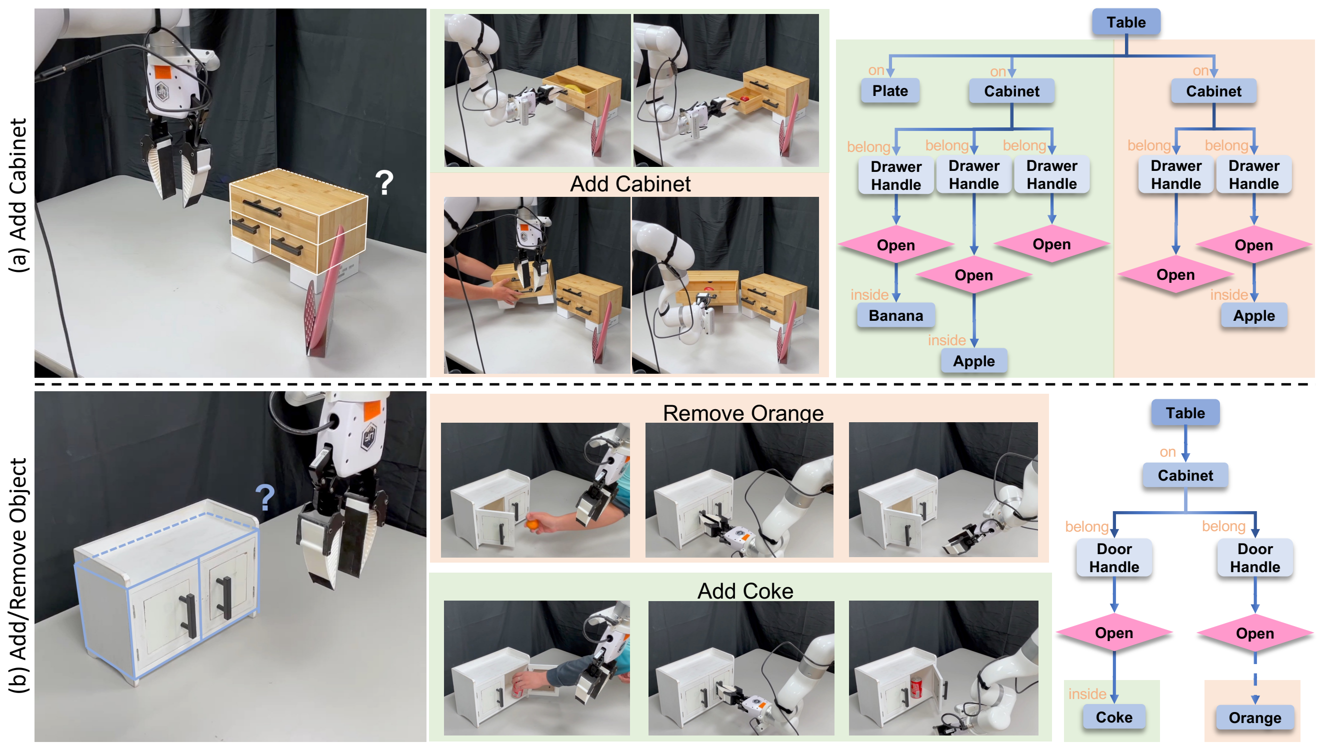

Figure 3. **Qualitative Results on Different Intervention Scenarios.** (a) This scenario involves adding a cabinet to the tabletop setting, and our system can auto-detect the new cabinet and explore the objects inside. (b) This scenario includes removing and adding objects from and into the cabinet. Our system can monitor hand interactions and re-explore the corresponding doors.

and memory modules identify another smaller Matryoshka doll in the environment, the decision module determines to pick it up. Our action stack dynamically adds this pickup action to the top of the stack, prioritizing the new action over picking back the previous, larger Matryoshka doll. This stack structure facilitates multi-step reasoning and constructs the system's logic in a deep and coherent structure.

Moreover, for the interactive scene exploration task, maintaining scene consistency is crucial in practice (e.g., the agent should close the fridge after exploring it). We employ a greedy strategy returning objects to their original states. This approach keeps the environment close to its pre-exploration state, making RoboEXP more practical for real-world applications.

3.3. Usage of ACSG

The ACSG constructed during the exploration stage shows beneficial for scenarios that require a comprehensive understanding of scene content and structure, such as household environments like kitchens and living rooms, office environments, etc. We list several examples illustrating the potential usage of the scene graph in various tasks.

Judging Object Existence. A direct application of our ACSG is to determine the presence or absence of specific objects in the current environment. For instance, during the exploitation stage of the scenario (Sec. 5) to set the dining table, if the spoon is missing, the robot can further seek human assistance.

