LLM-Planner: Few-Shot Grounded Planning for Embodied Agents with Large Language Models

Chan Hee Song
The Ohio State University
song.1855@osu.edu

Jiaman Wu
The Ohio State University
wu.5686@osu.edu

Clayton Washington
The Ohio State University
washington.534@osu.edu

Brian M. Sadler
DEVCOM ARL
brian.m.sadler6.civ@army.mil

Wei-Lun Chao
The Ohio State University
chao.209@osu.edu

Yu Su
The Ohio State University
su.806@osu.edu

Abstract

This study focuses on embodied agents that can follow natural language instructions to complete complex tasks in a visually-perceived environment. Existing methods rely on a large amount of (instruction, gold trajectory) pairs to learn a good policy. The high data cost and poor sample efficiency prevents the development of versatile agents that are capable of many tasks and can learn new tasks quickly. In this work, we propose a novel method, LLM-Planner, that harnesses the power of language language models (LLMs) such as GPT-3 to do few-show planning for embodied agents. We further propose a simple but effective way to enhance LLMs with physical grounding to generate plans that are grounded in the current environment. Experiments on the ALFRED dataset show that our method can achieve very competitive few-shot performance, even outperforming several recent baselines that are trained using the full training data despite using less than 0.5% of the paired training data. Existing methods can barely complete any task successfully under the same few-shot setting.

1. Introduction

We study embodied agents that follow natural language instructions to carry out complex tasks in a partially observable, visually-perceived environment. Recent years have seen surging interests in such embodied instruction following agents [10] based on simulated environments such as Matterport3D [2] and AI2-Thor [16]. Contemporary agents rely on pairs of natural language instructions and demonstrations for learning, which map the instruction (e.g., *"cook a potato"*) to the desired trajectory (i.e., the sequence of primitive actions such as [RotateRight, MoveAhead, Pickup potato, ...]) in a certain environment and starting state.

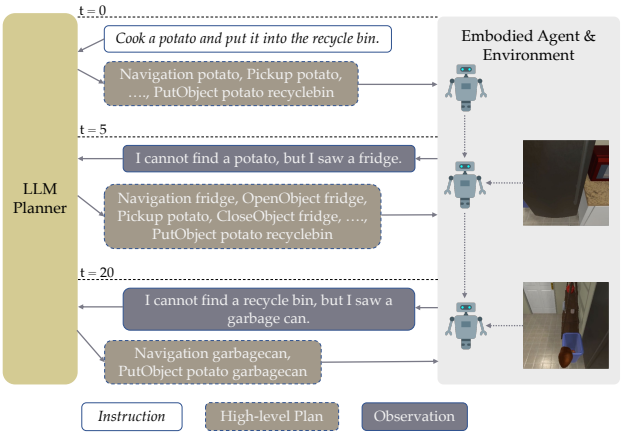

Figure 1. An illustration of LLM-Planner for high-level planning. After receiving the natural language instruction ($t = 0$), LLM-Planner first generates a high-level plan by prompting a large language model (e.g., GPT-3). When the embodied agent gets stuck during the execution of the plan, we re-plan based on observations from the current environment to generate a more grounded plan, which may help the agent get unstuck.

However, such paired data is highly costly to collect, while contemporary agents require a large amount of paired data for learning a task (e.g., thousands of paired examples from the ALFRED dataset [32] for each type of task). In contrast, humans can learn new tasks with very few demonstrations. In this paper, we investigate the question: *How to dramatically improve the sample efficiency of embodied agents so that they can quickly learn a task with only a few paired examples?*

Towards this goal, our first key realization is that natural language instructions do not need to be directly paired with low-level trajectories if we adopt hierarchical planning models (e.g., [31, 35]), which consist of a *high-level planner* and a *low-level planner*. The high-level planner

maps the instruction into a *high-level plan* (HLP), which is a sequence of abstract subgoals (e.g., [Navigation potato, Pickup potato, Navigation microwave, ...]) that the agent needs to reach, in the specified order, to accomplish the final goal. The low-level planner then maps each subgoal into a sequence of primitive actions for accomplishing that subgoal in the current environment and state. *Given a high-level plan, low-level planning is conditionally independent of the natural language instruction*. It becomes the traditional object localization and navigation problem [6] (for navigation subgoals) or simply executing the specified interaction action with the right objects (for interaction subgoals). The low-level planner can be trained with data synthesized from the simulator (see, e.g., [26]). Only the high-level planner needs paired training data in the form of (instruction, HLP).

