

## A. Fields of the Object Entry

An object in Persistent Object Memory has the following fields:

- **ID**: The unique object ID in the memory, together with the detected category. Our 3D object re-identification algorithm can be found in [Appendix C](#).
- **STATE**: The object state can be "open", "close", "in hand" or "normal". It is updated by VLM, which will be discussed in [Appendix D](#).
- **Related Objects(RO)**: A list of objects that have "on", "uphold", "in" and "contain" relations with the entry object. The detections of These relations are based on 3D bounding boxes. For example, Given the 3D bounding boxes  $B_1$  and  $B_2$  of object  $O_1$  and  $O_2$  correspondingly, if  $B_1$  has a higher altitude than  $B_2$ ,  $B_1$  has contact with  $B_2$  and  $B_1$  is inside the horizontal surface of  $B_2$ , then  $O_1$  is "on"  $O_2$  and  $O_2$  "upholds"  $O_1$ .
- **3D Bbox**: It is obtained by 2D-3D lifting and dynamically updated by the moving average algorithm. Please refer to [Appendix B](#) and [Appendix C](#) for more details.
- **OBJ Feat**: It is the CLIP feature of the object's cropped image. It is updated by the moving average algorithm. Details are provided in [Appendix C](#).
- **CTX Feat**: It is the CLIP feature of the frame where the object is visible. It is updated by the moving average algorithm. Please refer to [Appendix C](#) for details.

## B. 2D-3D Lifting

In this paper, 2D-3D Lifting refers to getting the 3D bounding boxes of the objects using 2D object detection bounding boxes, camera poses, and depth images. Different from methods that use point clouds or voxels to represent 3D object geometry, we found that representing object geometry as 3D bounding boxes is enough for embodied perception. More importantly, compared to point clouds or voxels, 3D bounding boxes are more memory-efficient and can be maintained and updated easily, which makes them a natural choice for 3D object perception in dynamic scenes.

To get the 3D bounding boxes of the objects, we first use YoloWorld[2] detector to predict the 2D bounding boxes of the objects. SAM-2[40] is then adopted to get the corresponding object masks for the detected objects given the frame and the bounding boxes. For each object, we use its 2D object mask to get its depth pixels and transform them into object surface points in the world coordinate system using the camera intrinsic and extrinsic. We then filter out the bad points (usually, the foreground and the background pixels caused by imperfect segmentation mask prediction) by simply sorting the object surface points by their distances to the camera, and removing the first 10% and the last 10% of the points to finally get the refined object surface points. The object bounding boxes are then computed based on the

minimum and maximum values of the points' coordinates.

## C. Object Re-Identification in Dynamic Scenes

Accurate object re-identification (re-ID) in dynamic scenes can better facilitate embodied perception, task planning, and reasoning. *Embodied VideoAgent* utilizes both object visual features and object 3D bounding boxes for object re-identification. The visual similarity score and the spatial similarity score of an object pair are detailed as follows.

### C.1. Visual Similarity Score

For a detected object on the 2D frame, we crop the object image from the frame using its 2D bounding box and extract the CLIP[39] and DINOv2[36] features of this image crop as the object's visual features. To calculate the visual similarity of two objects, we use the following visual similarity score[5]:

$$\text{Visual}(O_i, O_j) = 0.15 * \text{CLIP}(O_i, O_j) + 0.85 * \text{DINOv2}(O_i, O_j) \quad (1)$$

where  $\text{Visual}(O_i, O_j)$  denotes visual similarity of object  $O_i$  and  $O_j$ ,  $\text{CLIP}(\cdot, \cdot)$  and  $\text{DINOv2}(\cdot, \cdot)$  are the CLIP and DINOv2 similarities proposed in [5].

Besides the CLIP feature of the cropped object image (denoted as **OBJ Feat** in [Figure 2](#)), the CLIP feature of the frame containing the object is also stored as the context feature of the object (denoted as **CTX Feat** in [Figure 2](#)). The context feature will not be used for object re-ID, but it later enables retrieving objects by an open-vocabulary environment description ("blue wall", "kitchen", etc) during inference.

### C.2. Spatial Similarity Scores

Given two objects  $O_1$  and  $O_2$  and their 3D bounding boxes:  $[[x_1^{\min}, y_1^{\min}, z_1^{\min}], [x_1^{\max}, y_1^{\max}, z_1^{\max}]]$  and  $[[x_2^{\min}, y_2^{\min}, z_2^{\min}], [x_2^{\max}, y_2^{\max}, z_2^{\max}]]$ , their volumes and the volume of their intersection can be easily computed as:

$$\begin{aligned} V_1 &= (x_1^{\max} - x_1^{\min})(y_1^{\max} - y_1^{\min})(z_1^{\max} - z_1^{\min}), \\ V_2 &= (x_2^{\max} - x_2^{\min})(y_2^{\max} - y_2^{\min})(z_2^{\max} - z_2^{\min}), \\ x_{\text{inter}} &= \min(x_1^{\max}, x_2^{\max}) - \max(x_1^{\min}, x_2^{\min}), \\ y_{\text{inter}} &= \min(y_1^{\max}, y_2^{\max}) - \max(y_1^{\min}, y_2^{\min}), \\ z_{\text{inter}} &= \min(z_1^{\max}, z_2^{\max}) - \max(z_1^{\min}, z_2^{\min}), \\ V_{\text{inter}} &= \max(0, x_{\text{inter}}) * \max(0, y_{\text{inter}}) * \max(0, z_{\text{inter}}), \\ V_{\text{union}} &= V_1 + V_2 - V_{\text{inter}} \end{aligned}$$

where  $V_1$  and  $V_2$  are the volumes of  $O_1$  and  $O_2$ ,  $V_{\text{inter}}$  is the volume of their intersection and  $V_{\text{union}}$  is the volume of their union. we use three scores to evaluate the similarity of the two bounding boxes:

---

**Algorithm 2:** Static Object Re-Identification.

---

**Input:** detected object  $O_k$ , static object list  $\mathcal{S} = [S_1, S_2, \dots, S_m]$

**Output:** re-IDed object if  $O_k$  matches one of the static objects else  $O_k$

```
1 for  $S_i$  in  $[S_1, S_2, \dots, S_m]$  do
2   if  $\text{Spatial\_IoU}(O_k, S_i) > 0.2$  or  $(\text{Spatial\_MaxIoS}(O_k, S_i) > 0.2$  and  $O_k.\text{category} == S_i.\text{category})$  then
3     return True,  $S_i$ 
4 return False,  $O_k$ 
```

---

---

**Algorithm 3:** Dynamic Object Re-Identification.

---

**Input:** detected object  $O_k$ , dynamic object list  $\mathcal{D} = [D_1, D_2, \dots, D_n]$

**Output:** re-IDed object if  $O_k$  matches one of the dynamic objects else  $O_k$

```
1 for  $D_i$  in  $[D_1, D_2, \dots, D_n]$  do
2   if  $\text{Spatial\_Vol\_Sim}(O_k, D_i) > 0.7$  and  $\text{Visual}(O_k, D_i) > 0.45$  then
3     return True,  $D_i$ 
4 return False,  $O_k$ 
```

---

Intersection over Union (**IoU**):

$$\text{Spatial\_IoU}(O_i, O_j) = \frac{V_{\text{inter}}}{V_{\text{union}}}. \quad (2)$$

Maximum Ratio of Intersection over Subsets (**MaxIoS**):

$$\text{Spatial\_MaxIoS}(O_i, O_j) = \max\left(\frac{V_{\text{inter}}}{V_1}, \frac{V_{\text{inter}}}{V_2}\right). \quad (3)$$

Bounding Box Volume Similarity (**Vol\_Sim**)

$$\text{Spatial\_Vol\_Sim}(O_i, O_j) = \frac{\min(V_1, V_2)}{\max(V_1, V_2)}. \quad (4)$$

