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

Anonymous Author(s)
Affiliation
Address
`email`

Contents

A Additional Details of Interactive Exploration **2**

B Additional Details of RoboEXP System **3**

B.1 Details of Modules in RoboEXP System 3

B.1.1 Perception Module 3

B.1.2 Memory Module 3

B.1.3 Decision-Making Module 4

B.1.4 Action Module 4

B.2 Other Design in Interactive Exploration 5

B.3 Usage of ACSG 6

C Additional Details of Experiments **6**

C.1 Experiment Settings 6

C.2 GPT-4V Baseline 6

C.3 Extreme Illumination and Random Background 7

C.4 Performance on Different LMMs 7

C.5 Human Intervention 8

C.6 Room-Level Household Scenarios 8

D Video Timeline **9**

Submitted to the 8th Conference on Robot Learning (CoRL 2024). Do not distribute.

A Additional Details of Interactive Exploration

Due to space constraints, we did not include the pseudocode of the algorithm proposed in the main paper, 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).

Algorithm 1 Interactive Exploration

```
input: $\mathbf{O}^0, \mathbf{G}^0 = (\mathbf{V}^0, \mathbf{E}^0), \mathbf{U}^0 \leftarrow \mathbf{V}^0$
while $|\mathbf{U}^{t-1}| \neq 0$ do
    if choose object $\mathbf{o}_i \in \mathbf{U}^{t-1}$ then                    % explore object
        add spatial relations (Algorithm 2)                                    % memory
        obtain action $\mathbf{a}$ to explore $\mathbf{o}_i$                   % decision-making
        if action $\mathbf{a} \notin \mathbf{V}^{t-1}$ then
            $\mathbf{V}^t, \mathbf{U}^t = \mathbf{V}^{t-1} \cup \{\mathbf{a}\}, \mathbf{U}^{t-1} \cup \{\mathbf{a}\}$    % add node
            $\mathbf{E}^t = \mathbf{E}^{t-1} \cup \{\mathbf{e}_{\mathbf{o}_i \to \mathbf{a}}\}$                    % add edge
            $\mathbf{U}^t = \mathbf{U}^t \setminus \mathbf{o}_i$              % mark as explored
        end if
    else choose action $\mathbf{a}_k \in \mathbf{U}^{t-1}$
        if no obstruction then                                                 % decision-making
            take action $\mathbf{a}_k$                                         % action
            obtain new observation $\mathbf{O}^t$                              % perception
            if found new objects $\mathcal{O} \not\subset \mathbf{V}^{t-1}$ then
                $\mathbf{V}^t, \mathbf{U}^t = \mathbf{V}^t \cup \{\mathcal{O}\}, \mathbf{U}^{t-1} \cup \{\mathcal{O}\}$    % add nodes
                $\mathbf{E}^t = \mathbf{E}^t \cup \{\mathbf{e}_{\mathbf{a}_k \to \mathcal{O}}\}$                  % add edges
                $\mathbf{U}^t = \mathbf{U}^t \setminus \mathbf{a}_k$          % mark as explored
            end if
        else
            add action preconditions (Algorithm 3)                             % memory
        end if
    end if
end while
output: $\mathbf{G}^t$                                                         % final scene graph
```

In the above 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 $\mathbf{e}_{\mathbf{o}\to\mathbf{o}}$ (Algorithm 2).

Algorithm 2 Add Spatial Relations

```
input: $\mathbf{G}^{t-1} = (\mathbf{V}^{t-1}, \mathbf{E}^{t-1})$
$\mathbf{E}^t = \mathbf{E}^{t-1}$
for $\mathbf{o} \in \mathbf{V}^{t-1}$ do                                      % check relations
    if relation from $\mathbf{o}$ to $\mathbf{o}_i$ then                      % memory
        $\mathbf{E}^t = \mathbf{E}^t \cup \{\mathbf{e}_{\mathbf{o} \to \mathbf{o}_i}\}$    % add edge
    end if
    if relation from $\mathbf{o}_i$ to $\mathbf{o}$ then
        $\mathbf{E}^t = \mathbf{E}^t \cup \{\mathbf{e}_{\mathbf{o}_i \to \mathbf{o}}\}$    % add edge
    end if
end for
output: $\mathbf{G}^t$                                                         % 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 (Algorithm 3).

Algorithm 3 Add Action Preconditions