However, the high-level planner could still require a significant amount of paired training data, especially if we aim for versatile agents that are capable of many different tasks. Intuitively, high-level planning is closely related to commonsense reasoning [36]. In the previous example, the high-level planner's task is to infer that *"cooking a potato"* entails first *"go find a potato,"* then *"pick up the potato"* and *"go find an object, e.g., a microwave, that can be used for cooking,"* and so on. Our second key realization is that, since large language models (LLMs) such as GPT-3 [4] are shown to have captured tremendous commonsense knowledge [36], they can potentially be used as *few-shot high-level planner* for embodied agents.

To this end, we propose LLM-Planner, the first LLM-based high-level planner for embodied instruction following. Specifically, we follow the in-context learning paradigm [4, 20] and only use a small number of paired examples. In addition, no parameter update is needed, which saves development time.[1] For the example in Figure 1, at the beginning of an episode ($t = 0$), given a natural language instruction, we directly prompt an LLM to generate the HLP by giving it several exemplar pairs of (instruction, HLP) in its context. We also leverage established techniques such as dynamic in-context example retrieval [9, 19, 28, 29] and logit biases [11] to further improve the in-context learning performance.

While the HLPs generated by LLMs are already reasonable, they still lack a fundamental aspect of embodied agents — *physical grounding*; i.e., the generated HLP needs to be grounded to the environment the agent is in. Figure 1 demonstrates two common types of planning errors due to lack of grounding: 1) *Object mismatch* ($t = 20$): an object is denoted as garbage can in the environment but the human user refers to it as *"recycle bin."* 2) *Unattainable plans* ($t = 5$): the HLP requires finding a potato, but the only potato in the environment is stored in a closed fridge; the agent will wander endlessly as a result. We propose a novel *dynamic grounded re-planning* algorithm to empower LLM-Planner with physical grounding. Specifically, as an agent is executing the initial HLP, whenever it has taken too many steps to reach the current subgoal or has made too many failed attempts, we dynamically prompt the LLM again to generate a new continuation of the partial HLP that has been completed at that point. For grounding, we add the list of objects perceived in the environment so far into the prompt as a simple but effective description of the current environment. For the example in Figure 1, at $t = 5$, the agent is taking too long to find a potato. It re-prompts the LLM with the object fridge observed in the environment, and LLM-Planner generates a new HLP from scratch (because no subgoal has been completed so far) that directs the agent to look for a potato in the fridge. This is feasible because LLMs may have the commonsense knowledge that food like potatoes are often stored in a fridge, but we need to inform the LLM that there exists a fridge in the current environment. As another example, at $t = 20$, after completing all the subgoals in the HLP but the last one, the agent cannot find the required recycle bin but has observed a garbage can. LLM-Planner thus generates a new continuation of the already-completed partial HLP, which consists of the only remaining subgoal.

We evaluate LLM-Planner on the standard ALFRED [32] household dataset by integrating it with the perception and low-level planner from a strong baseline model, HLSM [3]. Using less than 0.5% paired training data, we show that LLM-Planner achieves comparable performance to HLSM and outperforms multiple other baselines, which are trained with the full training set. Under the same few-shot setting, existing methods can barely complete any task successfully. Our work opens a new door for developing extremely sample-efficient embodied agents by harnessing the power of large language models and grounding.

2. Preliminaries

Vision-and-Language Navigation. Embodied instruction following is often also referred as vision-and-language navigation (VLN), though it additionally involves interaction actions and usually features a much longer time horizon than typical VLN tasks (e.g., Room2Room [2]). To be consistent with the literature, we will use these two terms interchangeably. We will primarily focus on the standard ALFRED [32] dataset, which is built on top of the AI2-Thor [16] simulator, but our method can easily generalize to other datasets and environment. We choose ALFRED mainly considering its diversity in task types (7 different task types) and long-horizon tasks (on average 50 actions per task).

The VLN task is defined as follows: Given a language

[1] In the experiments we use 100 (instruction, HLP) pairs compared with 21,023 (instruction, trajectory) pairs used by existing methods.