These three scores evaluate object spatial proximity from three different perspectives:

- **Spatial\_IoU**: When two bounding boxes have similar volumes and have large intersection volume,  $\text{Spatial\_IoU}$  will approach its maximum value 1. It is a strong indicator (when  $\text{Spatial\_IoU} > 0.2$ ) of two bounding boxes referring to the same object.
- **Spatial\_MaxIoS**: When two bounding boxes demonstrate a strong containment relationship,  $\text{Spatial\_MaxIoS}$  will get closer to its maximum value 1. For example, given that  $O_1$  and  $O_2$  are both detected as 'table',  $O_2$  is  $\frac{1}{10}$  the volume of  $O_1$  and its bounding box is inside  $O_1$ ,  $\text{Spatial\_MaxIoS}$  will reach 1, while their  $\text{Spatial\_IoU}$  is only 0.1. It is used together with object categories to re-identify partially observed objects due to occlusion. In the above example,  $O_2$  is possibly a partial observation of  $O_1$  given that they have overlapping bounding boxes and the same object category.
- **Spatial\_Vol\_Sim**: when two bounding boxes have similar volume,  $\text{Spatial\_Vol_Sim}$  will have larger value. It is used along with visual similarity scores to match dynamic objects.

### C.3. Recognizing Dynamic Objects

With the knowledge of both object visual features and 3D bounding boxes, we can perform object re-identification based on both visual similarity and spatial similarity. For static objects, spatial similarity serves as a valuable metric for object re-ID. However, for dynamic objects, object re-ID should focus more on the visual similarity of the object pairs, since the object positions are dynamically changing. Therefore, before re-identifying the newly detected objects, we should first classify the existing objects in the object memory into static objects and dynamic objects.

The key idea of recognizing dynamic objects in the object memory is straightforward: if an object is not where it should be, then it must be moved by someone (becomes dynamic). We first retrieve the objects from the object memory whose 3D bounding boxes can be directly viewed on the current frame (achieved by world-to-camera transformation) with no occlusion (achieved by validating the depth values of the corresponding pixels). For each retrieved object, We then compare the visual features of "where it should be" on the current frame with its visual features in the object memory. If the visual similarity score is below a threshold (0.45 in our settings), then the object is not "where it should be" and should be marked as "dynamic". By this method, before performing object-reID on current detections, we split the objects in the object memory into two sets: static objects  $\mathcal{S}$  and dynamic objects  $\mathcal{D}$ .

### C.4. Object Re-ID for Static and Dynamic Objects

Algorithm 2 and Algorithm 3 are the object re-ID methods for static objects and dynamic objects correspondingly. Each algorithm receives a newly detected object  $O_k$  with visual features and its 3D bounding box, and a list of candidate

---

**Algorithm 4:** Object Memory Update.

---

**Input:** current observations  $\text{Obs}^t = \{\text{RGB}^t, \text{Depth}^t, \text{Pose}^t\}$ , previous object memory  $\mathcal{M}_O^{t-1}$   
**Output:** current object memory  $\mathcal{M}_O^t$

```
1 2DBoxes, categories = 2D_Detector( $\text{RGB}^t$ )
2  $\mathcal{S}, \mathcal{D} = \text{ObjectSplit}(\mathcal{M}_O^{t-1}, \text{Obs}^t)$  //See Appendix C.3
3 for  $i$  in range(len(2DBoxes)) do
4   category = categories[ $i$ ]
5   2DBox = 2DBoxes[ $i$ ]
6   3DBox = 2D_3D_Lifting(2DBox,  $\text{Obs}^t$ ) //See Appendix B
7   FeatCLIP = CLIP_Model( $\text{RGB}^t[2\text{DBox}]$ )
8   FeatDINOv2 = DINOv2_Model( $\text{RGB}^t[2\text{DBox}]$ )
9    $O_{\text{tmp}} = \text{Object3D}(\text{category}, 3\text{DBox}, \text{Feat}_{\text{CLIP}}, \text{Feat}_{\text{DINOv2}})$ 
10   $\text{sgn}, O_{\text{ID}} = \text{Static\_Object\_ReID}(O_{\text{tmp}}, \mathcal{S})$  //first try to re-identify  $O_{\text{tmp}}$  from static objects (Algorithm 2)
11  if  $\text{sgn} == \text{True}$  then
12     $O_{\text{ID}} = \text{Static\_Object\_Merge}(O_{\text{tmp}}, O_{\text{ID}})$ 
13  else
14     $\text{sgn}, O_{\text{ID}} = \text{Dynamic\_Object\_ReID}(O_{\text{tmp}}, \mathcal{D})$  //try to re-identify  $O_{\text{tmp}}$  from dynamic objects (Algorithm 3)
15    if  $\text{sgn} == \text{True}$  then
16       $O_{\text{ID}} = \text{Dynamic\_Object\_Merge}(O_{\text{tmp}}, O_{\text{ID}})$ 
17      move  $O_{\text{ID}}$  from  $\mathcal{D}$  to  $\mathcal{S}$ 
18    else
19      add  $O_{\text{tmp}}$  to  $\mathcal{S}$  //  $O_{\text{tmp}}$  is a brand new object
20  $\mathcal{M}_O^t = \mathcal{S} \cup \mathcal{D}$ 
21  $\mathcal{M}_O^t = \text{Related\_Object\_Update}(\mathcal{M}_O^t)$ 
22  $\mathcal{M}_O^t = \text{VLM\_Update}(\mathcal{M}_O^t, \text{RGB}^t)$ 
23 return  $\mathcal{M}_O^t$ 
```

---

objects (static object list or dynamic object list). They both return whether the object  $O_k$  can be successfully identified and the object ID of the matched object in the candidate list. If  $O_k$  is re-identified, it is merged into the matched object by performing a moving average on the fields of the 3D bounding box and visual features. Specifically, to merge the two objects matched by static object re-ID, the window size of the moving average is set to 10, leading to a mild change in object visual features and spatial occupation; for dynamic object merging, we set the window size to 2, allowing rapid change of visual features and bounding boxes due to object movement.

Algorithm 4 presents an overview of object memory update, including 3D object detection and re-ID. The main idea is to first divide the objects in  $\mathcal{M}_O^{t-1}$  into static ones  $\mathcal{S}$  and dynamic ones  $\mathcal{D}$ , and try to match the newly detected objects to these two kinds of objects through Algorithm 2 and Algorithm 3 respectively. If successfully matched, the newly detected objects will be merged with the matched objects in the object memory using the moving average as mentioned, otherwise, it will be viewed as a brand new object and added to the object memory. Finally, VLM-based

Memory update will be performed on  $\mathcal{M}_O^t$ , which will be discussed in Appendix D.

## D. VLM-based Memory Update

When *Embodied VideoAgent* serves as an observer of an egocentric video, *Embodied VideoAgent* needs to predict the actions of the camera wearer in the video and associate the object IDs in the object memory with the subjects of the actions. We use LaViLa[64] to annotate the action of the camera wearer every two seconds. For each action annotation, we first prompt an LLM (GPT-4o) to extract the objects in the annotation (e.g. "bottle" and "fridge" given the annotation "#C C picks the bottle from the fridge") and select candidate objects detected at that time according to their categories for matching. We then perform VLM-based object association illustrated in Figure 4, and save the actions to Action Buffer. Finally, we query the state change of the matched objects and update the "STATE" field of the object entries. In this paper, objects have one of the following states: "open", "close", "in hand" and "normal".

When *Embodied VideoAgent* is equipped with embodied actions, the procedure of VLM-based object association is

Table 6. Results of *Embodied VideoAgent* under noisy poses.