```
input: $\mathbf{G}^{t-1} = (\mathbf{V}^{t-1}, \mathbf{E}^{t-1}), \mathbf{U}^{t-1}$
if object $\mathbf{o}$ obstruct then                                % decision-making
    choose action $\mathbf{a}$
    $\mathbf{V}^t = \mathbf{V}^{t-1} \cup \{\mathbf{a}\}$, $\mathbf{U}^{t-1} \cup \{\mathbf{a}\}$          % add node
    $\mathbf{E}^t = \mathbf{E}^{t-1} \cup \{\mathbf{e}_{\mathbf{o}\to\mathbf{a}}\}$                          % add edge
    $\mathbf{E}^t = \mathbf{E}^{t-1} \cup \{\mathbf{e}_{\mathbf{a}\to\mathbf{a}_k}\}$                        % add edge
end if
output: $\mathbf{G}^t, \mathbf{U}^t$                                 % new scene graph & plan
```

B Additional Details of RoboEXP System

We provide additional details about our system and each module in it. We then discuss our system's design for the interactive scene exploration task, focusing on its application in closed-loop exploration processes that may require multi-step or recursive reasoning and handle potential interventions. Additionally, we explain the usage of our proposed ACSG.

B.1 Details of Modules in RoboEXP System

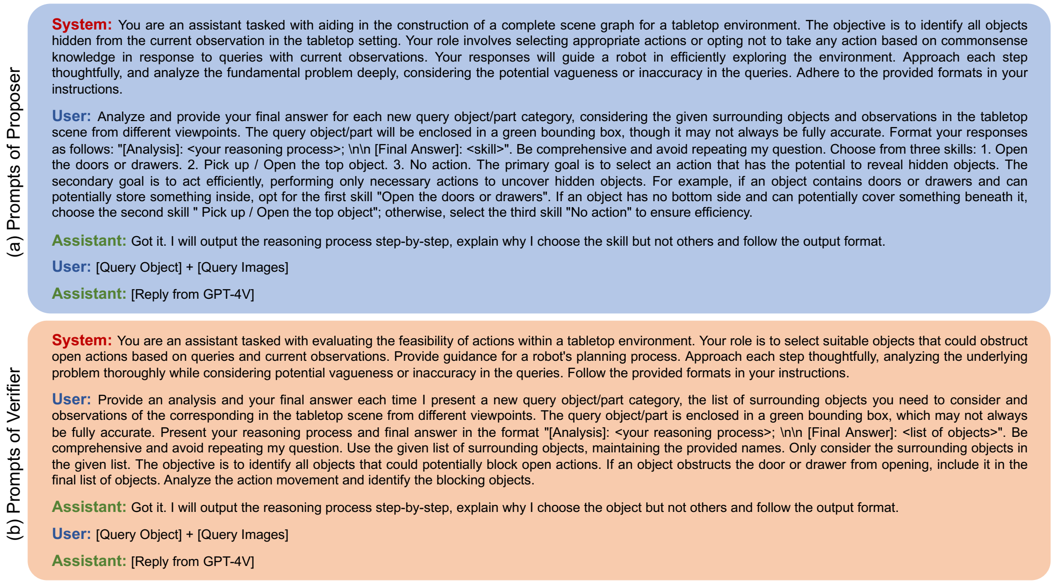
To tackle the interactive exploration task, we present our RoboEXP system, designed to autonomously explore unknown environments by observing and interacting with them. The system comprises four key components: perception, memory, decision-making, and action modules. This closed-loop system ensures the thoroughness of our task in interactive scene exploration.

B.1.1 Perception Module

Raw RGBD images are captured through the wrist camera in different viewpoints and processed by the perception modules to extract scene semantics, including object labels, 2D bounding boxes, segmentations, and semantic features. As mentioned in the main paper, to obtain per-instance CLIP features, we implement a strategy similar to the one proposed by Jatavallabhula et al. [1]. Specifically, we extend the local-global image feature merging approach by incorporating additional label text features to augment the semantic CLIP feature for each instance. Furthermore, we exclusively focus on instance-level features, disregarding pixel-level features, thereby accelerating the entire semantic feature extraction process.

B.1.2 Memory Module