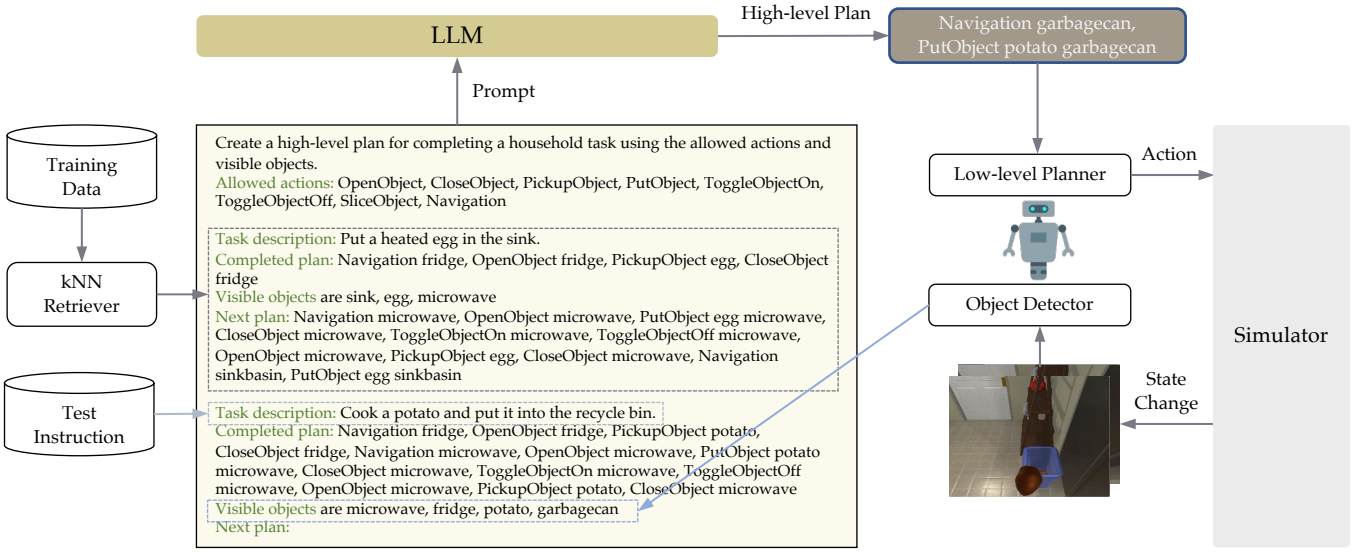

Figure 2. Overview of LLM-Planner with prompt design and dynamic grounded re-planning.

instruction I, an agent needs to predict and carry out a sequence of primitive actions in the environment E to accomplish the task. In datasets like ALFRED [32], the instruction I consists of a high-level goal I_H and (optionally) a list of step-by-step instructions I_L, as exemplified in Figure 2. A VLN task can thus be represented by a tuple (I, E, G), where G is the goal test of the task. We consider hierarchical planning models [35] for VLN, which is explored to various extent in several recent studies [3, 26, 31, 33], but none of them considers the few-shot setting or LLMs for planning. In this formulation, planning is modeled in a hierarchical fashion. The high-level planner maps the instruction I into a high-level plan (HLP) $L_h = [g_0, g_1, \cdots, g_T]$, where each subgoal g_i is specified as (high-level action, object). We define a high-level action to be a collection of primitive actions that can complete a single goal-condition in ALFRED [32]. However, while ALFRED contains primitive locomotive actions, e.g., (MoveForward, 25) and (RotateRight, 90), with specifiable changes in magnitude, all the interaction actions are abstracted into a single action, e.g., (PickupObject, knife), without any changes to the agent's posture. Therefore, the high-level action space consists of 1 navigation action (Navigation) and 7 interaction actions (PickupObject, PutObject, OpenObject, CloseObject, ToggleOnObject, ToggleOffObject, SliceObject).

The low-level planner maps each subgoal g_i into a sequence of primitive actions $L_l^i = \left[a_i^0, a_i^1, \cdots, a_i^{T_i}\right]$. State-of-the-art VLN methods [3, 26] use a map-based low-level planner and a simple path-finding algorithm to find the target object in the current subgoal from the map. It is important to note that, once the high-level plan L_h is specified, the low-level planning becomes independent of the instruction I. More formally, $P(L_l|I, L_h, E) = P(L_l|L_h, E)$. All the components involved in the low-level planner are either deterministic or trained using synthetic data from the simulator. No paired data involving instructions is needed.