OpenEQA Subset			
Method	ScanNet	HM3D	ALL
Video-LLaVA	32.9	27.8	30.6
LLaMA-VID	31.2	28.0	29.4
VideoAgent	38.9	41.4	40.0
<b>E-VideoAgent(GT poses)</b>	<b>39.7</b>	<b>43.0</b>	<b>41.2</b>
<b>E-VideoAgent(noisy poses)</b>	38.2	42.2	40.0





omitted since *Embodied VideoAgent* serves as an active planner with the knowledge of the object IDs of its target objects or receptacles. In this case, VLM serves as an action validator that judges whether an action is successfully performed and updates the "STATE" field of the target objects.

## E. Results under Noisy Camera Poses


We conduct the ablation study of the influence of the noisy camera poses. On OpenEQA benchmark, We provide *Embodied VideoAgent* (InternVL-2) with 1) the accurate camera poses provided in habitat simulator, denoted as **E-VideoAgent(GT poses)**; 2) the estimated camera poses and depths via DUST3R[50], denoted as **E-VideoAgent(noisy poses)**. Results in Table 6 show that *Embodied VideoAgent* can also handle perception tasks well based on the noisy poses, suffering little performance drops when using the estimated camera poses and depths. This suggests further applications of *Embodied VideoAgent* on RGB videos only, with the camera poses and depths being estimated by cutting-edge scene reconstruction methods.

## F. Embodied Perception

For embodied perception, we equip *Embodied VideoAgent* with the following tools:

-  **query\_db**: Given a query, this tool will return the candidate object entries from Persistent Object Memory. It is a combination of code-based retrieval (writing a piece of MySQL code to query the database) and similarity-based retrieval. For similarity-based retrieval,  **query\_db** supports `retrieve_objects_by_appearance` (based on text-image similarities between the query text and the **OBJ Feats**) and `retrieve_objects_by_environment` (based on text-image similarities between the query text and the **CTX Feats**).
-  **temporal\_loc**: Return the top-5 frame IDs that satisfy the description (e.g. when I walk in the front door). It is achieved by the text-image similarity between the input description and the frame features stored in the temporal memory  $\mathcal{M}_T$ .
-  **spatial\_loc**: Return the top-3 3D positions that satisfy the description (e.g. bedroom). It is achieved by calculating the center positions of the top-3 object spatial

clusters where objects have strong **CTX feat** similarities to the input text description. This is only used for embodied navigation.

-  **vqa**: Given an image (can be a video frame, a cropped object image, or a frame plotted with a 3D bounding box referring to a specific object), this tool will describe the image and then answer the question.

We use the following prompt for perception tasks, with **{tools}** in the prompt being the above tools. We choose GPT-4o as the LLM agent and InternVL2 as the VLM for visual question answering.

You are tasked with answering a question about a scene. There is a SQL database that contains the following tables:

```
TABLE Objects(
  object_id INT,
  category VARCHAR(255),
  volume FLOAT,
  PRIMARY KEY (object_id)
)
TABLE Objects_Frames(
  object_id INT,
  frame_id INT
)
```

The 'Objects' table catalogues the objects detected in the scene with their categories and volumes. The categories are from **{categories\_list}**. The 'Objects\_Frames' table records the appearing objects in the frames.

You have access to the following tools:

**{tools}**

ATTENTION:

1. Use double quotes for all string arguments when using tools, avoiding any single quotes. If the tool has two arguments, format the output within brackets, like this: ("What is the color of the object in the image?", 16).
2. Prioritize using `retrieve_objects_by_appearance`, `retrieve_objects_by_environment`, and `frame_localization` for object and frame localization, rather than `query_database`.
3. Once the relevant frames are located, use `frame_VQA` to ask questions about these frames.
4. For "where" questions, interpret them as spatial location queries, not as requests to identify specific frames. Use `frame_VQA` to provide these answers.
5. If the information is insufficient for a precise response, generate a response based on the available



data.

Use the following format:

Question: the input question you must answer

Thought: you should always think about what to do

Action: the action to take, should be one of  
[**{tool\_names}**]

Action Input: the input to the action

Observation: the result of the action... (this  
Thought/Action/Action Input/Observation can re-  
peat N times)

Thought: I now know the final answer

Final Answer: the answer to the original input  
question

Begin!

Question: **{input}**

Thought: **{agent\_scratchpad}**

### F.1. Example 1

- **Question:** "What color are the blinds?"
- **Ground-Truth Answer:** "White"
- **Predicted Answer:** "The blinds are white."

In the following Chain Of Thought (CoT), *Embodied VideoAgent* first successfully locates the frames containing the target object "blinds" by "frame\_localization" as shown in [Figure 9](#). Then, *Embodied VideoAgent* calls the VQA module (InternVL2) to first describe the located frame and then answer the question based on this frame, and finally gets the correct answer: "The blinds are white".

Entering new AgentExecutor chain...

**Thought:** To determine the color of the blinds, I need to locate frames that might contain blinds and then analyze those frames.

**Action:** frame\_localization

**Action Input:** "blinds"

**Result:** The most relevant frame indices are [190, 200, 210, 180, 240]

**Thought:** I have identified the frames that likely contain blinds. Now, I will use frame\_VQA to determine the color of the blinds in these frames.

**Action:** frame\_VQA

**Action Input:** ("What is the color of the blinds?", 190)

**Result:** Description: The image shows an indoor scene featuring a window with partially open horizontal blinds. Below the window is a heater or radiator, and part of a stove is visible on the left side. Outside the window, some greenery can be seen.

Answer to the question: The blinds are white.

**Thought:** I now know the final answer.

**Final Answer:** The blinds are white.

Finished chain.



Figure 9. Frame No.190 that contains blinds are successfully located by *Embodied VideoAgent* in [Section F.1](#).

## F.2. Example 2

- **Question:** "How many cups does the Mr coffee make?"
- **Ground-Truth Answer:** "12"
- **Predicted Answer:** "The Mr. Coffee machine makes 12 cups"

To answer this question, *Embodied VideoAgent* uses temporal localization to locate the frame containing the coffee machine as illustrated in Figure 10. Then, it queries InternVL2 for a detailed description of this coffee machine. InternVL2 successfully identifies the words on the box and finally answers that Mr. Coffee is capable of making 12 cups. The CoT for the inference is provided as follows.

Entering new AgentExecutor chain...

**Thought:** To answer the question about how many cups the Mr. Coffee makes, I need to locate frames that might show the Mr. Coffee machine and then analyze those frames to determine its capacity.

**Action:** frame\_localization

**Action Input:** "Mr. Coffee machine"

**Result:** The most relevant frame indices are [270, 280, 260, 250, 290]

**Thought:** I have identified the frames that likely contain the Mr. Coffee machine. I will now use the frame\_VQA tool to analyze these frames and determine the capacity of the Mr. Coffee machine.

**Action:** frame\_VQA

**Action Input:** ("How many cups does the Mr. Coffee make?", 270)

**Result:** Description: The image shows a kitchen section with multiple shelves.

- On the topmost shelf, there is a white box labeled "Hamilton Beach" with visible text indicating that it is related to a kitchen appliance, and a photo of the appliance is shown on the box. This section includes other miscellaneous items.
- The middle shelf contains a big brown box labeled "Mr. Coffee." The box features a picture of a coffee maker.
- In the bottom shelf, there are two colorful sponges – one red and one green – along with various other items that can't be clearly identified.

This portion of the room appears to be organized and neatly arranged, with items stored in an accessible manner.

Answer to the question: Based on the label visible in the image, the Mr. Coffee maker is capable of making 12 cups.

**Thought:** I have determined the capacity of the Mr. Coffee machine from the frame description.

**Final Answer:** The Mr. Coffee machine makes 12 cups.

Finished chain.



Figure 10. In Section F.2, The frame depicting the coffee machine named "Mr. Coffee" is successfully retrieved by frame\_localization.