The memory module is designed to construct our ACSG of the environment by assimilating observations over time. For the low-level memory, to ensure stable instance merging from 2D to 3D, we employ a similar instance merging strategy as presented in Lu et al. [2], consolidating observations from diverse RGBD sources across various viewpoints and time steps. In contrast to the original algorithm, which considers only 3D IoU and semantic feature similarity we additionally incorporate label similarity and instance confidence. To enhance algorithm efficiency, we represent low-level memory using a voxel-based representation with filtering designs, which allows for more efficient computation and cleaner memory updates. Meanwhile, given the crowded nature of objects in our tabletop setting, we have implemented voxel-based filtering designs to obtain a cleaner and more complete representation of the objects for storage in our memory.


(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.

B.1.3 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.

We adhere to the standard practice for designing prompts, as other papers do using LLM/LMM [3–5]. In order not to compromise the generalization ability of our system, we consistently use the same prompts across all scenarios and experiments. Our fundamental rule for prompt design is to minimize ambiguity and ensure alignment with our task. In our experiments, the average response time from GPT-4V is about 8 seconds for each question, which is acceptable as GPT-4V is only used in high-level task planning. For low-level motion planning, the use of action primitives allows us to meet the high-frequency requirement without having to constantly querying GPT-4V.

Our ACSG utilizes GPT-4V on every object node to progressively expand the graph. Hence, regardless of how complicated the scene is, each query posed to GPT-4V resides on a local node within our ACSG, essentially addressing the question, “Should I proceed with exploring this object, and if so, how?” As shown in our Matryoshka scenario, RoboEXP performs well in complex scenarios featuring five levels of hierarchical scene graphs and complicated exploration procedures. The commonsense knowledge learned by GPT-4V enables our system to efficiently explore the environment without having to manually design the exploration rules for diverse objects.

B.1.4 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. The input to each action primitive is an object node of our ACSG, from which we can extract all necessary semantic, state, and low-level geometry information of the object.

The integrated information can help us determine the specific grasping pose and path planning for opening and picking actions, which generalize to different instances in various positions and poses.