In-Context Learning/Prompting. Recently, in-context learning (also known as prompting) [4] have drawn great attention with the rise of LLMs. By designing different prompts, LLMs can be adapted to different downstream tasks with a few examples as demonstration without updating any of the parameters. In this work, we explore in-context learning with LLMs for embodied agent planning.

True Few-Shot Setting. While only using a small number of training examples, many few-shot studies use a large validation set for prompt design and model selection [4]. Recent studies [28] have shown that such large validation sets are responsible for overestimation of the efficacy of language models because they create a strong bias for model selection and violate the intended few-shot setting. To avoid such bias, we adhere to the true few-shot setting [28] in which prompt design and model selection is conducted via cross-validation on the same small training set instead of using a separate validation set.

3. LLM-Planner

In this section, we describe our method LLM-Planner, which leverages LLMs such as GPT-3 to do few-shot grounded high-level planning for embodied agents.

3.1. Overview

LLMs such as GPT-3 are pre-trained to generate natural language. To adapt them as high-level planners, the first

step is to design an appropriate prompt to guide them to generate high-level plans. We discuss our prompt design in Section 3.2. The choice of in-context examples is critical for the performance of LLMs, and recent work [9, 28] has shown that dynamically retrieving similar examples for each test example is beneficial. We adopt a k-nearest-neighbor (kNN) retriever to select the in-context examples (Section 3.3). We also use logit biases [11] to further constrain the output space of the LLM to the allowed set of actions and objects (Section 3.4). With all the above designs, we have obtained the *static* version of LLM-Planner, which can already generate reasonable HLPs. In Section 3.5, we propose a novel dynamic grounded re-planning algorithm to enhance LLMs with the ability to ground to the current environment, which further improves the HLP quality. Finally, we discuss how to integrate LLM-Planner into existing embodied agents to empower them with few-shot planning capabilities in Section 3.6. An overview of LLM-Planner is shown in Figure 2.

3.2. Prompt Design

While GPT-3 is shown to be powerful few-shot learner in variety of tasks, its power can only be unleashed with carefully designed prompts that are tailored for the desired behavior. The final HLP quality can be sensitive to minor design choices in the prompt (e.g., how the HLP is presented, or sometimes even the choice of punctuation). Therefore, we identify core components of the prompt and systemically compare different design choices under the true few-shot setting based on leave-one-out cross-validation (LOOCV). The evaluation for some of the key design choices is discussed in Section 4.5 and 4.6.

Our final optimal prompt is shown in Figure 2. The prompt begins with an intuitive explanation of the task and the list of allowable high-level actions. It is then followed by the in-context examples selected by the kNN retriever (Section 3.3). When we provide only the high-level goal instruction to GPT-3, we use the format "Task description: [high-level goal instruction]." When we include the step-by-step instructions, we include another line "Step-by-step instructions: [step-by-step instructions]" following the goal instruction. For dynamic grounded re-planning (Section 3.5), we add the subgoals that have been completed and the list of objects observed so far in the environment after the task description. Finally, we append the test example in the same format that ends with "Next plan:".

3.3. In-context Example Retrieval

The in-context examples are an important source of task-specific information for the LLM. Different examples could provide different information for the current task. Intuitively, if the current task is to *"cook a potato,"* an in-context example that demonstrates the HLP for *"cooking an egg"* is likely more informative than one that demonstrates how to *"clean a plate."* Specifically, we use a frozen BERT-base model [7] to evaluate the pairwise similarity between each training example and the current test example. The similarity of two examples is defined based on the Euclidean distance between the BERT embedding of their corresponding instruction. For each test example, we then retrieve the K most similar examples from the small set of paired training examples we have, where K is a hyperparameter that we tune under the true few-shot setting (Section 4.6).