### F.3. Example 3

- **Question:** "Is there anything I can pour my soda into?"
- **Ground-Truth Answer:** "Yes, there is a cup to the left of the desk."
- **Predicted Answer:** "Yes, there is a cup and a bottle in the scene that you can pour your soda into."

This question is about functional reasoning. *Embodied VideoAgent* first uses `retrieve_objects_by_appearance` to retrieve the objects that are most visually relevant to the phrase "cup" from the object memory. The captions of these objects are also returned by this tool via VLM captioning on the frames containing these objects, with 3D bounding boxes being the visual prompts, as shown in Figure 11. Then based on the retrieved objects, *Embodied VideoAgent* performs `object_VQA`, which answers questions based on the images in Figure 11. By examining object ID 2 and object ID 454, *Embodied VideoAgent* becomes certain that there exists a container that the user can pour soda into. The CoT is provided as follows.

Entering new AgentExecutor chain...

**Thought:** To determine if there is anything in the scene that can be used to pour soda into, I should look for objects that are typically used as containers for liquids. Common objects for this purpose include cups, bottles, and containers. I will start by retrieving objects that match these categories.

**Action:** `retrieve_objects_by_appearance`

**Action Input:** "cup"

**Result:** The objects that satisfy 'cup' are 2: 'The object in the bounding box is a green cup or mug located on the left side of the table.', 454: 'The object in the bounding box is a bottle, specifically labeled "454 bottle". There is a green box around the highlighted area, and the object appears to have a label with green and black colors.', 1748: 'The object in the bounding box is a rectangular, black desk organizer positioned next to a black desk lamp on the desk. The organizer has compartments filled with pens, pencils, markers, and other office supplies.', 9: 'The object within the bounding box in the image is labeled as a "water bottle."'

**Thought:** I have found some objects that could potentially be used to pour soda into, such as a cup and a bottle. I will now verify if these objects are suitable for pouring soda by asking specific questions about them.

**Action:** `object_VQA`

**Action Input:** ("Is this object suitable for pouring soda into?", 2)

**Result:** Description: The object in the bounding box is a cup.

Answer to the question: Although the object is not explicitly a cup, it resembles a cup and is likely suitable for pouring soda into.

**Thought:** The object with ID 2 resembles a cup and is likely suitable for pouring soda into. I will now check the other objects to see if they are also suitable.

**Action:** `object_VQA`

**Action Input:** ("Is this object suitable for pouring soda into?", 454)

**Result:** Description: The object in the bounding box is a beverage can.

Answer to the question: Yes, the object in question is suitable for pouring soda into, as it is shown as a bottle.

**Thought:** The object with ID 454 is a bottle and is suitable for pouring soda into. I have confirmed that both the cup (ID 2) and the bottle (ID 454) are suitable for pouring soda into.

**Final Answer:** Yes, there is a cup and a bottle in the scene that you can pour your soda into.

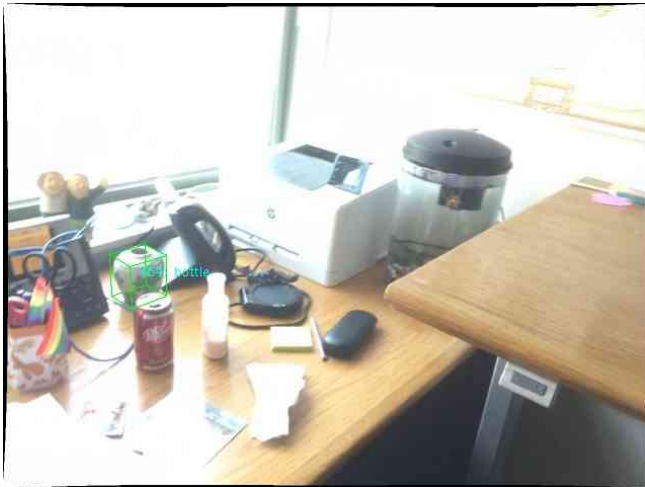
Finished chain.



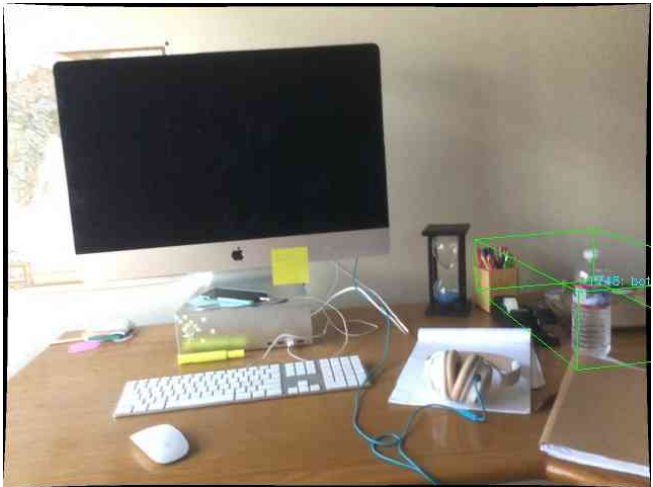
(a) Object ID 2, category "cup".



(b) Object ID 9, category "bottle".



(c) Object ID 454, category "bottle".



(d) Object ID 1748, category "bottle".

Figure 11. The objects that are most relevant to "cup" are successfully retrieved by `retrieve_objects_by_appearance` in [Section F.3](#).



- **Question:** "Could I put something on the chair by the front door?"
- **Ground-Truth Answer:** "Yes"
- **Predicted Answer:** "The chair by the front door has a small volume, so it might not be suitable for placing items on it."

In this example, *Embodied VideoAgent* successfully locates the chair (ID 107) by the front door as shown in Figure 12. Then, *Embodied VideoAgent* decides to query the volume of its 3D bounding box to answer the question "Could I put something on the chair by the front door". However, this chair is in the 2D texture of the scene, which fails 2D-3D lifting. The volume of its 3D bounding box (nearly being a plane) has a very small value, misleading *Embodied VideoAgent* into the wrong answer. In real-world applications, obtaining object depths through mirrors (windows, etc.) could have much noise, which should be taken into account for accurate perception.

Entering new AgentExecutor chain...

**Thought:** To answer this question, I need to determine if there is a chair by the front door and if it has enough space or surface area to place something on it. First, I will identify the objects categorized as "chair" and "door" in the scene. Then, I will check if any of these chairs are located near the front door.

**Action:** retrieve\_objects\_by\_environment

**Action Input:** "front door"

**Result:** The objects in the environment 'front door' are 493: 'door', 107: 'chair', 483: 'cubby shelf', 478: 'window', 482: 'photo frame'

**Thought:** I have identified that there is a chair (object ID 107) near the front door. Now, I need to determine if this chair has enough space or surface area to place something on it. I will check the volume of the chair to get an idea of its size.

**Action:** query\_database

**Action Input:** "SELECT volume FROM Objects WHERE object\_id = 107"

**Result:** [(0.0006219363573111673,)]

**Thought:** The volume of the chair near the front door is quite small, suggesting it might not have a large surface area. However, without specific information about the chair's design or surface, it's difficult to definitively say whether something can be placed on it. Given the small volume, it is likely that the chair is not designed to hold items on its surface.

**Final Answer:** The chair by the front door has a small volume, so it might not be suitable for placing items on it. Finished chain.

