

# Supplementary Materials for Pre-emptive Action Revision by Environmental Feedback for Embodied Instruction Following Agents

Anonymous Author(s)

Affiliation

Address

email

1 **Note:** Orange denotes reference to the main paper.

## 2 **A Additional Details of Agent Architecture (L104)**

3 Inspired by the recent success of the SLAM-based approaches [37, 39, 18], we adopt an architecture  
4 that 1) plans a sequence of subgoal actions from a natural language instruction, 2) maintains the  
5 agent’s memory in the form of a semantic spatial map built by an observation history with depth and  
6 masks predicted by pretrained perception models [1], [39], and 3) uses a deterministic algorithm [38]  
7 over the semantic spatial map for effective obstacle-free path planning.

8 **Semantic spatial map.** During exploration, the agent takes as input the current egocentric RGB  
9 observation,  $v_t$ , for each time step,  $t$ , and predicts its semantic information (*e.g.*, object masks,  
10  $\{m_i\}_{i=0}^N$ , a depth map,  $d_t$ , a spatial semantic map,  $Z_t$ , *etc*). Using  $\{m_i\}_{i=0}^N$ , the agent obtains the  
11 cropped images of objects,  $\{v_i^c\}_{i=0}^N$ , from  $v_t$ . Here,  $N$  denotes the number of all detected objects in  
12  $v_t$ , and  $v_i^c$  an object’s image crop with the masked-out background.  $Z_t$  denotes a 2D top-down map  
13 predicted by merging three inputs and is generated by the semantic map generator,  $\mathcal{G}$ , as follows:

$$Z_t = \mathcal{G}(d_t, \{m_i\}_{i=0}^N, Z_{t-1}). \quad (1)$$

14  $G$  transforms each pixel of  $d_t$  and  $\{m_i\}_{i=0}^N$  into a 3D point with its semantic label. These points  
15 are then summed along the gravitational axis to generate the current top-down semantic spatial map  
16 containing only current information. Finally,  $G$  accumulate the obtained top-down map on  $Z_{t-1}$ ,  
17 resulting in an updated semantic spatial map,  $Z_t$ .

18 **Navigation policy.** To interact with an object, an agent must first reach the object in its close  
19 vicinity. For this, previous approaches [35] often use behavior cloning [2] to let the agent mimic  
20 the navigational behavior of a vision-language navigation expert. However, such behavior cloning  
21 requires a large number of training trajectories and natural language annotations for satisfactory  
22 performance, but collecting these may not be trivial due to high computational costs and time.

23 To address this issue, recent approaches [36, 37, 26, 30] instead incorporate deterministic algorithms  
24 (*e.g.*, A\*, FMM [38], *etc*) to plan obstacle-free paths and observe significant improvement of  
25 navigational performance while alleviating the data collection burden. Inspired by this improvement,  
26 we also adopt a deterministic policy [38] to plan navigation routes.

## 27 **B Additional Details of RED**

28 We provide more details of RED. We further detail how we revise an action plan using large language  
29 models (LLMs) based on perceived environmental feedback, illustrated in Figure 1.

30 Additionally, we explain the ‘Appearance Detector Module’ of OHV, which takes egocentric views  
31 in different views as input and predicts the class of the picked-up object; the ‘Attribute Detector

Module' of APM, which takes the cropped image as input and predicts the object's attributes; and the 'Relationship Detector Module' of ASR, which takes the predicted masks as input and predicts the relationship between the target object and where it can be placed.

## B.1 Revising Actions by Environmental Feedback with LLMs (L116)

**Subgoal planning.** Given a natural language instruction,  $\mathcal{X}$ , such as dialogs or directives, a large language model,  $\mathcal{L}$ , predicts the task-relevant contexts,  $\mathcal{E}$ , such as task type, target object, its expected location, *etc.* Then, they are integrated into an LLM-generated high-level action plan,  $\mathcal{T}$ , for the corresponding task type through the integration process, 'Intg.'

Here,  $\mathcal{T}$  is created to satisfy the desired states of the task type. For example, if the task type is "CLEAN ALL X" and X is an object, the desired state is defined as  $\text{object} = \{\text{clean}\}$ . To create the plan, we assume that the initial state is the opposite of the desired one, such as  $\text{object} = \{\text{dirty}\}$ , and then create a plan to clean the object.

Integration of  $\mathcal{T}$  and  $\mathcal{E}$  resembles context-aware planning [18], but we use an unfine-tuned LLM for context prediction instead of a trained model with benchmark data. Specific information obtained from the instruction can be incorporated into the action plan. For example, if the instruction indicates the mug is inside the fridge, an action to open the fridge is added before picking up the mug.