Algorithm 1 Dynamic Grounded Re-planning with LLM-Planner

$I \leftarrow$ Instruction
$O \leftarrow$ Set of observed object
$G \leftarrow$ List of completed subgoals so far
$S \leftarrow$ LLM-Planner(I, O, G) ▷ Full HLP
$t \leftarrow 0$ ▷ Time step
$k \leftarrow 0$ ▷ Subgoal index
$s \leftarrow S[k]$ ▷ First subgoal
$a_t \leftarrow$ Low-Level-Planner(s) ▷ First action
while k <len(S) **do**
  execute a_t
  $O_t \leftarrow$ Object-Detector(current camera input)
  O.insert(O_t)
  if current subgoal s fails or after n time steps **then**
    $S \leftarrow$ LLM-Planner(I, O, G) ▷ New HLP
    $k \leftarrow 0$
    $s \leftarrow S[k]$
  else if current subgoal s is completed **then**
    $k \leftarrow k + 1$
    $s \leftarrow S[k]$ ▷ Get next subgoal
  end if
  $t \leftarrow t + 1$
  $a_t \leftarrow$ Low-Level-Planner(s)
end while

3.4. Logit Biases

Since the HLPs that LLM-Planner needs to predict consist of tokens from a fixed set of actions and objects, we leverage the *logit bias* option of the GPT-3 API to constrain the output space. By providing a list of allowable actions and objects to GPT-3, we can enforce it to allocate the majority of its probability mass during generation to those tokens and reduce the the generation of invalid tokens. The strength of this constraint is controlled by the value of the logit bias, with a larger value indicating a stronger constraint. Specifically, we add the same logit bias to all tokens in the list of actions, visible objects (for dynamic planning), or all possible objects (for static planning), and the value is selected through cross-validation.

3.5. Dynamic Grounded Re-planning

Using LLM-Planner as a *static* high-level planner that only predicts an HLP at the begining of a task already shows good data efficiency and accuracy. As discussed earlier, however, such static planning lacks physical grounding to the environment and can lead to incorrect objects and unattainable plans (Figure 1). When such issues happen, the agent cannot complete the current subgoal specified in the HLP, which will lead to one of two possible situations: 1) it fails to execute an action (e.g., bumping into a wall or failing to interact with an object), or 2) it takes a long time and still has not completed the current subgoal (e.g., wandereing endlessly). Intuitively, knowing the objects in the current environment can be very helpful for addressing both of these issues. For example, knowing that there is a fridge, the LLM may produce an HLP that directs the agent to go to the fridge and try to find a potato in that, because it may have learned the commonsense knowledge that food is likely stored in a fridge.

To this end, we present a simple but novel way to enhance LLM-Planner with physical grounding by injecting a list of observed objects, which may be detected using the object detector of the embodied agent, from the environment into the prompt (Figure 2). We add logit biases to these observed objects so LLM-Planner can prioritize producing a plan with those objects if they are relevant for the current task.

With such flexible grounded re-planning capability, we further propose a dynamic grounded re-planning algorithm (Algorithm 1) to dynamically update the HLP during the course of completing a task. This is in contrast with most existing work that adopts a similar hierarchical planning model [26], which only predicts a fixed HLP up front and sticks to that no matter what happens during the execution. In our algorithm, re-planning will be triggered under either of two conditions: 1) the agent fails to execute an action, or 2) after a fixed number of time steps. A new continuation of the already-completed partial HLP will be generated LLM-Planner based on the observed objects, and the agent will carry on with the new plan, which may help it get unstuck.

3.6. Integration with Existing VLN models

We now discuss how to integrate LLM-Planner with the existing models to empower them with the few-shot planning capability. LLM-Planner provides a fairly generic and flexible interface for integration. As shown in Algorithm 1, it only needs the embodied agent to provide an object list and has a low-level planner that can turn the predicted HLP into low-level actions. It has no assumption about the inner working of the agent. For evaluating the end-to-end task completion performance of LLM-Planner, we integrate it with a strong baseline method, HLSM [3], which satisfies such an interface.

4. Experiments

4.1. Dataset

We evaluate the efficacy of LLM-Planner in generating high-level plans using the ALFRED [32] benchmark, a vision-and-language navigation dataset that requires embodied agents to follow instructions and use visual input to complete tasks in a simulated, spatially continuous household environment. The dataset consists of 7 task types, ranging in difficulty from moving a single object to a new location to placing a heated slice of an object into a receptacle. Each task is accompanied by human-written annotations of a high-level goal and a series of more granular step-by-step instructions, created by human annotators as they watched expert demonstrations of the tasks. Due to the noise in the natural language instructions and the complexity of planning required to complete such long-horizon tasks, ALFRED is a challenging test of an embodied agent's ability to produce robust and accurate plans. We use stratified random sampling to choose 100 training examples that are representative of the 7 task types.