Object Retrieval. One notable advantage of our ACSG is its ability to capture all actions and their preconditions. Utilizing this information, retrieving any object becomes straightforward by following the graph structure and executing actions in topological order along the paths from the root to the target object node. For example, in the obstruction scenario (Sec. 5), the ACSG can provide the sequence of actions required to fetch the tape: 1) removing the condiment blocking the cabinet door, 2) opening the cabinet via the door handle, and 3) retrieving the tape. Such insights are crucial for tasks like cooking.

Advanced Usage. The high-level representation of the environment provided by our ACSG serves as a simplified yet effective model. Similar to the approach proposed by Gu et al. [49], integrating the scene graph with Large Language Models (LLM) or Large Multi-modal Models (LMM) offers enhanced capabilities, including natural language interaction. This enables the robot to respond to human preferences expressed in natural language (e.g., fetching a coke when the person is thirsty) or through visual cues (e.g., fetching a mug when the table is dirty).

Figure 4. **All Testing Objects.** We present various objects utilized in our work, encompassing different types of cabinets, fruits, dolls, condiments, beverages, food items, tapes, tableware, and fabric.

4. Additional Details of Experiments

4.1. Robot and Environment Setups

All our experiments are conducted in a real-world setting. In these scenarios, we mount one RealSense-D455 camera on the wrist of the robot arm to collect RGBD observations, with the execution of actions performed by the UFACTORY xArm 7. The end effector for our robot arm is the soft gripper. Our experimental setup encompasses a diverse range of objects, as illustrated in Fig. 4. To assess the effectiveness of our system, we devised five types of experiments, each encompassing 10 distinct settings. These settings vary in terms of object number, type, and layout, as illustrated in Fig. 5.

Baseline. We employ the pure GPT-4V as our baseline model along with the chain-of-thoughts (CoT) to enhance its capabilities, as outlined in a method similar to that proposed by Hu et al. [48]. This baseline operates in a closed-loop fashion, receiving three RGB observations from different viewpoints during each iteration. At each turn, it generates the current scene graph, encompassing hidden objects, and suggests the next action to be taken. Upon determining that all tasks are completed, the model outputs "Done" (refer to the complete prompts in the Appendix). To ensure the baseline is robust, we utilize manual actions as ground truth references for the proposed actions. For instance, if the baseline suggests opening a specific drawer, we manually perform the action and prompt the model with the new observation to generate another action. In contrast, in the exploration experiments described below, all actions from our system are automatically executed by our action module on the physical robot. The full prompt of the GPT-4V baseline is illustrated in Fig. 2.

Evaluation. As mentioned in the main paper, we have designed five key metrics. To assess the effectiveness and efficiency of ACSG, we engage human evaluators in the tasks to construct the ground truth version of ACSG. The five main metrics employed for evaluation are as follows:

1) **Success:** This metric evaluates the success percentage across 10 variants for each task. We define success for each experiment as 1 when the final outputted ACSG exactly matches the GT version, and 0 otherwise.

2) **Object Recovery:** This metric quantifies the percentage of hidden objects successfully identified.

3) **State Recovery:** A binary value indicates whether the final state resembles the original state before exploration. This includes considerations for partial states and object positions (e.g., in the top drawer of a cabinet or on the table).

4) **Unexplored Space:** Evaluating the percentage of successfully explored need-to-explore space to reduce the robot's uncertainty about the scene. The identification of the need-to-explore space relies on human annotation.

5) **Graph Edit Distance (GED):** GED measures the disparity between the outputted graph and the GT graph. We adopt a simplified version of GED with six operations—three for nodes (add, delete, edit) and three for edges (add, delete, edit), with each operation incurring a cost of 1.

These metrics provide a comprehensive evaluation of the method's performance. Additionally, we visualize the number of objects and actions during the exploration process to show the exploration strategies employed by different methods.

4.2. Human Intervention

Our RoboEXP system possesses the capability to autonomously adapt to changes in the environment. We employ two types of human interventions to demonstrate these points (refer to Sec. 5).

The first type of intervention (Fig. 3a) involves adding new cabinets to the scene. In this scenario, we add a cabinet to the explored area, allowing our system to automatically explore the newly added cabinets and update the ACSG.

The second type of intervention (Fig. 3b) involves adding new objects to or removing existing ones from the cabinets in the current scene. Our system can monitor human interactions and discern which objects require re-exploration. Subsequently, it autonomously updates the ACSG based on re-exploration.