The generated action plan consisting of high-level actions,  $\{a_n^h\}_{n=1}^{N_h}$ , in a triplet format is systematically converted to the executable action plan with low-level actions,  $\{a_n^l\}_{n=1}^{N_l}$ , in a tuple format by a rule-based function,  $f_c$ , following the prior work [18]. This process is expressed as in Equation 2:

$$\mathcal{E} = \mathcal{L}(\mathcal{X}), \quad \{a_{n_h}^h\}_{n_h=1}^{N_h} = \text{Intg}(\mathcal{T}, \mathcal{E}), \quad \{a_{n_l}^l\}_{n_l=1}^{N_l} = f_c(\{a_{n_h}^h\}_{n_h=1}^{N_h}). \quad (2)$$

**Revising actions.** While performing the task, the agent may encounter unexpected environmental discrepancies (see Section 3.1). Once we obtain environmental feedback (*i.e.*, environmental discrepancies) based on visual and language input, we build a prompt,  $\mathcal{P}$  with this feedback to query  $\mathcal{L}$  for plan revision. Specifically, we build the prompt with 1) a system prompt,  $\mathcal{P}_s$ , 2) the original plan (*i.e.*, current action plan in Figure 1),  $\{a_n\}_{n=1}^N$  ( $\{a_n^h\}_{n=1}^{N_h}$  or  $\{a_n^l\}_{n=1}^{N_l}$ ), and 3) the feedback prompt,  $\mathcal{P}_f$ .

The format of the current action plan,  $\{a_n\}_{n=1}^N$ , can be the sequence of the actions either in an executable tuple (low-level action),  $a_{n_l}^l$ , or a triplet (high-level action),  $a_{n_h}^h$ . A triplet action is predefined and consists of a unit of action that includes several executable actions. For example, ('Move', 'Target', 'Parent') signifies the combination of ('Target', 'PickupObject') and ('Parent', 'PutObject'). We operate with both types of action plans simultaneously. For instance, the action plan is input in triplet form if the actions can be revised at the triplet level, while in tuple form if the revision is required at the detailed executable action level. If the output is in triplet form, it is systematically translated into executable tuple form by  $f_c$ . This revision process continues whenever the environmental discrepancy is detected until the agent successfully completes the task.

$\mathcal{P}$  for  $\mathcal{L}$  varies according to the type of discrepancy and given action plan consisting of various objects and actions. We provide several examples of the  $\mathcal{P}$  including  $\mathcal{P}_s$ , a current action plan,  $\mathcal{P}_f$ , and the corresponding output of the  $\mathcal{L}$  in Listing 1 to 5. We provide the prompt to extract  $\mathcal{E}$  in Listing 6 and one to generate a revised action plan in Listing 7.

## B.2 Appearance Detector Module (L149)

To verify that the picked-up object is the intended one, we compare two predicted classes,  $c_0$  and  $c_i$ , from the target object,  $o$ . Here,  $c_0$  is the predicted class of  $o$  from the  $i^0$  viewpoint at the time of interaction, and  $c_i$  is that of  $o$  from the  $i^{th}$  viewpoint when the agent picks up the object and views it from the front. To predict  $c_0$  and  $c_i$ , the agent takes egocentric images at the  $i^0$  and  $i^{th}$  viewpoints. Then, it predicts the class of the object in the center of the image through mask prediction.

The  $i^0$  viewpoint is usually a head-lowered view to simultaneously prevent collisions with obstacles on the floor during exploration and find the object. However, this behavior may cause the agent to see

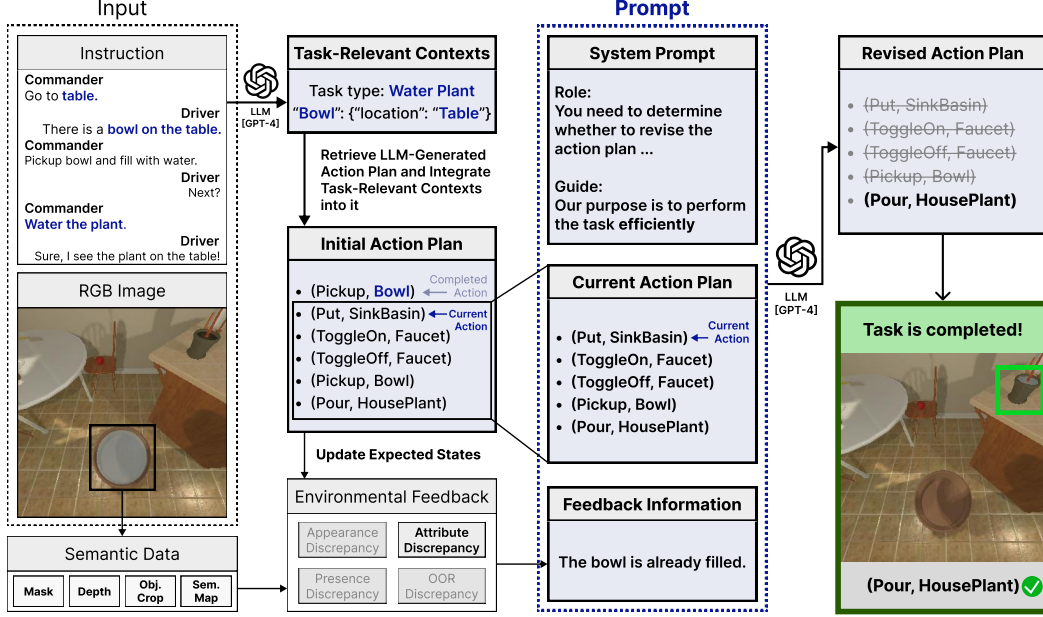


Figure 1: **Overall Process of Revising Actions by LLMs.** We first extract task-relevant contexts from the dialog using an LLM. These contexts are integrated into a retrieved action plan generated by the LLM to form an initial action plan. Using semantic data from an RGB image and expected states from the initial plan, RED verifies discrepancies as environmental feedback. The prompt including the system prompt explaining the role and guide for the LLM, the current action, and the feedback information is given to the LLM to generate a revised action plan. The bottom right image shows the task is completed along with the efficiently revised plan.

the object from a top-down view, resulting in a partial view of only the upper part of the object. To address this issue, the  $i^{th}$  viewpoint is defined such that the agent views the object at a horizontal angle of  $0^\circ$  after picking it up. This allows the agent to clearly see the entire shape of the object without occlusion. By comparing the predicted classes,  $c_0$  and  $c_i$ , from these two viewpoints, the agent can detect differences in appearance. Furthermore, if there is a difference, ( $c_0 \neq c_i$ ), the agent can adjust its actions accordingly.

