Supplementary Materials for
Context-Aware Planning and Environment-Aware Memory for Instruction Following Embodied Agents

Byeonghwi Kim Jinyeon Kim Yuyeong Kim[1] Cheolhong Min Jonghyun Choi
Yonsei University [1]Gwangju Institute of Science and Technology
{byeonghwikim,jinyeonkim,cheolhong.min,jc}@yonsei.ac.kr uyeongkim@gm.gist.ac.kr

Note: Blue characters denote the main paper's reference.

A. Details of ALFRED Benchmark

The task goal is to generate a sequence of actions and object masks for interaction with the corresponding objects such that an agent satisfies all conditions that define the task to be successful. If the agent does not satisfy even a single condition, the agent is considered failed at the task.

To train and assess such agents, the benchmark consists of three splits; 'train,' 'validation,' and 'test.' Agents can be trained with the 'train' split and validate their approaches in the 'validation' split with the ground-truth information of the tasks in those splits. The agents are then evaluated in the 'validation' and 'test' split but they do not have any access to the ground-truth information of the tasks.

To complete a task, the agent receives a goal statement that provides a high-level description of the task's goal and step-by-step instructions that provide detailed explanations for how to complete respective steps (*i.e.*, sub-goals). Given the goal statement and step-by-step instruction, the agent receives an egocentric RGB image in the shape of 300×300 and takes an action and an object mask for each time step.

The action space of the agent consists of five navigation actions, seven interaction actions, and a STOP action that indicates the termination of task completion. The navigation actions are MOVEAHEAD for moving ahead, ROTATERIGHT/ROTATELEFT for rotating right/left to 90°, and LOOKUP/LOOKDOWN for looking up/down to 15°. The interaction actions are PICKUPOBJECT for picking up an object (*e.g.*, an apple, a potato, *etc.*), PUTOBJECT for putting an object in a receptacle (*e.g.*, a countertop, a desk, *etc.*), OPENOBJECT/CLOSEOBJECT for opening/closing an object (*e.g.*, a cabinet, a drawer, *etc.*), TOGGLEOBJECTON/TOGGLEOBJECTOFF for turning on/off an object (*e.g.*, a microwave, a lamp, *etc.*), and SLICEOBJECT for slicing an object (*e.g.*, a tomato, a bread, *etc.*). For object interaction, the agent has to additionally predict a binary object mask in the same shape as the RGB image (*i.e.*, 300×300), and the agent interacts with the object to which the highest number of the mask pixels belongs.

For evaluation, the benchmark uses three types of metrics; success rate (SR), goal-condition success rate (GC), and PLW scores (PLWSR and PLWGC). The primary metric is the success rate (SR) which measures the percentage of completed tasks. This metric indicates the task completion ability of the agent. Another metric is the goal-condition success rate (GC) which measures the percentage of satisfied goal conditions. This metric indicates the partial task completion ability of the agent. Finally, path-length-weighted (PLW) scores penalize SR and GC by the length of the agent's actions. This metric indicates the ability to complete tasks efficiently.

B. Additional Qualitative Analysis

B.1. Context-Aware Planning

In the same manner as in Figure 5 and 6 in the main paper, we provide another qualitative example in Figure 1. Figure 1 also shows similar results for the task ("Put a clean spoon in a drawer."). 'CAPEAM w/o CAP' can pick up the relevant object, a spoon, and clean it as described in the instruction but tries to pick up a task-irrelevant object, a ladle, instead of the spoon due to the wrong object prediction, leading to task failure. On the other hand, CAP enables our agent to keep interacting with the task-relevant object (*i.e.*, the spoon) and the agent finally puts the spoon in a drawer as described in the instruction, implying the impact of CAP.

B.2. Environment-Aware Memory

Like Figure 7 and 8 in the main paper, Figure 2 shows the impact of Object Location Caching (Sec. 3.2). This enables our agent to preserve the location and mask of an object with a changed state so that the agent can utilize the information when necessary. Although our agent without EAM ('CAPEAM w/o EAM') successfully slices an object

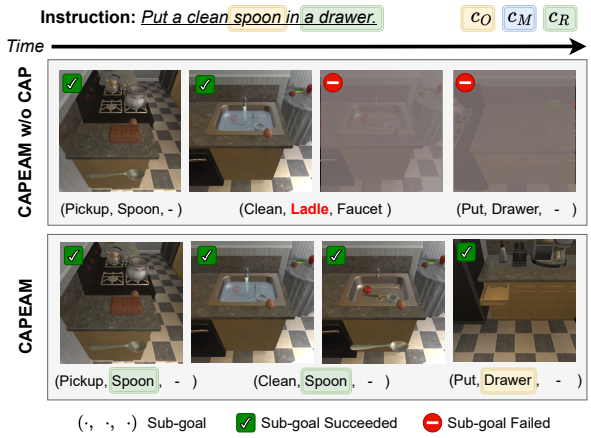

Figure 1: **Another qualitative example of our agent with and without the 'Context-aware Planning' (CAP).** The elements of the 'context' are denoted by c_O in yellow, c_M in blue, and c_R in green. Our method (CAPEAM) plans a sequence of sub-goals with task-relevant objects. However, 'CAPEAM w/o CAP' interacts with task-irrelevant objects (*i.e.*, Ladle), leading to task failure.