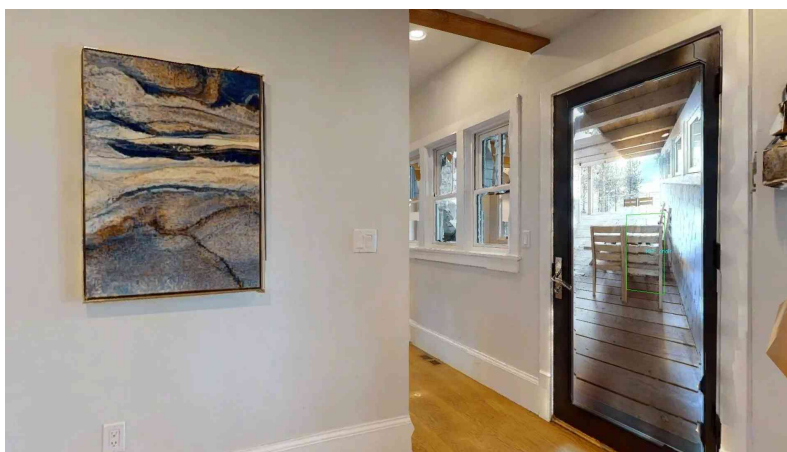









Figure 12. The 2D bounding box of the detected chair is not successfully 3D-lifted since it is a 2D texture in the scene.

## G. Two-Agent Framework

In AI-Habitat simulator [43], we equip *Embodied VideoAgent* with the following embodied actions:

-  **CHAT**: Communicate with the user.
-  **SEARCH**: Search for the target object by navigating in the apartment. We use Frontier-Based Exploration (FBE) as the navigation strategy.
-  **GOTO**: Go to the target receptacle or object and look at it. We use A-star Algorithm for GOTO action.
-  **PICK**: Pick an object in view. It is simplified as making the object disappear and storing the object ID as the inventory object.
-  **PLACE**: Place the inventory object in/on a receptacle in view. The Place Action will first examine the precondition for the placement by checking the bounding boxes of the inventory object and the receptacle and the relation "in" or "on".
-  **OPEN**: Open an articulated receptacle in view. Simplified as applying force to the joints of the articulated receptacles.
-  **CLOSE**: Close an articulated receptacle in view. Simplified as applying reversed force to the joints of the articulated receptacles.

We adopt the scenes from Habitat HSSD scene dataset[21] for embodied tasks. We choose 118 scenes from HSSD, replacing some rigid receptacles in the original scenes with articulated assets (fridge, microwave, etc) to enable OPEN and CLOSE actions.

For each scene, 15 different object layouts are created. In each layout, objects from various categories are placed on/into the receptacles in the scene using a unique object initialization algorithm, which initializes the positions of the objects according to their functionality (e.g. eggs and tomatoes are prioritized to be placed in the fridge rather than on the bed).

The embodied interaction episodes are generated based on two LLM agents: the User Agent (task designer) and the Assistant Agent (*Embodied VideoAgent*). For *Embodied VideoAgent*, it is equipped with both the embodied actions and the perception tools. A VLM (gpt-4o) is prompted to judge whether the tasks are successfully finished and score the quality of the episodes. The prompts for the user agent, the assistant agent and the scoring VLM are provided respectively.

You are a task designer interacting with a robot in a room. The room contains the following objects: **{object\_list}** and the following receptacles: **{recep\_list}**. Your goal is to engage in a casual conversation with the robot and assign it an open-ended task based on your needs.

Guidelines:

1. The task should involve no more than 2 objects from the room.
  2. The robot should complete the task using basic actions like GOTO, OPEN, CLOSE, PICK, and PLACE.
  3. If the robot asks for the location of an object, prompt it to search rather than giving explicit details.
  4. Use general object categories instead of specific IDs (e.g., say "a dish sponge" instead of "dish sponge 1").
  5. Adjust the task if the robot encounters difficulties. Once the task is completed, express satisfaction and thank the robot.
- Start by initiating a casual conversation and assigning a simple task!

You are acting as a robot in an apartment. The available receptacles are: **{receptacles}**

Your goal is to complete the task assigned by the user, with the following conditions:

Tools and Constraints:

You have one inventory slot, so you can carry only one object at a time.

You can use the following tools:

**{tools}**

ATTENTION:

1. Use the CHAT tool frequently to communicate in a casual manner, keeping the user informed of your progress.
2. For every action involving an object or receptacle, first GOTO the target and then perform actions like PICK, PLACE, OPEN, or CLOSE. Example: GOTO('glass'), then PICK('glass'); GOTO('fridge'), then OPEN('fridge').
3. Ensure your inventory is empty before picking up a new object.
4. The SEARCH tool can find objects by navigating the room, but it cannot check inside articulated receptacles (like fridges or microwaves). Use GOTO, OPEN, and CLOSE to check inside these receptacles.
5. Before completing the task, use CHAT to confirm the user's satisfaction.

Use the following format:

Task: the initial task assigned by the user

Thought: you should always think about what to do

Action: the action to take, should be one of `[{tool_names}]`

Action Input: the input to the action

Observation: the result of the action... (this Thought/Action/Action Input/Observation can repeat N times)

Final Answer: the chat message sent to user when the user is satisfied

Begin!

Task: `{input}`

Thought: `{agent_scratchpad}`

You are a scoring machine. Now there are a number of tasks that need to be scored by robots. For each task, I will provide the following information: task description, tools used to complete the task, output when using each tool, feedback after using each tool and the picture you see when the task is completed. You need to score the robot's task completion based on this information. The scoring range is 0-5 points(interger).

Next I'll give you an example of how you should score.

Example: Task description: "Hello there! How are you doing today? The room is looking quite interesting with all these objects. I was just wondering, could you help me with a little something?"

I've been curious about the toy collection in here. Could you find one of the toy fruits and the toy fire truck, and then place them together on one of the tables? I'm sure they might look intriguing next to each other.

Take your time to explore and let me know once you've placed them. Thank you!"

Tool list: ["CHAT", "SEARCH", "SEARCH", "GOTO", "PICK", "GOTO", "PLACE", "GOTO", "PICK", "GOTO", "PLACE", "CHAT"]

Answer list: [

"Great! Take your time, and let me know if you need any assistance along the way. I'm eagerly waiting to see how they'll look together on the table!",

"Target object found! Their names are ["toy\_fruits\_1"].",

"Target object found! Their names are ["toy\_fire\_truck\_1"].",

"Go to toy\_fruits\_1 successfully! In the current view, you can see ["toy\_fruits\_1", "toy\_fire\_truck\_1", "box\_1", "plant\_container\_1"].",

"Pick up toy\_fruits\_1 successfully!",

"Go to table\_1 successfully! In the current view,

you can see ["backpack\_1", "battery\_charger\_1"].",  
"Place toy\_fruits\_1 on table\_1 successfully!",  
"Go to toy\_fire\_truck\_1 successfully! In the current view, you can see ["toy\_fire\_truck\_1", "box\_1", "plant\_container\_1"].",  
"Pick up toy\_fire\_truck\_1 successfully!",  
"Go to table\_1 successfully! In the current view, you can see ['backpack\_1', 'toy\_fruits\_1', 'battery\_charger\_1'].",  
"Place toy\_fire\_truck\_1 on table\_1 successfully!",  
"Fantastic work! Thank you so much for placing them together; they do make quite an interesting pair. I appreciate your help, and that's all for now. If there's anything else later, I'll be sure to let you know. Thanks again!"

]

Final view: "A picture of toy fruit and toy fire truck on the table 1"

Output:

Finished: [True]

Score: [5]

Reason: The robot first looked for toy fruit and toy fire truck in the room. Then it successfully placed them on a table, completing the task. The robot completed the task perfectly, so I give it a score of 5.

Now you need to score the task that will be shown to you according to the above format. You need to give me the answer according to the output format in the sample. You need to carefully analyze the information I provide you and the picture after the task is completed, and then give a comprehensive score to determine whether the task is completed.

Task description: TASK\_DESCRIPTION

Tool list: TOOL\_LIST

Answer list: ANSWER\_LIST

Final view: I'll show you after the question.

Please give me output with the format the same with shown in example above.