### B.3 Attribute Detector Module (L169)

We consider attributes related to the desired state to determine the success of the task. Furthermore, the expected attribute is one of the initial states related to the attribute, and the initial states are defined as the opposite of the desired state (detailed in Section B.1).

We take a retrieval-based approach to predict the attribute,  $\hat{\phi}_o$ , of a detected target object,  $o$ . This approach is preferred over training a model because it requires neither extensive training nor large datasets. Additionally, if a new attribute is added, a training-based model must be retrained or fine-tuned, whereas the retrieval-based approach only needs to add new options for comparison.

To predict  $\hat{\phi}_o$ , we retrieve the most similar image and assign the attribute of the object in the retrieved image to the detected target. To compare images focusing only on the object and excluding the background, we use the cropped image of the target object,  $v_o^c$ , and the cropped images,  $\{v_i^{ct}\}_{i=0}^N$ , from the training dataset. We compare these cropped images through cosine similarity, 'Cos', after extracting features processed by a ViT-B/32 model [3], pre-trained with CLIP weights [4], as:

$$\operatorname{argmax}_i \operatorname{Cos}(v_o^c, v_i^{ct}) \quad i = 0, 1, \dots, N, \quad (3)$$

where  $v_o^c$  and  $v_i^{ct}$  are the features of  $v_o^c$  and  $v_i^{ct}$ , respectively. Then, we consider  $\hat{\phi}_o$  as the attribute of the object in  $v_i^{ct}$  having the highest cosine similarity score.

## 98 B.4 Relationship Detector Module (L189)

99 To determine the relationship between the target object,  $o$ , and the object it is placed on,  $o_p$ , we need  
100 to find the most ‘adjust’ mask,  $m_i$ , of  $o_p$  to the target object’s mask in all detected masks,  $\{m_i\}_{i=0}^N$ ,  
101 in the current egocentric view. Here, the mask at  $i = 0$  is defined as the mask of the target object,  
102  $m_0$ , of  $o$ . To find  $m_i$ , we dilate  $\{m_i\}_{i=0}^N$  and calculate the intersection over union (IoU) between the  
103 enlarged mask of the target object,  $m'_0$ , and the enlarged masks of other objects,  $\{m'_i\}_{i=1}^N$  as follows:

$$\operatorname{argmax}_i \operatorname{IoU}(m'_0, m'_i); \quad i = 1, 2, \dots, N. \quad (4)$$

104 We define  $m'_i$  as the mask with the largest IoU score with  $m'_0$ , considering it the ‘adjust’ mask. The  
105 object represented by  $m'_i$  is referred to as the object  $o_p$  that the target object is placed on.

## 106 C Benchmark and Baseline Details (L194)

107 We validate RED in two challenging benchmarks: TEACH [13], for dialog instruction following,  
108 and ALFRED [12], which provides declarative instructions, to assess generalization in different task  
109 setup. We provide details for each benchmark and baselines used below.

### 110 C.1 TEACH

111 **Benchmark.** The TEACH benchmark aims to allow agents to navigate and interact with objects  
112 based on instructions, with task completion achieved by meeting specified conditions, such as cleaning  
113 at least one mug for the instruction “clean a mug”.

114 The instruction is a dialog in natural language, which is comprised of two components: the COM-  
115 MANDER that provides task-relevant information based on oracle information about the task and the  
116 FOLLOWER that performs the task through the dialog. Upon receiving instructions, the FOLLOWER  
117 translates the natural language instructions and egocentric visual observations into executable actions.  
118 The executable actions are expected to succeed in the task.

119 The agent can take 16 different actions. Eight actions (FORWARD, BACKWARD, TURN LEFT, TURN  
120 RIGHT, LOOK UP, LOOK DOWN, STRAFE LEFT, STRAFE RIGHT) are designated for navigation,  
121 and the other eight actions (PICKUP, PLACE, OPEN, CLOSE, TOGGLEON, TOGGLEOFF, SLICE,  
122 AND POUR) are for interaction. Navigation actions are discrete: head movements adjust by  $30^\circ$ , turns  
123 are by  $90^\circ$ , and movements are in 0.25m increments. During interaction, the agent selects the object  
124 at coordinate (x, y) in its egocentric view.

125 Additionally, the TEACH benchmark focus on Execution from Dialogue History (EDH) and Trajectory  
126 from Dialogue (TfD). This benchmark is divided into train, validation, and test splits. Evaluation  
127 metrics encompass success rate (SR), goal-condition success rate (GC), and path-length-weighted  
128 (PLW) scores.