For opening action primitives related to doors or drawers, 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 choose the proper viewpoint from the open direction as the parameter for the primitive, allowing for the implementation of the action primitive.

B.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 [5, 6]. 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 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.

B.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 (App. D) 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 (App. D), 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. [7], 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).

C Additional Details of Experiments

We conduct experiments in different settings to validate the effectiveness of the model. We provide additional experiments and results, including those with different lighting conditions and backgrounds, using different LMMs, intervention experiments, and several more room-level scenarios.

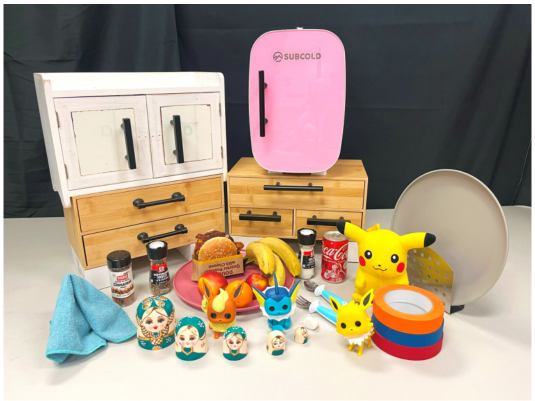


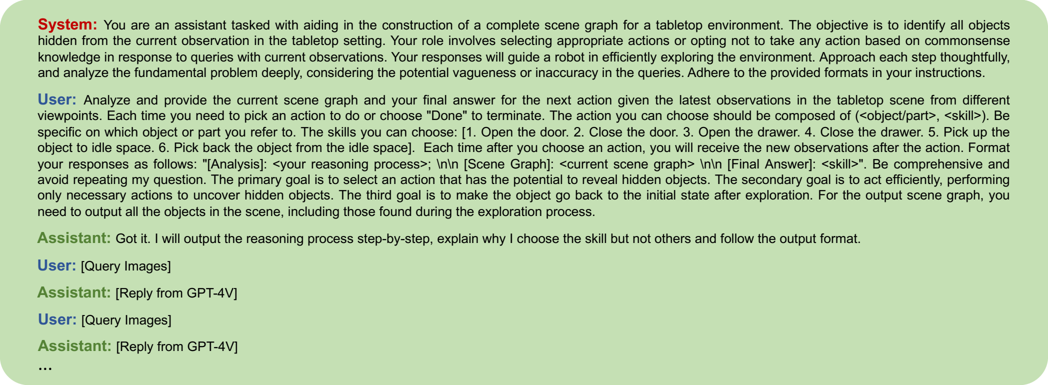
Figure 2: **All Testing Objects.** We present various objects utilized in our work, encompassing different types of cabinets, fruits, dolls, condiments, beverages, food items, tapes, tableware, and fabric.

C.1 Experiment Settings

Our experimental setup encompasses a diverse range of objects, as illustrated in Fig. 2. To assess the effectiveness of our system, we devised five types of experiments for the main quantitative results, each encompassing 10 distinct settings. These settings vary in terms of object number, type, and layout, as illustrated in Fig. 7.

C.2 GPT-4V 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. [6]. The full prompt of the GPT-4V baseline is illustrated in Fig. 3.



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]

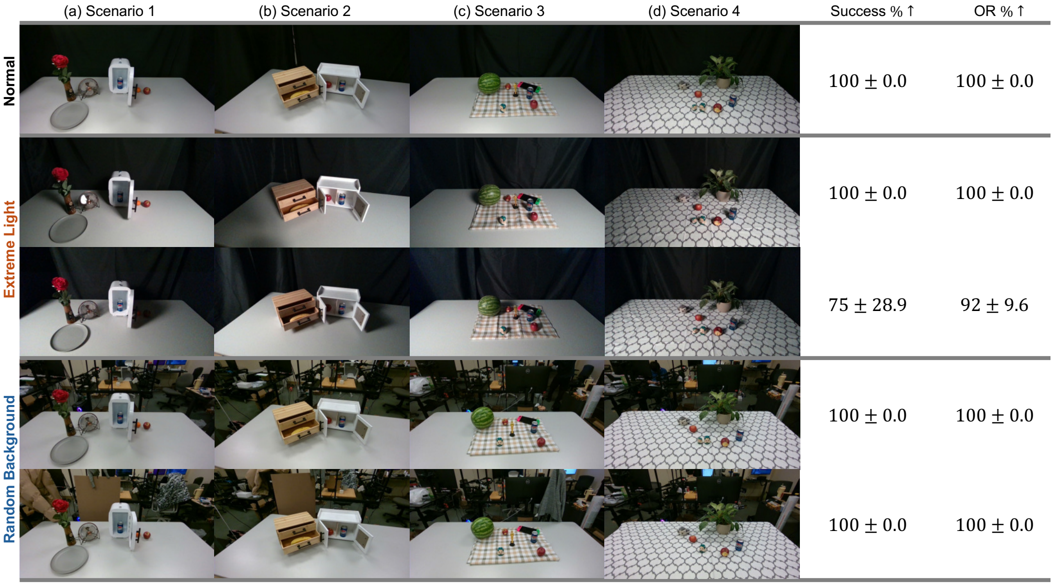
Assistant: [Reply from GPT-4V]

User: [Query Images]

Assistant: [Reply from GPT-4V]

...

Figure 3: **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.



	Success % ↑	OR % ↑
Normal	100 ± 0.0	100 ± 0.0
Extreme Light	100 ± 0.0	100 ± 0.0
	75 ± 28.9	92 ± 9.6
Random Background	100 ± 0.0	100 ± 0.0
	100 ± 0.0	100 ± 0.0

Figure 4: **Experiments on Extreme Illumination and Random Background.** We conduct experiments in four scenarios with varying lighting conditions and random backgrounds. The reported numbers are averages over four scenarios for each condition. Our system performs well across all conditions. (OR refers to our Object Recovery metric).