## G.1. Example 1

Figure 13 shows an interaction example using the two-agent pipeline. Given the partial scene knowledge, the user agent asks the assistant agent (*Embodied VideoAgent*) to find two objects: a glass and a hard drive, to compare their surface reflection. *Embodied VideoAgent* then performs the SEARCH action, which will start Frontier-Based Exploration (FBE) until the target object is found in the view. During exploration, a glass is found on Table\_2, and *Embodied VideoAgent* reports this progress to the user agent. The user agent hints that the next object, the hard drive, is possibly located in an office. *Embodied VideoAgent* then uses QUERY\_DB tool and successfully retrieves the hard drive discovered by FBE during searching for the glass. *Embodied VideoAgent* then goes to the hard drive, picks it up, and places it on Tables\_2 where the glass is located for comparison, and finally accomplishes the task assigned by the user agent.

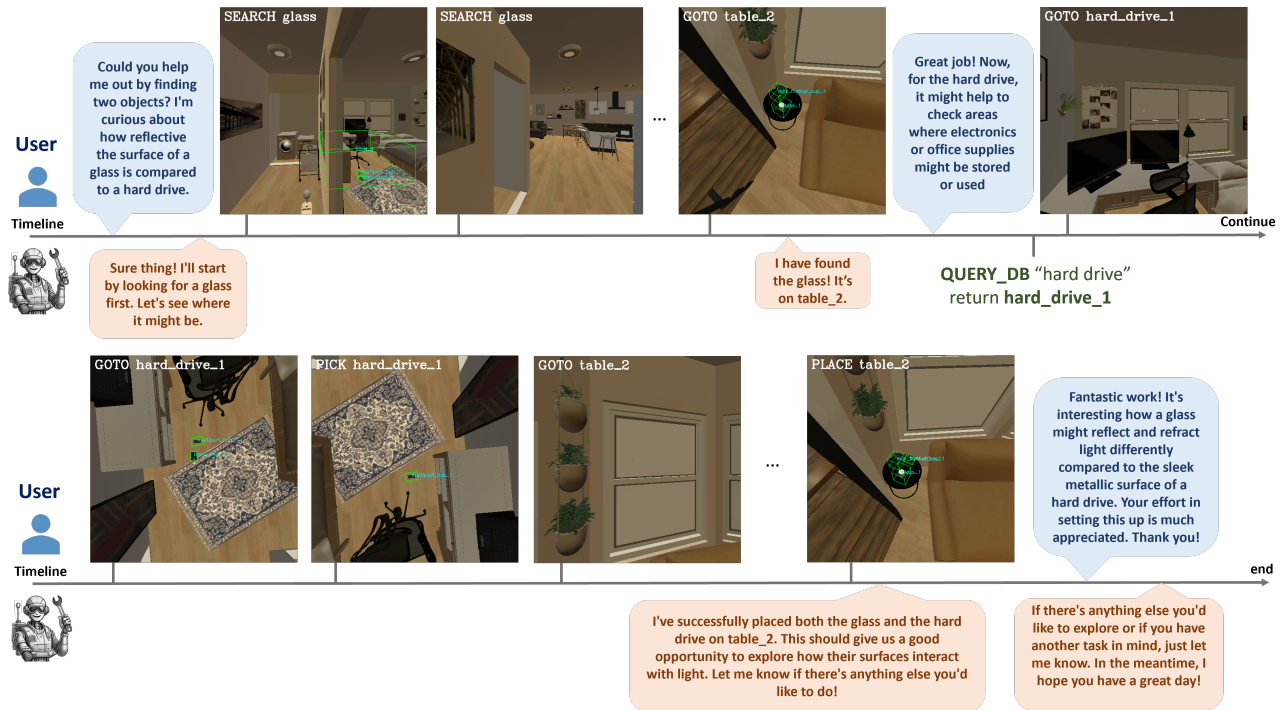


Figure 13. An example of interaction data, which is detailed in Section G.1. *Embodied VideoAgent* finds the two objects (a glass and a hard drive) requested by the user agent and places them on the same table for comparison.

## G.2. Example 2

In Figure 14, the user agent requests *Embodied VideoAgent* to find a candy bar. After navigating through the entire apartment and checking the closed receptacles such as the fridge, *Embodied VideoAgent* still cannot find the candy bar, and report this issue to the user agent. The user agent then adjusts the task, asking *Embodied VideoAgent* to place a lamp on one of the tables. *Embodied VideoAgent* successfully retrieves the lamp stored in the object memory, which is discovered during searching for the candy. *Embodied VideoAgent* finally completes the adjusted tasks by picking up the lamp, navigating to a table, and placing the lamp on the table. This case shows that the user agent can flexibly change the task when it is too hard to be accomplished by the robot.

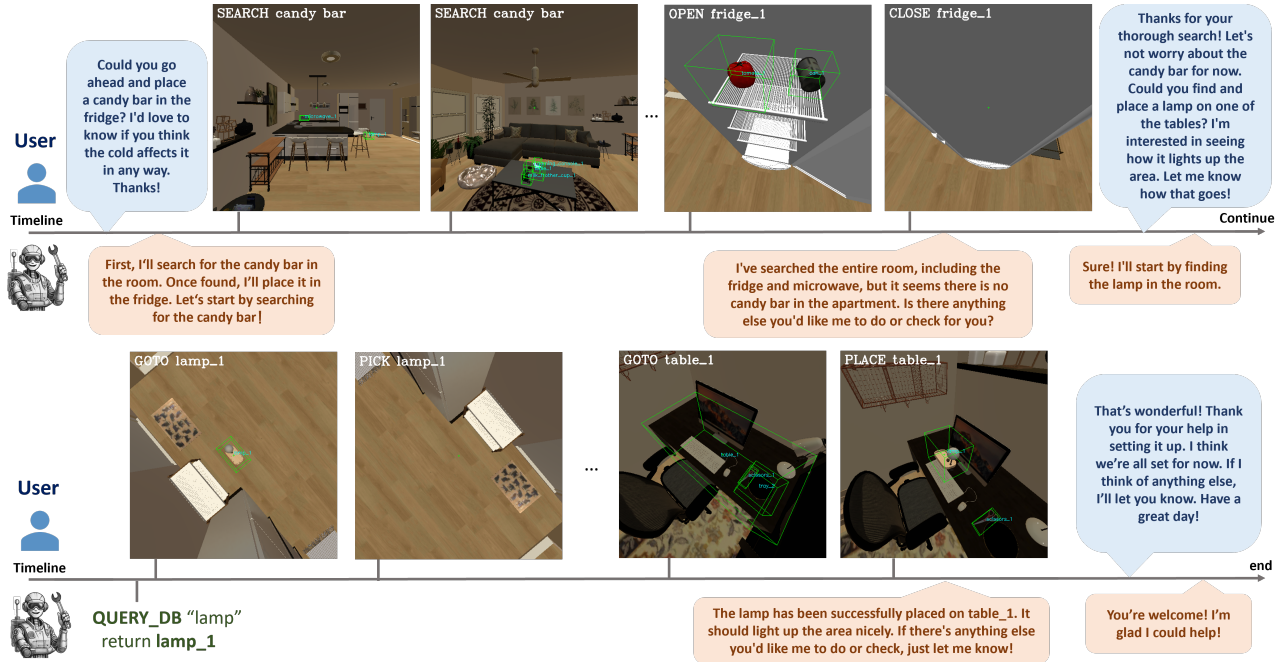


Figure 14. When *Embodied VideoAgent* fails to accomplish the initial task assigned by the user (in this case, *Embodied VideoAgent* cannot find the candy bar after searching the entire apartment), the user agent will interactively adjust the task. Finally, *Embodied VideoAgent* completes the adjusted task by placing the lamp on a table.