4.3. Remaining Challenges

Although our system has proven effective, there is room for improvement. The breakdown of the failure rate in the quantitative results suggests that failures primarily arise from detection and segmentation errors within the perception module. To address this issue, we envision two future directions: 1) enhancing the capabilities of visual foundation models for open-world semantic understanding, and 2) utilizing temporal cues and semantic fusion techniques to improve perception robustness through continuous observations.

Furthermore, our system would benefit from enhanced LMM capacities and the integration of sophisticated skill modules, including learning-based or model-based path planning. Such improvements would improve both the decision-making and action modules, thereby further reducing failure cases.

5. Video Timeline

Scenario A. Exploration-Exploitation

Exploration: 00:43 - 01:16

Exploitation: 01:17 - 01:37

Scenario B. Recursive Reasoning

Exploration: 01:49 - 02:26 (Two scenarios)

Scenario C. Obstruction

Exploration: 02:33 - 02:59

Scenario D. Intervention

Exploration: 03:05 - 04:09 (Two scenarios)

CVPR
#*****

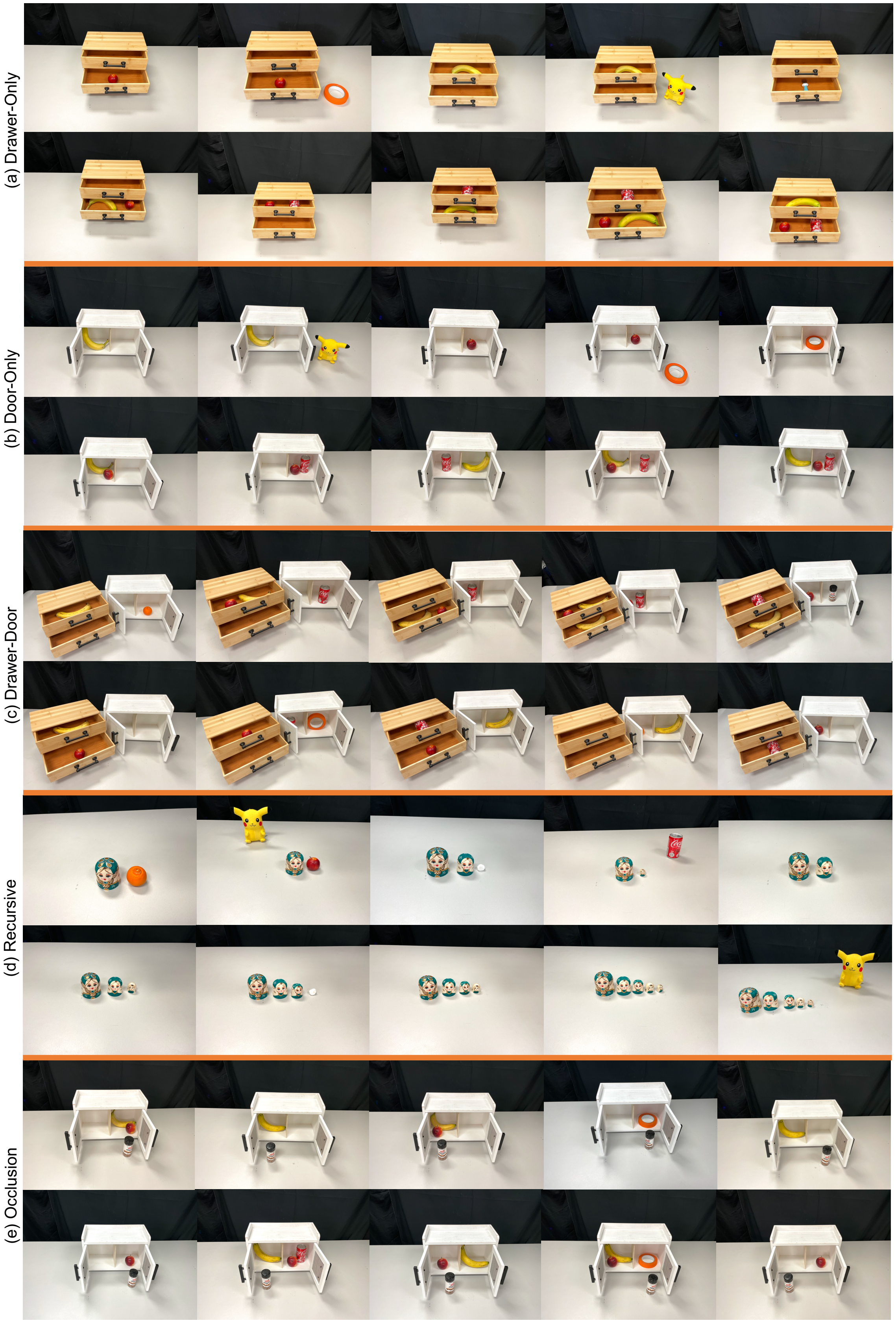

Figure 5. **Experiment Settings.** Varied object numbers, types, and layouts in our experimental settings of the quantitative results.