129 **State-of-the-art baseline models.** We compare our RED with the recently proposed state-of-the-  
130 art methods: E.T. [35], JARVIS [36], FILM [37], DANLI [26], and HELPER [30]. E.T. learns a  
131 direct mapping from a natural language dialog and an egocentric observation to a corresponding  
132 action and the position of an object to be interacted with. JARVIS employs an LLM trained on  
133 the TEACH dialog dataset to produce high-level subgoals, replicating the ones executed by human  
134 demonstrators. It utilizes a semantic map alongside the E.T. to locate objects. FILM enhances an  
135 LLM through fine-tuning to generate parameterized plan. Mirroring Jarvis, it leverages a semantic  
136 map to execute subgoals and employs a semantic policy for object search. DANLI fine-tunes an LLM  
137 for high-level subgoal prediction and employs symbolic planning with an object state and spatial map  
138 for execution plan formulation. It incorporates an object search module and manual error correction  
139 mechanisms. HELPER utilizes a Large Language Model (LLM) to generate initial high-level actions  
140 with additional data. When a failure occurs, it predicts the error reasons through a pretrained vision  
141 and language model, and revises the action using the LLM.

Table 1: **Alternative TEACH EDH evaluation split.** The path-length-weighted (PLW) metrics are given in the parentheses for each value. The highest and second highest values per fold and metric are shown in **bold** and underline, respectively.

Model	Validation				Test			
	Unseen		Seen		Unseen		Seen	
E.T. [35]	8.35 (0.86)	6.34 (3.69)	8.28 (1.13)	8.72 (3.82)	7.38 (0.97)	6.06 (3.17)	8.82 (0.29)	9.46 (3.03)
DANLI [26]	<u>17.25</u> (7.16)	23.88 (19.38)	16.89 (9.12)	25.10 (22.56)	16.71 (7.33)	23.00 (20.55)	<u>18.63</u> (9.41)	24.77 (21.90)
HELPER [30]	<u>17.25</u> (3.22)	<u>25.24</u> (8.12)	<u>19.21</u> (4.72)	<u>33.54</u> (10.95)	<u>17.55</u> (2.59)	<u>26.49</u> (7.67)	<u>17.97</u> (3.44)	<u>30.81</u> (8.93)
<b>RED (Ours)</b>	<b>21.52</b> (4.64)	<b>26.88</b> (7.25)	<b>23.84</b> (4.20)	<b>33.79</b> (10.64)	<b>22.04</b> (4.24)	<b>26.77</b> (7.67)	<b>19.61</b> (4.96)	<b>31.86</b> (10.19)

## C.2 ALFRED

**Benchmark.** The ALFRED benchmark requires agents to complete a long-horizon task by understanding declarative natural language instructions with egocentric observations. The declarative instructions comprise two types of instruction: one is a high-level description that provides a single sentence to complete the task and the other is a step-by-step instruction that details the process of performing a task. By following the instructions, the agent executes the two types of predefined actions. The navigation actions include MOVEAHEAD, ROTATERIGHT, ROTATELEFT, LOOKUP, and LOOKDOWN. The interaction actions include PICKUPOBJECT, PUTOBJECT, OPENOBJECT, CLOSEOBJECT, TOGGLEOBJECTON, TOGGLEOBJECTOFF, and SLICEOBJECT.

**State-of-the-art baseline models.** We compare our RED with the recently proposed state-of-the-art methods: HLSC [39], FILM [52], and CAPEAM [18]. HLSC employs a hierarchical controller to translate natural language instructions into actions the agent can execute. The high-level controller identifies the next subgoal based on the given instructions and map, while the low-level controller generates a sequence of actions to accomplish this subgoal. FILM uses a pre-constructed template as a high-level action plan. It employs two BERT [6] classifier submodules to identify the instruction type and determine the template arguments. It applies a deterministic algorithm [38] to plan a path without obstacles. CAPEAM employs context-aware planning to devise a sequence of subgoals and execute each subgoal using the appropriate detailed planners. It also utilizes extra memory to avoid interacting with unsuitable objects.

## D Additional Experiment (L212)

We conduct an additional experiment, exclusively done by DANLI [26] for fair comparison, in the TEACH benchmark [13]. In this benchmark, we investigate the performance of our RED in different splits provided on the TEACH GitHub and in [26]. The leaderboard for EDH of the TEACH benchmark is unavailable, preventing the evaluation on its true test set. Thus, we leveraged the original validation splits for seen and unseen scenarios, aligning with the approach taken in most prior studies [30, 36, 37].

In Table 1, we present the alternative validation and test splits. We observe that our method outperforms others in the new split of EDH, achieving improvements with notable margins in SR and GC, similar to its performance in the original split. In the seen environment, the agent encounters fewer misperceptions and navigation errors than in the unseen environment, making our RED less effective.

## E Qualitative Analysis (L291)

**DTA.** First, Figure 2a describes two different scenarios: RED without (top) and with (bottom) DTA in Section 3.1.1. An agent generates an initial plan considering the information mentioned in the instruction (“mug, potentially in a cabinet”) so that it predicts that the mug is in the cabinet and has a plan of opening the cabinet first, instead of picking up the mug. Thus, in the upper scenario (w/o DTA), the agent keeps its original plan to open the cabinet even if the usable mug is observed in its sight. On the contrary, the agent with DTA changes the plan to skip the action with the cabinet and to

179 pick up the mug right away after perceiving the mug. As a result, it can finish the task efficiently as it  
180 adapts to the discrepancy of the target object’s presence in the environment.