4.2. Metrics

We report two main metrics used by ALFRED and one metric created by us to calculate the high-level planning accuracy. Success rate (SR) is the percentage of tasks fully completed by the agent. A task is only considered complete when all the subgoals are completed. Therefore, each task can either be successfully or unsuccessful and success rate does not take account of partially succeeded task. Goal-condition success rate (GC) is the percentage of completed goal-conditions. Goal-conditions are defined as collection of state changes necessary to complete the task. For example, in the task *"Slice a heated bread"*, bread being sliced and bread being heated are both goal-conditions.

To directly evaluate high-level planning, we introduce a new metric named high-level planning accuracy (HLP ACC), i.e., the accuracy of the predicted HLP compared to the ground-truth HLP. For the static planning setting, we compare the generated HLP with the ground-truth HLP and deems a plan as incorrect if it does not perfectly match the ground truth, and correct otherwise. For the dynamic planning setting, we report a range since we do not know what the future HLP will be if the low-level planner/controller fails to execute a subgoal. Specifically, the upper bound of the range is obtained by treating the an HLP as correct if it perfectly matches a certain prefix of the ground truth. The lower bound is obtained in the same way as the static planning setting.

4.3. Implementation Details

We choose 100 examples for our LLM-Planner among 21,023 ALFRED training examples. We apply stratified

Model	Test Unseen		Test Seen		Valid Unseen			Valid Seen		
	SR	GC	SR	GC	SR	GC	HLP ACC	SR	GC	HLP ACC
Full-data setting: 21,023 (instruction, trajectory) pairs										
Goal instruction only										
HiTUT [38]	11.12	17.89	13.63	21.11	10.23	20.71	–	18.41	25.27	–
HLSM [3]	20.27	27.24	25.11	35.79	**18.28**	**31.24**	31.24 – 70.1	29.63	38.74	38.74 – 77.6
Step-by-step instructions										
E.T. [27]	8.57	18.56	**38.42**	**45.44**	7.32	20.87	–	**46.59**	**52.92**	–
HiTUT [38]	13.87	20.31	21.27	29.97	12.44	23.71	–	25.24	34.85	–
M-TRACK [33]	16.29	22.60	24.79	33.35	17.29	28.98	–	26.70	33.21	–
FILM [26]	27.80	**38.52**	28.83	39.55	–	–	54.93	–	–	60
LEBP [18]	**28.30**	36.79	28.97	36.33	–	–	–	–	–	–
Few-shot setting: 100 (instruction, high-level plan) pairs										
Goal instruction only										
LLM-Planner (Static) + HLSM	11.58	18.47	13.05	20.58	11.10	22.44	28.67	11.82	23.54	27.45
LLM-Planner + HLSM	13.41	22.89	15.33	24.57	12.92	25.35	33.81 – 55.85	13.53	28.28	35.08 – 54.33
Step-by-step instructions										
HLSM [3]	0.6	3.72	0.8	6.88	0.0	1.8	0.0	0.1	2.8	0.0
FILM [26]	0.2	6.71	0.0	4.23	0.0	9.65	0.0	0.0	13.19	0.0
LLM-Planner (Static) + HLSM	15.83	20.99	17.87	23.10	14.26	26.12	43.24	15.84	25.43	39.87
LLM-Planner + HLSM	**16.42**	**23.37**	**18.20**	**26.77**	**15.36**	**29.88**	46.59 – 68.31	**16.45**	**30.11**	50.33 – 71.84

Table 1. Main results on the ALFRED dataset. "(Static)" means the static planning setting, otherwise it is the default dynamic grounded planning setting. Some methods support using only the goal instruction or additionally using the step-by-step instructions. We compare under both configurations. LLM-Planner achieves a strong performance that is competitive with existing methods despite using less than 0.5% of paired training data. Under the same few-shot setting, existing methods can barely complete any task successfully.