C.3 Extreme Illumination and Random Background

RoboEXP is robust to extreme lighting conditions and complex backgrounds. To demonstrate this, we tested under four different scenarios, each with varying lighting conditions and random backgrounds. Fig. 4 shows twenty different settings and their corresponding results. In various conditions and scenarios, our system is able to successfully conduct interactive exploration and construct the ACSG, indicating the robustness of RoboEXP to these factors.

C.4 Performance on Different LMMs

RoboEXP is compatible with different multimodal foundation models beyond GPT-4V. We conducted additional experiments using the latest LLaVA-v1.6-34b as the core of our decision module and compared it against GPT-4V under the same settings. Tab. 1 shows that both models can work with our RoboEXP system, yet the capacity of LMMs does influence the overall performance. In general, GPT-4V achieves a higher success rate and more consistent behaviors across different scenarios.

Table 1: **Quantitative Results on Different LMMs.** We conduct experiments with GPT-4V and LLaVA acting as the core of the RoboEXP decision module, under the same fifteen settings as in Fig. 4.

Metric	Success % ↑	Object Recovery % ↑	State Recovery % ↑	Unexplored Space % ↓	Graph Edit Dist. ↓
Ours (LLaVA)	25±25.6	50±29.6	**100**±0.0	23±21.9	2.5±0.98
Ours (GPT-4V)	**95**±12.9	**98**±4.3	**100**±0.0	**0**±0.0	**0.1**±0.26

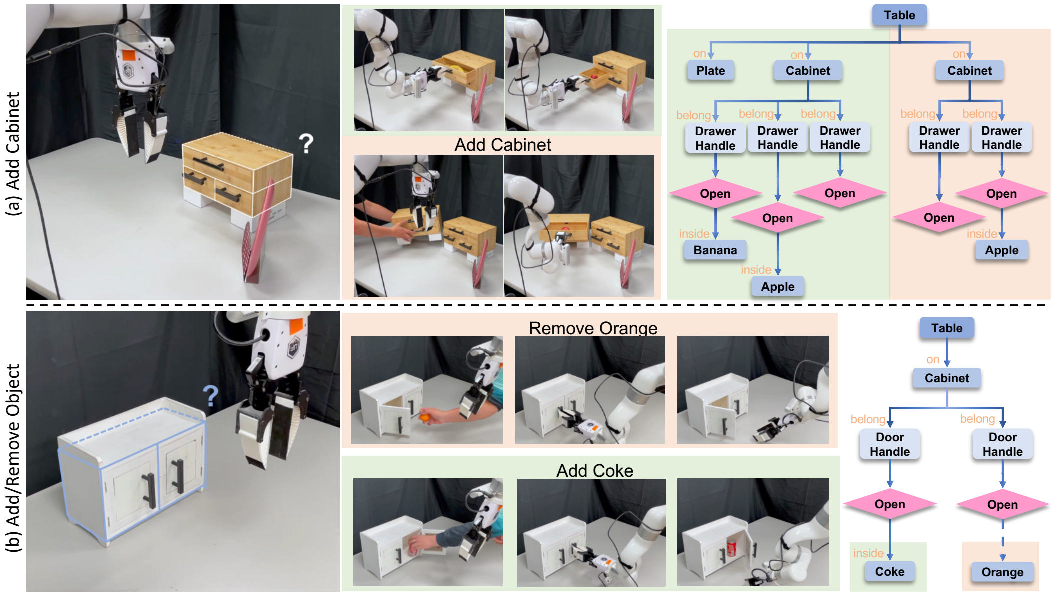


Figure 5: **Qualitative Results on Different Intervention Scenarios.** (a) This scenario involves adding a cabinet to the tabletop setting, and our system can auto-detect the new cabinet and explore the objects inside. (b) This scenario includes removing and adding objects from and into the cabinet. Our system can monitor hand interactions and re-explore the corresponding doors.

C.5 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 App. D).

The first type of intervention (Fig. 5a) 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. 5b) 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.

C.6 Room-Level Household Scenarios