181 **OHV.** Figure 2b describes the difference in the existence of OHV in Section 3.1.2. The first  
182 image shows the agent needing to pick up a ‘Mug’ but mistakenly picking up a ‘Cup’ instead by  
183 misperception. The agents without OHV will not try to correct the wrong action since they do not  
184 consider the appearance discrepancy after the action is done. In contrast, our agent with OHV verifies  
185 whether the interacted object aligns with the desired one by examining its appearance from multiple  
186 angles. If it detects the discrepancy, the agent rectifies its mistake by revising actions. As a result,  
187 OHV helps the agent to prevent the failure that comes from the interaction with the wrong object.

188 **APM.** Next, we investigate the benefit of APM (in Section 3.1.3) depicted in Figure 2c with the  
189 scenarios where agents want to open a microwave. The agent without APM attempts to open it as it  
190 supposes that the microwave is not operating, leading to an interaction failure. In contrast, the agent  
191 with APM considers the attribute discrepancy coming from the difference in the microwave’s state.  
192 With this, it adds actions that toggle it off first to future actions, enabling successful interaction.

193 **ASR.** Finally, we elucidate the advantage of ASR in Section 3.1.4. Figure 2d describes the benefit of  
194 considering relationship discrepancy. We refer to the relationship corresponding to the goal condition  
195 in the instruction as the goal relationship. An agent without ASR may pick up a remote and put it  
196 down with a higher confidence score, even though it already satisfies the goal relationship, resulting  
197 in only one remote on the table. If this were the second interaction (picking it up and putting it  
198 down on the table), the agent would think that it completed the task since it completed interactions  
199 twice. However, the task would fail since there are no two remotes on the table. In contrast, the agent  
200 with ASR considers relationship discrepancy and interacts with a remote not in the goal relationship  
201 instead of the one already in a goal relationship.