### G.3. Example 3

Figure 15 is an example that demonstrates the thought of *Embodied VideoAgent* during the episode. The user agent first initializes the task as setting up a cozy corner by placing a cushion on a table. *Embodied VideoAgent* finds the cushion after a simple search and plans to place it on one of the tables retrieved from the memory. However, the embodied action "PLACE" returns an exception due to the limited area of the first three tables. The thought of *Embodied VideoAgent* shows that *Embodied VideoAgent* can dynamically change its plan according to the situation. Finally, *Embodied VideoAgent* finished the task by successfully placing the cushion on the fourth table.

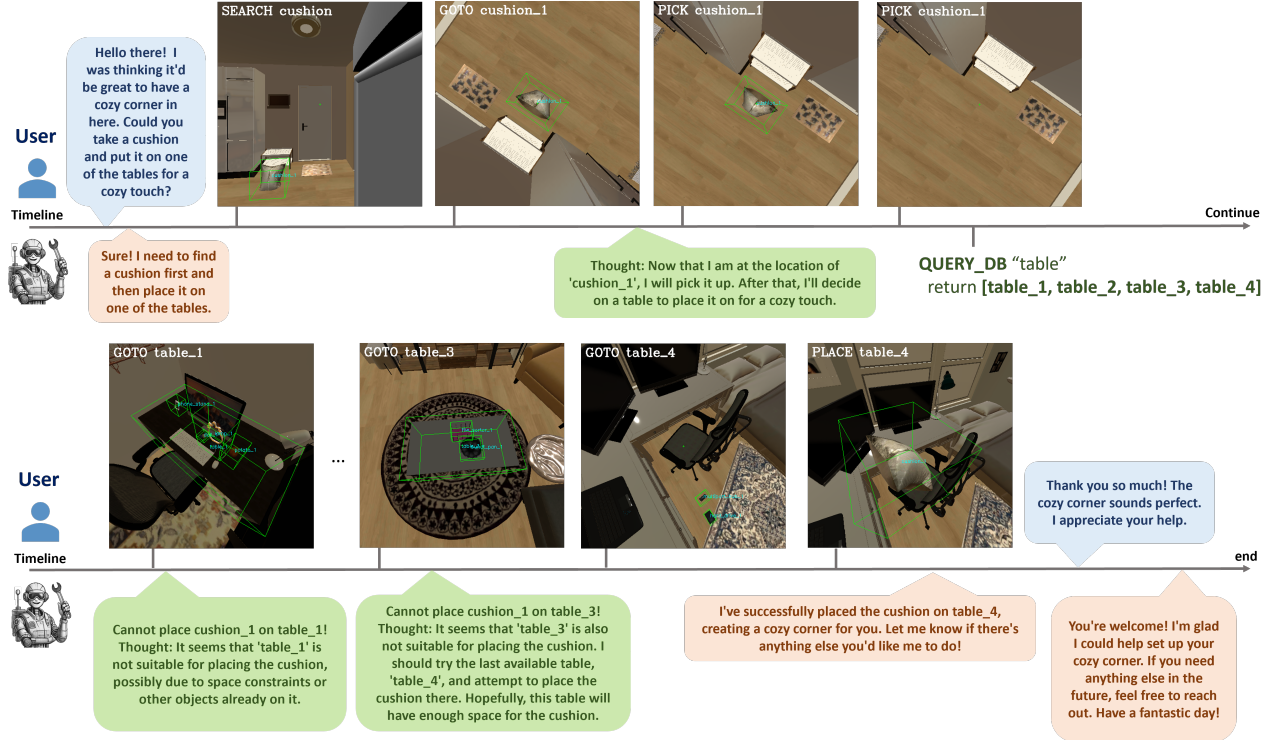


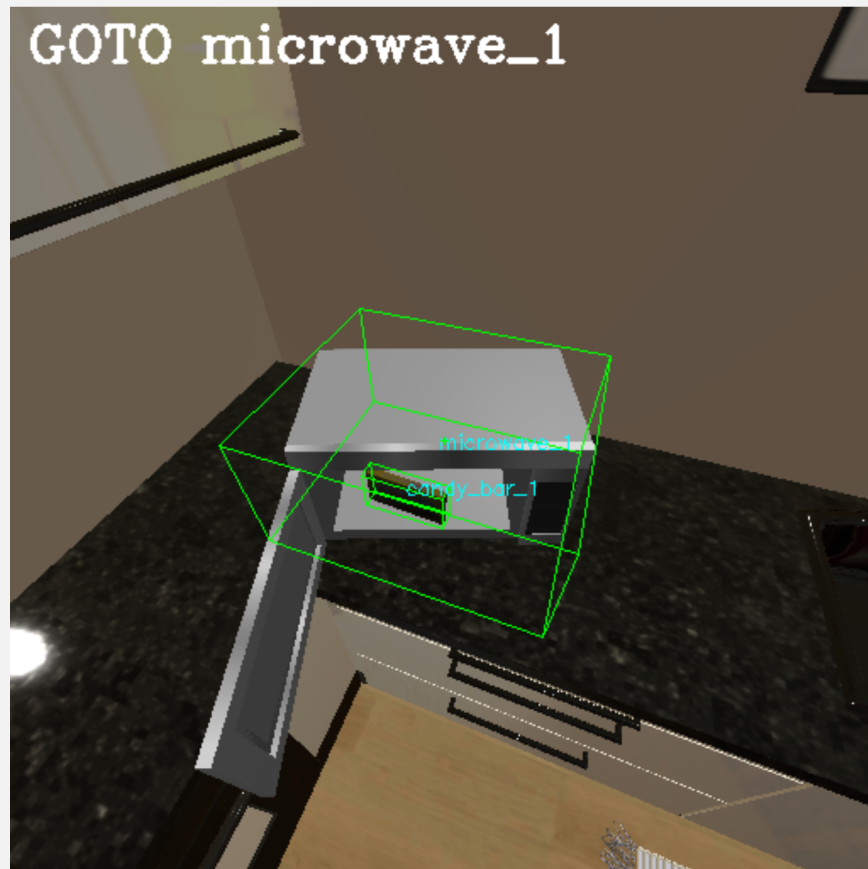
Figure 15. In this case, *Embodied VideoAgent* is asked to set up a cozy corner using the cushion. After successfully finding the cushion, *Embodied VideoAgent* retrieves candidate tables from the memory for placement. *Embodied VideoAgent* fails to place the cushion on the first three tables due to spatial constraints. Finally, *Embodied VideoAgent* successfully places the cushion on the fourth table, demonstrating the planning ability of *Embodied VideoAgent* when experiencing failed embodied actions.

## **H. Cases**

In this section, we will provide some evaluation cases of the generated episodes.

## History:

"task": "Hello there! How are you today? I was just admiring the variety of objects we have in this room. Out of curiosity, could you help me with something? I'd love to know the current state of the toy animal. Could you please find it and let me know if it's in good condition?", "tool\_cnt": 13, "action\_cnt": 3474, "action\_list": [ "CHAT", "SEARCH", "GOTO", "OPEN", "CLOSE", "GOTO", "OPEN", "GOTO", "GOTO", "GOTO", "GOTO", "GOTO", "CHAT", "GOTO", "FRAME\_VQA", "CHAT", "CHAT" ], "time": 389.35402250289917, "ans\_list": [ "Great! Take your time to search, and let me know what you find when you reach the toy animal. I'm curious to hear about its condition!", "All places except articulated receptacles have been searched but no toy animal was found!", "Go to fridge\_1 successfully! target is currently closed. In the current view, you can see [].", "Open fridge\_1 successfully! In the current view, you can see ['glass\_2'].", "Close fridge\_1 successfully!", "Go to microwave\_1 successfully! target is currently closed. In the current view, you can see [].", "Open microwave\_1 successfully! In the current view, you can see ['candy\_bar\_1'].", "Go to table\_1 successfully! In the current view, you can see ['doll\_1', 'potato\_1', 'phone\_stand\_1', 'lamp\_1'].", "Go to table\_2 successfully! In the current view, you can see ['glass\_1', 'milk\_frother\_cup\_1'].", "Go to table\_3 successfully! In the current view, you can see ['file\_sorter\_1', 'bundt\_pan\_1'].", "Go to table\_4 successfully! In the current view, you can see ['hard\_drive\_1', 'multiport\_hub\_1'].", "Go to bathtub\_1 successfully! In the current view, you can see [].", "Oh, it seems like the toy animal is playing hide and seek with us! No worries at all. Let's try something different then. Could you check the condition of the candy bar instead? Let me know if it looks fine or if it might have melted. Thanks!", "Go to microwave\_1 successfully! target is currently open. In the current view, you can see ['candy\_bar\_1'].", "The candy bar appears intact and full in the image, not showing any signs of having been partially consumed or broken off.", "That's great to hear! Thanks for checking on that for me. Everything seems to be in order then. I appreciate your help with this little investigation. If there's anything else you need from me, just let me know. Have a wonderful day!", "Thank you! I'll definitely reach out if I need anything. Take care and enjoy the rest of your day!" ]