(here, an apple) and puts the knife back on the countertop, the agent does not memorize the object's location and therefore it has to explore the environment to navigate to the apple whose state changes (*i.e.*, sliced). In this example, the agent fails at reaching the sliced object and eventually fails at the task. However, our agent equipped with EAM ('CAPEAM') preserves the location and mask of the sliced object and can navigate back to the saved location, which reduces unnecessary exploration and possible task failure. In this example, the agent successfully reaches the sliced objects and moves them to the designated countertop.

C. A Video with Additional Qualitative Results

We also provide an additional qualitative analysis in the attached video files. The video contains two qualitative examples of the ablation of each 'Context-Aware Planning (CAP)' and 'Environment-Aware Memory (EAM),' respectively. The agents for the video take as input only the goal statement (*i.e.*, no step-by-step instructions).

C.1. Example 1: Context-Aware Planning

For the task "Throw two bars of soap in the trash bin," our agent without CAP (*noCAP.mp4*) predicts an object irrelevant to the task (*i.e.*, SoapBottle), which is out of context (*i.e.*, c_O = SoapBar, c_M = None, and c_R = GarbageCan). Although the agent succeeds in implementing the agent-executable actions, they lead to an undesired result (*i.e.*, move a soap bottle, not a soap bar) and thus the agent fails.

However, our agent equipped with CAP (*CAP.mp4*) correctly plans to pick two 'SoapBar' objects and succeeds in taking all the planned executable actions. As the actions

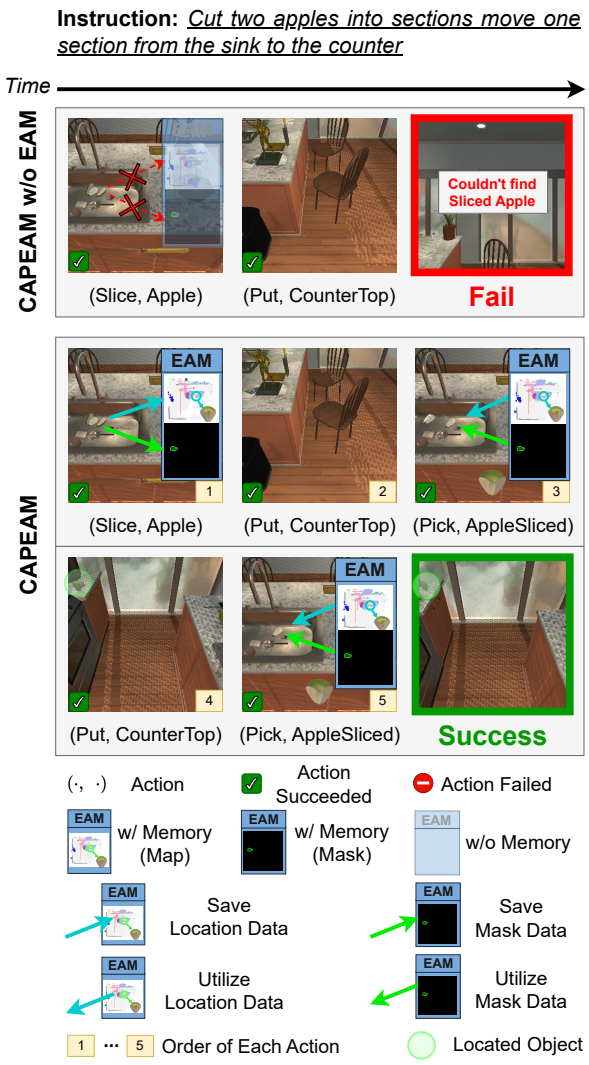

Figure 2: **Another qualitative example of the benefit of 'Environment-aware Memory' (EAM) (for 'Object Location Caching').** While our agent without EAM ('CAPEAM w/o EM') succeeds in slicing the target object (*i.e.*, 'Apple') and putting the knife on the countertop, the agent does not remember the location of the sliced object and therefore it has to explore the environment again to reach the sliced object, eventually leading to task failure. However, after slicing the apple and putting the knife on the countertop, our agent with EAM ('CAPEAM') remembers the location of the object with the changed state (*i.e.*, a 'sliced' apple) with its mask and thus our agent can navigate back to its location and successfully move two sliced apples to the designated countertop.

lead to the desired result (*i.e.*, move two soap bars), the agent finally succeeds, implying the CAP's efficacy.

C.2. Example 2: Environment-Aware Memory

For the task "Put a cup with a fork in it in the sink," while our agent without EAM (*noEAM.mp4*) can predict a cup

without a fork, it cannot recognize the cup with a different visual appearance (*i.e.*, the cup with the fork) and thus the agent interacts with the wrong object, "Pan," and fails.

On the other hand, after putting the fork in the cup, our agent with EAM (*EAM.mp4*) can still recognize the cup thanks to the preserved mask in EM used as an approximation of the mask with a different visual appearance. The agent succeeds in conducting all predicted actions and finally completes the task, which implies the impact of EAM.