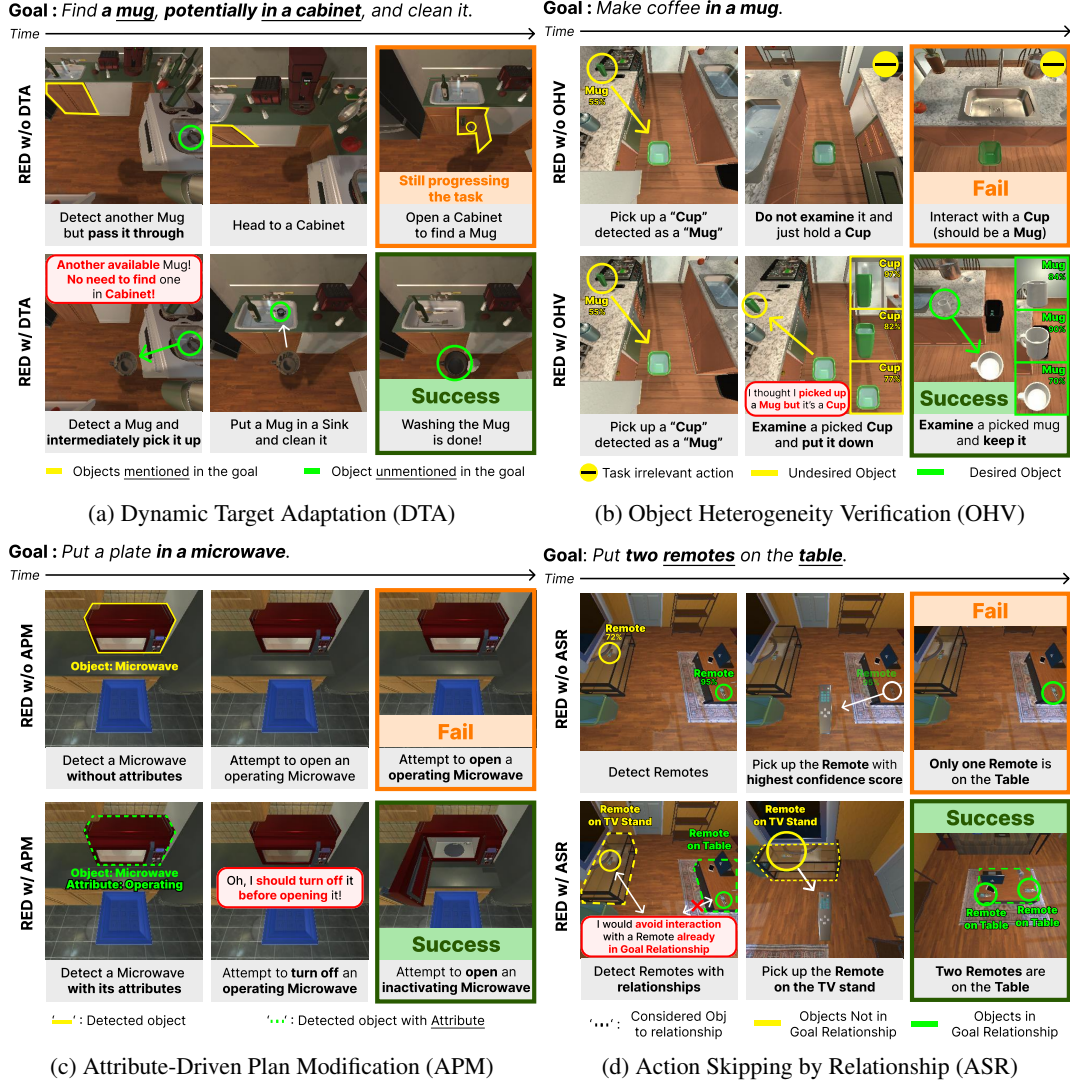


Figure 2: Qualitative analysis of benefit of DTA, OHV, APM, and ASR.

```

202 ### INPUT
203
204 # System Prompt
205
206 You need to determine whether to revise the action sequence to solve
207 the task considering 'INFO' and then write down the final sequence
208 of actions if needed.
209 This action sequence is made for solving household task. Each action's
210 format is tuple.
211 The possible action spaces are as follows.
212 ['Target', 'PickupObject'], ['Target', 'PutObject'], ['Target', '
213 OpenObject'], ['Target', 'CloseObject'], ['Target', '
214 ToggleObjectOn'], ['Target', 'ToggleObjectOff'], ['Target', '
215 PourObject'], ['Target', 'SliceObject']
216 For example, ['Target', 'PickupObject'] is a command to pick up the
217 target.
218 Modify the action sequence by adding or subtracting action to suit the
219 situation if needed. If it is not needed to be revised, just
220 write down given action sequence.
221 In this context, 'action' refers to each element in tuple format in
222 the list(action sequence).
223 You can add or subtract an action at the very first part of the action
224 sequence if needed. Do not modify the actions that follow, and
225 write them down as originally provided.
226 Do not add additional explanation. Just write the final action
227 sequence in the right format (tuples in list).
228
229
230 GUIDE: When you want to pick up an object, it sometimes is located in
231 receptacles that should be opened to pick up the object.
232 In this case, action sequence contains the sequence of open, pickup(or
233 slice), close.
234 Your objective is just picking up(or slicing) the object.
235
236 # Current Action Plan
237
238 [
239     "Recep",
240     "OpenObject"
241 ],
242 [
243     "Target",
244     "PickupObject"
245 ],
246 [
247     "Recep",
248     "CloseObject"
249 ],
250 [
251     "Apple",
252     "SliceObject"
253 ],
254 [
255     "CounterTop",
256     "PutObject"
257 ]
258
259 # Feedback Information
260
261 After checking, the object is found in another place not in the
262 receptacles that should be opened.
263
264 ### OUTPUT
265
266 [

```



```

267         "Target",
268         "PickupObject"
269     ],
270     [
271         "Apple",
272         "SliceObject"
273     ],
274     [
275         "CounterTop",
276         "PutObject"
277     ]
278 ]

```

Listing 1: **Example of the input (*i.e.*, Prompt) and the output (*i.e.*, Revised Action Plan) of LLM.** This example describes the result where DTA is applied. Compared to the current action plan, the action plan is revised to skip the actions (*i.e.*, open and close the receptacle) to make a plan efficient. Target and Recep will be replaced with each corresponding object based on the context.

```

279 ### INPUT
280
281 # System Prompt
282
283
284 You need to determine whether to revise the action sequence to solve
285 the task considering 'INFO' and then write down the final sequence
286 of actions if needed.
287 This action sequence is made for solving household task. Each action's
288 format is tuple.
289 The possible action spaces are as follows.
290 ['Target', 'PickupObject'], ['Target', 'PutObject'], ['Target', '
291 OpenObject'], ['Target', 'CloseObject'], ['Target', '
292 ToggleObjectOn'], ['Target', 'ToggleObjectOff'], ['Target', '
293 PourObject'], ['Target', 'SliceObject']
294 For example, ['Target', 'PickupObject'] is a command to pick up the
295 target.
296 Modify the action sequence by adding or subtracting action to suit the
297 situation if needed. If it is not needed to be revised, just
298 write down given action sequence.
299 In this context, 'action' refers to each element in tuple format in
300 the list(action sequence).
301 You can add or subtract an action at the very first part of the action
302 sequence if needed. Do not modify the actions that follow, and
303 write them down as originally provided.
304 Do not add additional explanation. Just write the final action
305 sequence in the right format (tuples in list).
306
307 GUIDE: When picking up an object, it may not always be the intended
308 object.
309 But you can only handle one object in your hand which means you have
310 to put the object and repick the object if you want to pick
311 another object.
312 When you are not sure where to put the object, just use 'Parent' as
313 the place you put it on. (In the action, ['A', 'PutObject'], 'A'
314 should be the location to place the object on, not the object that
315 you are holding.)
316 Let's assume that you have done the first action.
317 In this case, do not revise the first action, but if you think some
318 actions should be added, do it right after the first action.
319
320 # Current Action Plan
321
322 [
323     "Target",
324     "PickupObject"
325 ],
326 [
327     "Bed",
328     "PutObject"
329 ]
330
331 # Feedback Information
332
333 After checking after the first action (pickup), it turns out that the
334 object picked up is not the object that was intended to be picked
335 up.
336
337 ### Output
338
339 [
340     "Target",
341     "PickupObject"
342 ],
343 [