**Scores and Reasons:**

Finished: [True] Score: [4] Reason: The robot was tasked with determining the condition of a toy animal, but it was unable to locate it, as indicated by the response after checking various locations. The task pivoted to inspecting the condition of a candy bar in the microwave. The robot successfully checked the candy bar's condition, confirming it was intact and relaying this information. The task was adapted successfully, but the original request to find and assess the toy animal was not fulfilled, leading to a score of 4 instead of 5.

## History:

"task": "Hello there! How are you doing today? I was wondering, could you help me with a little task? I have a bit of a sweet tooth at the moment. Could you find a cake pan in the room and place it on one of the tables for me? That would be great, thanks!", "tool\_cnt": 5, "action\_cnt": 1162, "action\_list": [ "CHAT", "SEARCH", "GOTO", "PICK", "GOTO", "PLACE", "CHAT" ], "time": 133.73462581634521, "ans\_list": [ "Great to hear that! Take your time, and feel free to let me know if you need any assistance along the way. I'm here to help!", "Target object found! Their names are ['cake\_pan\_1'].", "Go to cake\_pan\_1 successfully! In the current view, you can see ['cake\_pan\_1'].", "Pick up cake\_pan\_1 successfully!", "Go to table\_1 successfully! In the current view, you can see ['folder\_1'].", "Place cake\_pan\_1 on table\_1 successfully!", "You've done an excellent job, thank you! That's all I needed for now. I really appreciate your help. If there's anything else later, I'll be sure to let you know. Have a great day!" ]





**Scores and Reasons:**

Finished: [True] Score: [5] Reason: The robot correctly identified and picked up the cake pan, then successfully placed it on table\_1 as instructed. The task was completed flawlessly with no errors or issues. The picture confirms that the cake pan is placed on table\_1. Thus, I give it a score of 5.

## History:

"task": "Hello! How are you doing today? It's nice to have some company here. I've been wanting to add a bit more ambience to the room. Could you please find a candle and a candle holder, and place them together on one of the tables? It would really help set a cozy atmosphere. Let me know how it goes!", "tool\_cnt": 9, "action\_cnt": 469, "action\_list": [ "CHAT", "SEARCH", "CHAT", "SEARCH", "CHAT", "SEARCH", "CHAT", "OBJECT\_RETRIEVAL", "CHAT", "GOTO", "CHAT", "GOTO", "CHAT", "PICK", "CHAT", "GOTO", "PLACE", "CHAT" ], "time": 107.40170407295227, "ans\_list": [ "That sounds like a great plan! Take your time searching for the candle, and feel free to let me know if you need any assistance or encounter any obstacles along the way. Once you find it, you can search for the candle holder next. You're doing great!", "Target object found! Their names are ['candle\_holder\_1'].", "Great job finding the candle holder! Now, let's see if you can spot the candle. Once you have both items, just place them together on one of the tables. You're almost there, and I appreciate your effort!", "Target objects found in object memory! Their names are ['candle\_holder\_1'].", "No worries at all! Take your time and continue searching for the candle. Sometimes things can be a bit tricky to find. You're doing excellent so far. Once you locate it, you can place both the candle and the candle holder on one of the tables to create that cozy atmosphere. Let me know how it goes!", "Target objects found in object memory! Their names are ['candle\_holder\_1'].", "That sounds like a good idea. Perhaps try looking in spaces where a candle might be stored or placed, like around the tables or shelves. Sometimes they can blend in or be tucked away a bit. Don't hesitate to explore different parts of the room. You've got this, and I'm here to support you if you need any more tips!", "The objects that are most related to 'candle' are 'bottle\_1': 'The object annotated in the image is Lattice\_1, which is represented by a green bounding box around a specific area within the scene. This suggests that Lattice\_1 is an identified region or a significant part of something within the image, potentially indicating a structural element, pattern, or other features that the annotation is highlighting for recognition, analysis, or classification purposes.', 'candle\_holder\_1': 'The object annotated in the image is a chair within the room set-up. Specifically, it is on the right side of the image, near the desk and the bed. The chair is black with a high back and swivel base, and it is used, as indicated by a green marker.', 'helmet\_1': 'A single person standing on a tennis court with a tennis racket and a tennis ball, wearing green and red shorts and a green shirt, holding a tennis ball in their outstretched hand, facing the camera.', 'lamp\_1': 'The object annotated in the image appears to be a circle on the gray area on the floor near the wall.', 'plant\_saucer\_1': 'The object annotated in the image is an office chair with wheels. It is black and has adjustability in the height and tilt.', 'table\_4': 'The object annotated in the image is a chair located near a bed within what appears to be a bedroom setting. The bed has a decorative patterned bedspread, and the background includes a wall-mounted flat-screen TV, a couple of shelves, and a light fixture on the ceiling.'", "Great choice! Searching around the tables or other common areas could definitely be where the candle might be hiding. Sometimes candles are placed on surfaces or tucked into corners. Trust your instincts, and take a careful look around. You've been doing an excellent job so far, and I'm confident you'll find it soon. Keep me posted on your progress!", "Go to table\_1 successfully! In the current view, you can see ['canister\_1', 'plant\_saucer\_1'].", "Sounds perfect! It's good to methodically check each area. Hopefully, table\_2 holds the candle we're looking for. You're thorough in your search, and that's the best way to go about it. Let me know what you find!", "Go to table\_2 successfully! In the current view, you can see ['candle\_1', 'can\_1', 'stuffed toy\_1'].", "Fantastic! I'm so glad you found it. Now that you have both the candle and the candle holder, you can place them on any table that you think would look nice. Once you set them up, it will really help add that cozy touch to the room. Let me know once you're done, and thank you for your persistence and effort!", "Pick up candle\_1 successfully!", "Perfect choice! Placing them on table\_3 sounds like a great spot. Once you have them set up, step back and enjoy the lovely ambience they create. Let me know when you're finished. You've done a fantastic job completing this task, and I genuinely appreciate your help in making the space feel more inviting!", "Go to table\_3 successfully! In the current view, you can see ['pitcher\_1', 'battery\_charger\_1'].", "Place candle\_1 on table\_3 successfully!", "Wonderful! Thank you for setting up the candle and candle holder on table\_3. It truly adds a cozy touch to the room, and you've done an excellent job. I really appreciate your help. If there's anything else you'd like to explore or do, just let me know. Otherwise, enjoy the new ambience you've created!" ]

PLACE table\_3



**Scores and Reasons:**

Output: Finished: [False] Score: [2] Reason: The task required the robot to find and place both a candle and a candle holder together on a table. According to the provided information, the robot only successfully picked up and placed the candle\_1 on table\_3. There is no mention of the candle holder being picked up, transported, or placed on the table. The picture also shows a pitcher and a battery charger on the table, rather than a candle and candle holder. Therefore, the task was not completed as described, resulting in a low score of 2.