random sampling to ensure we have a fair representation of all 7 task types in the 100-example set. For the kNN retriever, we use the pretrained BERT-base-uncased model from the Huggingface Transformers Library [37]. For the LLM, we use the public GPT-3 [4] API with 9 in-context examples chosen from the 100 training examples by the kNN retriever. We set the temperature to 0 and apply a logit bias of 0.1 to all allowable output tokens as described in Section 3.4. The object list for dynamic grounded re-planning is retrieved from the object detector. Specifically, we use the pretrained object detector from HLSM's perception model to retrieve the label of all detected objects. We limit the number of detected objects to 10 and only include objects with a label confidence more than 80% to reduce noise. It is worth noting that we can potentially use any object detector to obtain the object list, and we only use HLSM's perception model to save computation cost and time. To avoid violating our few-shot assumption, we use the pretrained navigation, perception, and depth model from HLSM which are trained using only synthesized trajectories from the simulator, without any paired training data involving natural language instructions or human annotations.

4.4. Main Results

The main results are shown in Table 1. We first compare the performance of HLSM when using our LLM-Planner as the high-level planner compared with its native version, which is trained using the full training set of ALFRED. We find that LLM-Planner's few-shot performance is competitive to the original HLSM, and outperforms several recent baselines such as E.T., HiTUT, and M-TRACK, despite using less than 0.5% of paired training data. On the other hand, when trained using the same 100 examples (i.e., retraining HLSM's high-level planner), HLSM (and FILM as well) can barely complete any task successfully. We leave the other modules, such as navigation, depth estimation, and perception, intact since they do not use the paired data involving natural language instructions. Furthermore, we see a considerable improvement from dynamic grounded re-planning over static planning, especially in the goal instruction only setting, where it improves 1.83% SR in the unseen test split. This confirms the effectiveness of the dynamic grounded planning. But we also note that there is still a large room for further improvement.

	LOOCV HLP accuracy
Best Model	40.59
– kNN Retriever	17.48
– Logit Biases	38.10
– Both	13.43

Table 2. LLM-Planner's component ablation using LOOCV.

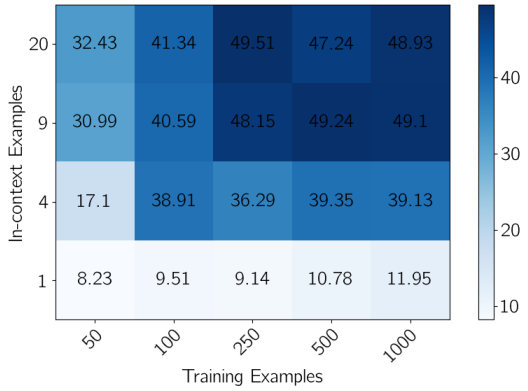

Figure 3. LOOCV HLP accuracy results for number of in-context examples vs. number of training examples

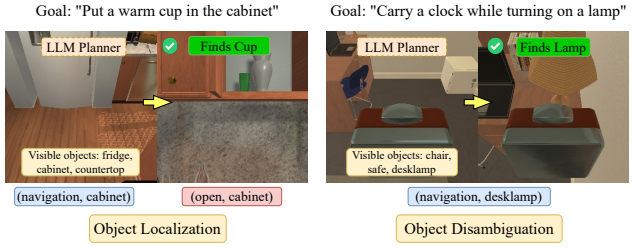

Figure 4. Case studies for LLM-Planner

4.5. Ablation Study for LLM-Planner

General Ablation We ablate on different components of LLM-Planner to demonstrate the importance of techniques to use to improve the LLM-Planner's performance. Mainly we ablate on the using the kNN retrieval and the logit biases for GPT-3. We follow the LOOCV process and use only the high-level planning accuracy to determine our choices. Results from this study are in Table 2. We first ablate the kNN retriever module, by replacing it with a retriever that randomly selects in-context examples from the 100 example set. As we can see from Table 2, removing kNN retriever and letting it select random (and often unrelated) examples lead to a significant decrease in HLP accuracy. This is expected because the getting relevant in-context examples is crucial to GPT-3 because it will bias it's HLP prediction to follow similar trajectory as the relevant in-context examples.