```

```

344         "Parent ",
345         "PutObject "
346     ],
347     [
348         "Target ",
349         "PickupObject "
350     ],
351     [
352         "Bed ",
353         "PutObject "
354     ]
355 ]

```

Listing 2: **Example of the input (*i.e.*, Prompt) and the output (*i.e.*, Revised Action Plan) of LLM.** This example describes the result where OHV is applied. Compared to the current action plan, the actions (*i.e.*, put the target on the parent and pick up the target again) are inserted to put down the wrong object and pick up an appropriate one. Target and Parent will be replaced with each corresponding object based on the context.

```

356 ### INPUT
357
358
359 # System Prompt
360
361 You need to determine whether to revise the action sequence to solve
362 the task considering 'INFO' and then write down the final sequence
363 of actions if needed.
364 This action sequence is made for solving household task. Each action's
365 format is tuple.
366 The possible action spaces are as follows.
367 ['Target', 'PickupObject'], ['Target', 'PutObject'], ['Target', '
368 OpenObject'], ['Target', 'CloseObject'], ['Target', '
369 ToggleObjectOn'], ['Target', 'ToggleObjectOff'], ['Target', '
370 PourObject'], ['Target', 'SliceObject']
371 For example, ['Target', 'PickupObject'] is a command to pick up the
372 target.
373 Modify the action sequence by adding or subtracting action to suit the
374 situation if needed. If it is not needed to be revised, just
375 write down given action sequence.
376 In this context, 'action' refers to each element in tuple format in
377 the list(action sequence).
378 You can add or subtract an action at the very first part of the action
379 sequence if needed. Do not modify the actions that follow, and
380 write them down as originally provided.
381 Do not add additional explanation. Just write the final action
382 sequence in the right format (tuples in list).
383
384 GUIDE: Microwave cannot be opened if the microwave is toggled on.
385
386 # Current Action Plan
387
388     [
389         "Microwave",
390         "OpenObject"
391     ],
392     [
393         "Mug",
394         "PickupObject"
395     ],
396     [
397         "Microwave",
398         "CloseObject"
399     ],
400     [
401         "SinkBasin",
402         "PutObject"
403     ],
404     [
405         "Faucet",
406         "ToggleObjectOn"
407     ],
408     [
409         "Faucet",
410         "ToggleObjectOff"
411     ]
412
413 # Feedback Information
414
415 As a result of checking at this point, the microwave oven is already
416 turned on.
417
418
419 ### OUTPUT
420

```

```

421     [
422         "Microwave",
423         "ToggleObjectOff"
424     ],
425     [
426         "Microwave",
427         "OpenObject"
428     ],
429     [
430         "Mug",
431         "PickupObject"
432     ],
433     [
434         "Microwave",
435         "CloseObject"
436     ],
437     [
438         "SinkBasin",
439         "PutObject"
440     ],
441     [
442         "Faucet",
443         "ToggleObjectOn"
444     ],
445     [
446         "Faucet",
447         "ToggleObjectOff"
448     ]

```

Listing 3: **Example of the input (i.e., Prompt) and the output (i.e., Revised Action Plan) of the LLM.** This example describes the result where APM is applied. Compared to the current action plan, the action (i.e., toggle off the microwave) is added in the revised action plan to avoid interaction failure.

```

450 ### INPUT
451
452 # System Prompt
453 You need to determine whether to revise the action sequence to solve
454 the task considering 'GUIDE' and then write down the final
455 sequence of actions if needed.
456 This action sequence is made for solving household task. Each action's
457 format is a triplet.
458 There are two options.
459 First, if you think that the first action is not needed, remove that
460 action(triplet) and write down the actions after that.
461 You can only revise the first action; do not modify the actions (from
462 the second actions) that follow, and write them down as originally
463 provided.
464 In this context, 'action' refers to each element in triplet format in
465 the list(action sequence).
466 Second, if you still think the first action is necessary, write it
467 down exactly as received.
468 Do not add additional explanation. Just write the final action
469 sequence in the right format (triplets in the list).
470
471 GUIDE: Our purpose is to perform the task efficiently.
472
473 # Current Action Plan
474
475 [
476     "Clean",
477     "Target",
478     "None"
479 ],
480 [
481     "Move",
482     "Bread",
483     "Toaster"
484 ],
485 [
486     "Move",
487     "Bread",
488     "Toaster"
489 ],
490 [
491     "ToggleOn",
492     "Toaster",
493     "None"
494 ],
495 [
496     "Move",
497     "Bread",
498     "Plate"
499 ],
500 [
501     "Move",
502     "Bread",
503     "Plate"
504 ]
505
506 # Feedback Information
507 After checking, it appears that the target has already been cleaned.
508
509 ### OUTPUT
510
511 [
512     "Move",
513     "Bread",
514

```



```

515         "Toaster"
516     ],
517     [
518         "Move",
519         "Bread",
520         "Toaster"
521     ],
522     [
523         "ToggleOn",
524         "Toaster",
525         "None"
526     ],
527     [
528         "Move",
529         "Bread",
530         "Plate"
531     ],
532     [
533         "Move",
534         "Bread",
535         "Plate"
536     ]
537 ]