References

[1] Jiayuan Mao, Tomás Lozano-Pérez, Joshua B Tenenbaum, and Leslie Pack Kaelbling. Learning reusable manipulation strategies. In *CoRL*, 2023. 1

[2] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. Clevrer: Collision events for video representation and reasoning. *arXiv preprint arXiv:1910.01442*, 2019. 1

[3] Renhao Wang, Jiayuan Mao, Joy Hsu, Hang Zhao, Jiajun Wu, and Yang Gao. Programmatically grounded, compositionally generalizable robotic manipulation. *ICLR*, 2023. 1

[4] Zhutian Yang, Jiayuan Mao, Yilun Du, Jiajun Wu, Joshua B. Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Compositional Diffusion-Based Continuous Constraint Solvers. In *CoRL*, 2023. 1

[5] Weiyu Liu, Jiayuan Mao, Joy Hsu, Tucker Hermans, Animesh Garg, and Jiajun Wu. Composable part-based manipulation. In *CoRL*, 2023.

[6] Zhenfang Chen, Kexin Yi, Yunzhu Li, Mingyu Ding, Antonio Torralba, Joshua B Tenenbaum, and Chuang Gan. Comphy: Compositional physical reasoning of objects and events from videos. In *ICLR*, 2022.

[7] Jiayuan Mao, Tomas Lozano-Perez, Joshua B. Tenenbaum, and Leslie Pack Kaelbling. PDSketch: Integrated Domain Programming, Learning, and Planning. In *NeurIPS*, 2022. 1

[8] R. Bajcsy. Active perception. *Proceedings of the IEEE*, 1988. 1

[9] Active perception vs. passive perception. In *Proc. of IEEE Workshop on Computer Vision*, 1985. 1

[10] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon" next-best-view" planner for 3d exploration. In *ICRA*, 2016. 1

[11] Ana Batinovic, Antun Ivanovic, Tamara Petrovic, and Stjepan Bogdan. A shadowcasting-based next-best-view planner for autonomous 3d exploration. *RA-L*, 2022.

[12] Menaka Naazare, Francisco Garcia Rosas, and Dirk Schulz. Online next-best-view planner for 3d-exploration and inspection with a mobile manipulator robot. *RA-L*, 2022. 1

[13] Shengyong Chen, Youfu F Li, Wanliang Wang, and Jianwei Zhang. *Active sensor planning for multiview vision tasks*. 2008. 1

[14] Peihao Chen, Dongyu Ji, Kunyang Lin, Weiwen Hu, Wenbing Huang, Thomas Li, Mingkui Tan, and Chuang Gan. Learning active camera for multi-object navigation. *NeurIPS*, 2022. 1

[15] Mahsa Ghasemi, Erdem Bulgur, and Ufuk Topcu. Task-oriented active perception and planning in environments with partially known semantics. In *ICML*, 2020. 1

[16] Tushar Nagarajan and Kristen Grauman. Learning affordance landscapes for interaction exploration in 3d environments. In *NeurIPS*, 2020. 1

[17] Yian Wang, Ruihai Wu, Kaichun Mo, Jiaqi Ke, Qingnan Fan, Leonidas Guibas, and Hao Dong. AdaAfford: Learning to adapt manipulation affordance for 3d articulated objects via few-shot interactions. In *ECCV*, 2022. 1

[18] Roberto Martín-Martín and Oliver Brock. Building kinematic and dynamic models of articulated objects with multi-modal interactive perception. In *2017 AAAI Spring Symposium Series*, 2017. 1

[19] Neil Nie, Samir Yitzhak Gadre, Kiana Ehsani, and Shuran Song. Structure from action: Learning interactions for articulated object 3d structure discovery. *arXiv preprint arXiv:2207.08997*, 2022.

[20] Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. In *CVPR*, 2022. 1

[21] Christopher Collander, William J Beksi, and Manfred Huber. Learning the next best view for 3d point clouds via topological features. In *ICRA*, 2021. 1

[22] Daryl Peralta, Joel Casimiro, Aldrin Michael Nilles, Justine Aletta Aguilar, Rowel Atienza, and Rhandley Cajote. Next-best view policy for 3d reconstruction. In *ECCV Workshops*. Springer, 2020.

[23] Linghao Chen, Yunzhou Song, Hujun Bao, and Xiaowei Zhou. Perceiving unseen 3d objects by poking the objects. In *ICRA*, 2023.