RoboEXP can work well in room-level household environments. To demonstrate this, we conducted two experiments within an apartment (see Fig. 6), specifically in the dining area and bedroom. We integrated four RGB-D observations captured by a handheld RealSense D455 with ICP-based multi-way alignment. Our system successfully constructs corresponding scene graphs within the room-level household environments. Once the static scene graph is constructed, our decision module effectively identifies the correct objects for exploration. Specifically, it accurately identifies the fridge in Fig. 6 (a) and the cabinet in Fig. 6 (b) for further exploration. Tab. 2 shows the complete responses from GPT-4V in our decision module on determining the actions to take in our two household scenarios.

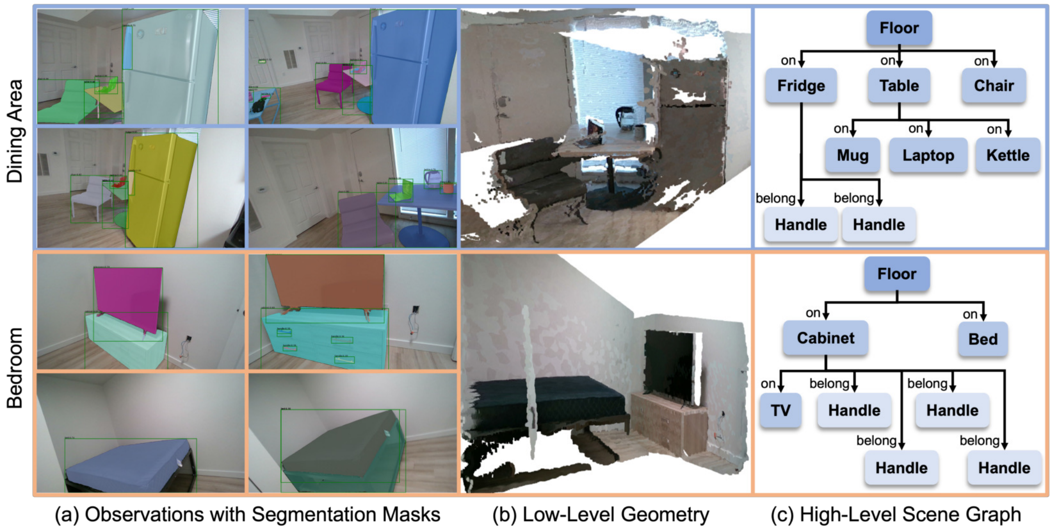


Figure 6: **Experiments on Room-Level Household Scenarios.** We conduct experiments in two room-level household environments. The dining area includes a table, fridge, and items on the table, whereas the bedroom includes a bed, cabinet, and a TV. The figure presents (a) the observations with segmentation masks; (b) the low-level reconstructed geometry; (c) the built high-level scene graph.

D 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)

Extension to Mobile Robot

Exploration: 04:13 - 04:53

References

[1] K. M. Jatavallabhula, A. Kuwajerwala, Q. Gu, M. Omama, T. Chen, S. Li, G. Iyer, S. Saryazdi, N. Keetha, A. Tewari, et al. Conceptfusion: Open-set multimodal 3d mapping. *arXiv preprint arXiv:2302.07241*, 2023.

[2] S. Lu, H. Chang, E. P. Jing, A. Boularias, and K. Bekris. Ovir-3d: Open-vocabulary 3d instance retrieval without training on 3d data. In *CoRL*, 2023.

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

[4] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *ICRA*, 2023.

[5] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

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

[7] Q. Gu, A. Kuwajerwala, S. Morin, K. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, C. Gan, C. de Melo, J. Tenenbaum, A. Torralba, F. Shkurti, and L. Paull. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. *arXiv preprint arXiv: 2309.16650*, 2023.

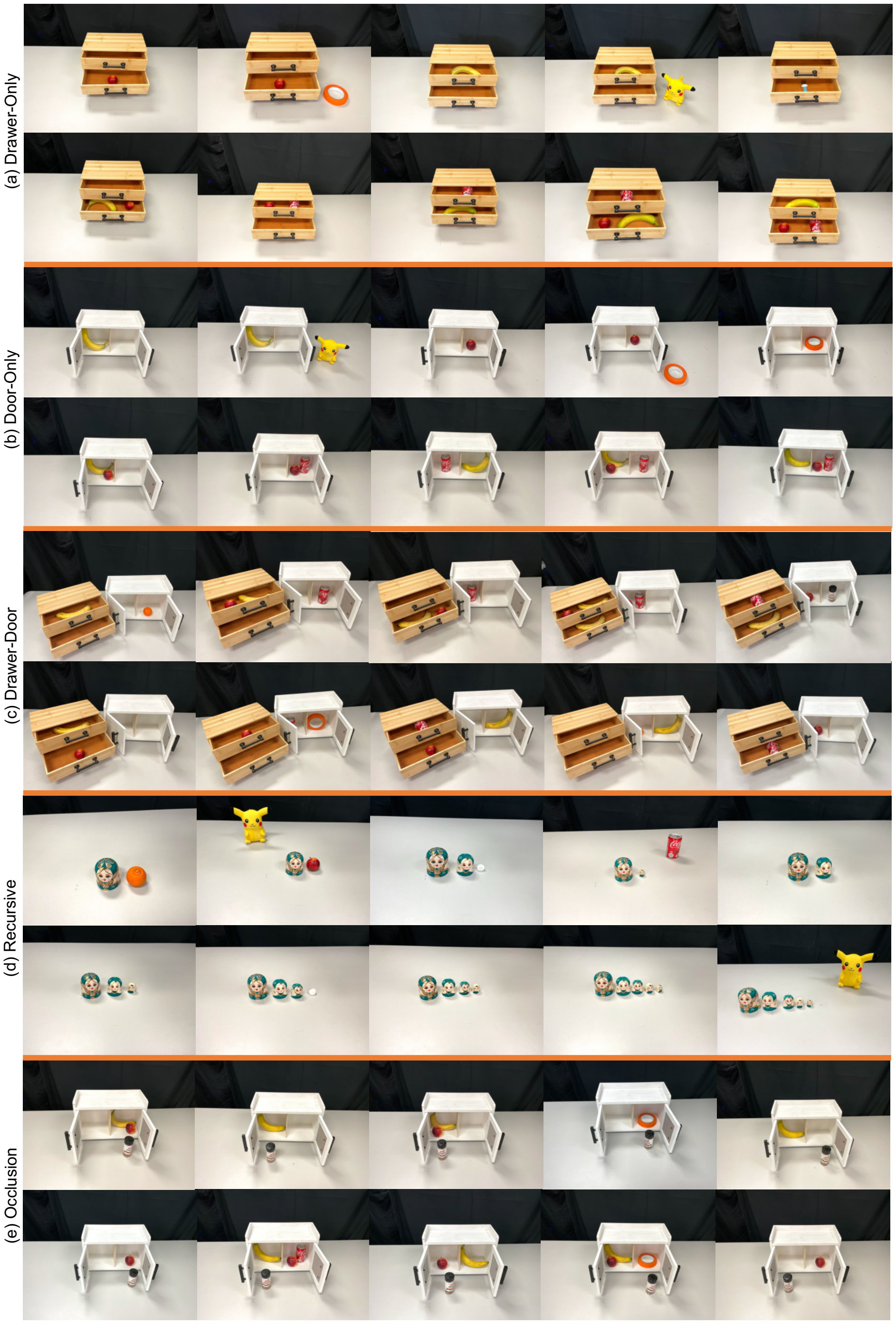


Figure 7: **Experiment Settings.** Varied object numbers, types, and layouts in our experimental settings of the quantitative results.

Table 2: **Decision Module Logs for Household Scenarios.** We present all responses from GPT-4V for our two household scenarios: the dining space case and the bedroom case. The inputs to our decision module are images from various viewpoints of the query object in a green bounding box.

Query Object	LMM Response
Dining Space	