```

Listing 4: **Example of the input(*i.e.*, Prompt) and the output (*i.e.*, Revised Action Plan) of LLM.** This example describes the result where APM is applied. Compared to the current action plan, the action (*i.e.*, clean the target) is deleted to skip an unnecessary action. A Target will be replaced with the corresponding object based on the context.

```

538 ### INPUT
539
540 # System Prompt
541 You need to determine whether to revise the action sequence to solve
542 the task considering 'GUIDE' and then write down the final
543 sequence of actions if needed.
544 This action sequence is made for solving household task. Each action's
545 format is a triplet.
546 There are two options.
547 First, if you think that the first action is not needed, remove that
548 action(triplet) and write down the actions after that.
549 You can only revise the first action; do not modify the actions (from
550 the second actions) that follow, and write them down as originally
551 provided.
552 In this context, 'action' refers to each element in triplet format in
553 the list(action sequence).
554 Second, if you still think the first action is necessary, write it
555 down exactly as received.
556 Do not add additional explanation. Just write the final action
557 sequence in the right format (triplets in the list).
558
559 GUIDE: There can be some objects that are already located in the
560 desired destination.
561 If you think executing the following action should be avoided as it is
562 no longer needed, add a Pass action in triplet form (same as
563 given action) with Pass for action, None for Target and Parent,
564 before the given action (including given action) without further
565 explanation. If you think the following actions are still needed,
566 repeat the given actions.
567
568
569 # Current Action Plan
570
571 [
572     "Move",
573     "Target",
574     "Parent"
575 ]
576
577 # Feedback Information
578 After checking the object and its location, it is observed that the
579 object(Target) is already in the desired location(Recep).
580
581 ### OUTPUT
582
583 [
584     "Pass",
585     "None",
586     "None"
587 ],
588 [
589     "Move",
590     "Target",
591     "Parent"
592 ]
593
594

```

Listing 5: **Example of the input(*i.e.*, Prompt) and the output (*i.e.*, Revised Action Plan) of LLM.** This example describes the result where ASR is applied. Given the current action plan (*i.e.*, move a target to a recep), LLM adds 'Pass' action in triplet which leads to skipping the ongoing interaction before the given action. Then the agent will not interact with the target since it is already located in the desired place as explained in the GUIDE. Target and Parent will be replaced with each corresponding object based on the context.

```

595 Driver tries to solve the task. Commander gives information helpful to
596
597     solve the task. You have to get information through the dialog.
598     Find the initial states of the objects and summarize them into a
599     dictionary. If you cannot find proper information in the dialog,
600     you should answer 'X'.
601
602 Just write a dictionary without giving an additional explanation. In a
603     dictionary, fix the keys same with the example answers. Each of
604     the keys has properties(keys) e.g., "location" which contains the
605     initial location of the object. In some tasks, the driver may be
606     asked to find multiple objects of one kind. In those cases, if an
607     object is on Cabinet and another object is on CounterTop, you
608     should output ['Cabinet', 'CounterTop']. (If you cannot know where
609     the potato is initially, answer 'X'.), "receptacle" which is the
610     place that the object should be placed on or in ultimately. "
611     quantity" which represents the number of the objects, "
612     quantity_of_slices" which represents the number of the object's
613     slices. Some objects (e.g., plate, mug) have a key i.e., "Cleaned"
614     which represents whether the object should be cleaned (then write
615     "T") or not (then write "F"). If you cannot find the proper
616     information, just write 'X'.
617
618 Dialog: {DIALOG}
619 Answer:
620

```

Listing 6: **Prompt for Extracting Task-Relevant Contexts from the Dialog.** {} denotes the section in the prompt that is replaced to each corresponding data. This prompt is designed to extract the useful information from the dialog to use it when making an initial action plan.

```

621 # System Prompt
622
623 You need to determine whether to revise the action sequence to solve
624 the task considering 'INFO' and then write down the final sequence
625 of actions if needed.
626 This action sequence is made for solving household task. Each action's
627 format is tuple.
628 The possible action spaces are as follows.
629 ['Target', 'PickupObject'], ['Target', 'PutObject'], ['Target', '
630 OpenObject'], ['Target', 'CloseObject'], ['Target', '
631 ToggleObjectOn'], ['Target', 'ToggleObjectOff'], ['Target', '
632 PourObject'], ['Target', 'SliceObject']
633 For example, ['Target', 'PickupObject'] is a command to pick up the
634 target.
635 Modify the action sequence by adding or subtracting action to suit the
636 situation if needed. If it is not needed to be revised, just
637 write down given action sequence.
638 In this context, 'action' refers to each element in tuple format in
639 the list(action sequence).
640 You can add or subtract an action at the very first part of the action
641 sequence if needed. Do not modify the actions that follow, and
642 write them down as originally provided.
643 Do not add additional explanation. Just write the final action
644 sequence in the right format (tuples in list).
645
646 {GUIDE}
647
648 # Current Action Plan
649 {CURRENTACTIONPLAN}
650
651 # Feedback Information
652 {FEEDBACK}
653

```

Listing 7: **Prompt for Revising Action Plan.** {} denotes the sections in the prompt that are replaced to each corresponding data. The prompt is designed to envelope the system prompt which gives the overall guideline for the LLM's task, the current action plan which is the source plan, and the feedback information which contains the feedback from the environmental discrepancy.

## References

- [1] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017.
- [2] M. Bain and C. Sammut. A framework for behavioural cloning. In *Mach. Intell.* 15, 1995.
- [3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. 2020.
- [4] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.