[24] Muzhi Han, Zeyu Zhang, Ziyuan Jiao, Xu Xie, Yixin Zhu, Song-Chun Zhu, and Hangxin Liu. Reconstructing interactive 3d scenes by panoptic mapping and cad model alignments. In *ICRA*, 2021.

[25] Muzhi Han, Zeyu Zhang, Ziyuan Jiao, Xu Xie, Yixin Zhu, Song-Chun Zhu, and Hangxin Liu. Scene reconstruction with functional objects for robot autonomy. *IJCV*, 2022. 1

[26] Zhirong Wu, Shuran Song, Aditya Khosla, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets for 2.5 d object recognition and next-best-view prediction. *arXiv preprint arXiv:1406.5670*, 2014. 1

[27] Yiheng Han, Irvin Haozhe Zhan, Wang Zhao, and Yong-Jin Liu. A double branch next-best-view network and novel robot system for active object reconstruction. In *ICRA*, 2022.

[28] Björn Browatzki, Vadim Tikhanoff, Giorgio Metta, Heinrich H Bülthoff, and Christian Wallraven. Active in-hand object recognition on a humanoid robot. *IEEE Transactions on Robotics*, 2014. 1

[29] Qihang Fang, Yingda Yin, Qingnan Fan, Fei Xia, Siyan Dong, Sheng Wang, Jue Wang, Leonidas Guibas, and Baoquan Chen. Towards accurate active camera localization. In *ECCV*, 2022. 1

[30] Jun Lv, Yunhai Feng, Cheng Zhang, Shuang Zhao, Lin Shao, and Cewu Lu. Sam-rl: Sensing-aware model-based reinforcement learning via differentiable physics-based simulation and rendering. *RSS*, 2023. 1

[31] Youssef Zaky, Gaurav Paruthi, Bryan Tripp, and James Bergstra. Active perception and representation for robotic manipulation. *arXiv preprint arXiv:2003.06734*, 2020. 1

[32] Quoc V Le, Ashutosh Saxena, and Andrew Y Ng. Active perception: Interactive manipulation for improving object detection. *Standford University Journal*, 2008. 1

[33] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *CVPR*, 2018. 1

[34] Snehal Jauhri, Sophie Lueth, and Georgia Chalvatzaki. Active-perceptive motion generation for mobile manipulation. *arXiv preprint arXiv:2310.00433*, 2023. 1

[35] Steven D Whitehead and Dana H Ballard. Active perception and reinforcement learning. In *Machine Learning Proceedings 1990*. 1990. 1

[36] John Schulman, Barret Zoph, Christina Kim, Jacob Hilton, Jacob Menick, Jiayi Weng, Juan Felipe Ceron Uribe, Liam Fedus, Luke Metz, Michael Pokorny, et al. Chatgpt: Optimizing language models for dialogue. *OpenAI blog*, 2022. 1

[37] R OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[38] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model. In *arXiv preprint arXiv:2303.03378*, 2023. 1

[39] OpenAI. Gpt-4v(ision) system card. *https://cdn.openai.com/papers/GPTV System Card.pdf*, 2023. 1

[40] Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. The dawn of lmms: Preliminary explorations with gpt-4v(ision). *arXiv preprint arXiv: 2309.17421*, 2023. 1

[41] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 1

[42] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022. 1

[43] Jianing Yang, Xuweiyi Chen, Shengyi Qian, Nikhil Madaan, Madhavan Iyengar, David F Fouhey, and Joyce Chai. Llm-grounder: Open-vocabulary 3d visual grounding with large language model as an agent. *arXiv preprint arXiv:2309.12311*, 2023.

[44] Yinpei Dai, Run Peng, Sikai Li, and Joyce Chai. Think, act, and ask: Open-world interactive personalized robot navigation. *arXiv preprint arXiv:2310.07968*, 2023. 1

[45] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *ICRA*, 2023. 1

[46] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023. 3

[47] William Shen, Ge Yang, Alan Yu, Jansen Wong, Leslie Pack Kaelbling, and Phillip Isola. Distilled feature fields enable few-shot language-guided manipulation. In *CoRL*, 2023. 1

[48] Yingdong Hu, Fanqi Lin, Tong Zhang, Li Yi, and Yang Gao. Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning. *arXiv preprint arXiv: 2311.17842*, 2023. 1, 3, 5

[49] Qiao Gu, Alihusein Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, Chuang Gan, Celso Miguel de Melo, Joshua B. Tenenbaum, Antonio Torralba, Florian Shkurti, and Liam Paull. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. *arXiv preprint arXiv: 2309.16650*, 2023. 4