Fridge	**[Analysis]:** The object in question is a fridge, which is typically used to store food and drinks. The green bounding box in the images highlights the fridge, which has doors that can be opened. Since the goal is to identify all objects hidden from the current observation in the tabletop setting, and a fridge is an appliance that commonly contains items inside, it is reasonable to assume that opening the fridge could reveal hidden objects. **[Final Answer]:** Open the doors or drawers.
Chair	**[Analysis]:** The object in question is a chair, which is typically not used to store objects in a way that would require opening or removing parts to reveal them. Chairs are designed for sitting, and while it is possible for items to be placed on or under them, the chair itself does not have compartments that would be opened or closed. The images show the chair from different angles, and there is no indication that there are hidden compartments or objects on the chair that would necessitate an action to uncover them. The chair does not have a top object that would be opened or picked up, and it does not have doors or drawers. **[Final Answer]:** No action.
Table	**[Analysis]:** The object in question is a table, which is highlighted with a green bounding box in the images provided. From the different viewpoints, it is clear that the table has a flat top surface with no visible drawers or doors that could be opened to reveal hidden objects. The table does not appear to have a covering that could hide something underneath it. The table's design is simple, with a single central support and no storage compartments. Therefore, there is no action required that would potentially reveal hidden objects in this case. **[Final Answer]:** No action.
Laptop	**[Analysis]:** The object in question is a laptop, which is placed on a table. Laptops do not have a hidden compartment that would typically contain other objects, nor do they obstruct the view of objects underneath them when they are in an open position, as seen in the images. The laptop is already open, and there is no indication that there is anything on top of it or that it is covering anything beneath it. Therefore, there is no action required that would reveal any hidden objects in this case. **[Final Answer]:** No action.
Mug	**[Analysis]:** The mug in the green bounding box is sitting on a tabletop and does not appear to be covering anything, nor does it have a lid that could be hiding something inside. Mugs are typically open containers and do not have compartments that would conceal objects. Therefore, there is no need to pick it up to look underneath. **[Final Answer]:** No action.
Kettle	**[Analysis]:** The object in question is a kettle, which is sitting on top of a table. Kettles are typically not used to store other items, especially when they are placed in a setting like a tabletop where they are likely used for their intended purpose of boiling water. The kettle does not have drawers or doors, and it is unlikely that it is obscuring any items on the table due to its shape and size. Therefore, there is no need to open any part of the kettle, and it is not obstructing any items underneath it. **[Final Answer]:** No action.
Bedroom	
Cabinet	**[Analysis]:** The object in question is a cabinet, as indicated by the green bounding box in the provided images. Cabinets typically have doors or drawers that can be opened to reveal contents stored inside. In the images, the cabinet appears to have several drawers. Since the primary goal is to select an action that has the potential to reveal hidden objects, and the cabinet has drawers that could contain items, the appropriate action would be to open these drawers. **[Final Answer]:** Open the doors or drawers.
Bed	**[Analysis]:** The object in question is a bed, which is typically not a storage furniture piece with doors or drawers. However, some beds do come with storage options underneath, such as drawers or a lifting mechanism for accessing a storage space. In the provided image, the bed appears to have a solid base without visible drawers or doors. There is no indication that the bed has a mechanism to lift the mattress to reveal storage underneath. Therefore, it is unlikely that there are hidden objects within the bed that can be accessed by opening doors or drawers. **[Final Answer]:** No action.
Television	**[Analysis]:** The object in question is a television, which is placed on top of a surface, likely a media console or a dresser with drawers. Televisions are not typically used to store objects behind or within them, as they are electronic devices designed for display purposes. The images show the television from two different angles, and there is no indication that there are any hidden compartments or objects behind the screen that would be revealed by moving it. Additionally, televisions are generally not designed to have openable parts that would conceal objects. The media console or dresser beneath the television does have drawers, but since the query is specifically about the television, these are not the focus for the action to be taken. **[Final Answer]:** No action.