Furthermore, we found out that enabling logit biases to favor objects that appear in the environment lead to a decent boost in the high-level planning accuracy. By having GPT-3 favor objects that appear in the environment makes GPT-3 more robust in the cases where the instruction is ambiguous or objects are referred as different names. For example, for an instruction *"Turn on the lamp"*, GPT-3 have no idea what type of lamp does the instruction means. By enabling the logit biases to favor objects that appear in the environment (e.g. TableLamp), we can correctly guide GPT-3 to output (TurnOnObject, TableLamp). Another example is when the instruction refers to RecycleBin but the object name used in the environment is GarbageCan. In this case, enabling logit biases does help in these scenarios and can correctly guide the GPT-3 to output the relevant and correct objects.

For each setting including plans generated by LLM-Planner, temperature, and k (number of in-context examples) hyperparameters were tuned using 5-fold cross-validation on the training set of 100 examples chosen by stratified random sampling to ensure all task types are represented in the set.

KNN Retriever Ablation The embedding type for the kNN retrieval module is a BERT sentence embedding of the high-level instructions in settings that omit step-by-step instructions and a BERT sentence embedding of concatenated high-level and step-by-step instructions in the settings that include step-by-step instructions. BERT embeddings were selected over RoBERTa [22] embeddings through LOOCV on the small training set in which the BERT embedding module retrieved at least one in-context example with the same ground-truth action sequence as the held-out example at a higher rate than the RoBERTa embedding module.

4.6. Fine-grained Analyses

Effect of Number of Examples To validate our choice of number of training examples for the kNN retriever and number of in-context examples for the prompt, we perfrom LOOCV on different number of both hyperparameters. As shown in Figure 3, we find out that the HLP accuracy reaches a ceiling around 500 examples and that simply increasing the training example size does not lead to the improvement in HLP accuracy. Furthermore, we find that the optimal number of in-context examples is 9. Although adding more in-context examples do not usually hurt the performance, we found out that it is not meaningful enough to justify additional computational cost.

Case Studies We show 2 examples where LLM-Planner does object localization and disambiguation in the environment in Figure 4. For the first case, even using only the high-level goal instruction, the LLM-Planner could successfully predict that the cup is likely to be located in the cabinet after failing to pick up the cup in the environment. This shows the semantic knowledge of the LLM-Planner

and comparable to what semantic map tries to achieve in FILM [26]. For the second case, we show that the LLM-Planner can correctly ground the word *"lamp"* to the desklamp in the environment by using the grounding environment (visible object list) as a part of the prompt to the LLM-Planner.

5. Related Work

5.1. Vision-and-language Navigation

As vision-and-language navigation task gets more complex, various models have been proposed that shifted the trends from the existing single-neural network model architectures. In navigation-only datasets such as R2R [2], models that create the output action sequence end-to-end by utilizing a Transformer achieves high performance [27, 34]. With the emerge of large-scale pre-trained language models (LLM) [4, 7, 22], the instructions are encoded by LLM models with higher generalizability for downstream navigation task [17]. In the meantime, transformer has proven its power in vision [8] thus leading to a number of Transformer-based model [12, 24, 25] that jointly learns the linguistic and visual representations with cross-attention to be grounded on the environment. However, in the navigation and interaction dataset such as ALFRED [32], models [3, 18, 26] that separated the high-level and low level planning interface have been dominated the leaderboard.

To tackle the challenges of navigating in partial observed environment, [3, 18, 21, 26] use a pretrained language model to generate high-level plans and leverage a semantic map to guide the agent achieving the target objects in the high-level plans for low-level planning.

This modular approach has shown strong performance where the training data has been hard to collect since each modular component can be trained independently to further improve the performance. $(SL)^3$ [31] only uses 10% of annotated data to learn how to generate natural language subtasks with goals and then match admissible actions to subtasks. We take this modular approach one step further and propose to use a few-shot setting on large language model (LLM) to conduct high level planning and utilize it's rich pre-trained information.

5.2. General In-context Learning and Prompting

In-context learning [4] uses few examples in prompts as a sample of the task to LLM without requiring additional fine-tuning in different tasks. The way to choose optimal in-context examples is also studied [19]. However, it is unknown how the in-context learning is well-grounded to other modality (vision). In this paper, we present a way of grounding environment in a form of object list.

LLM is very sensitive to the prompt design, especially in few-shot or zero-shot setting. Simply adding the significant keywords, such as a description of task, may increase the performance dramatically [15]. Prompt design is an important aspect of in-context learning and some recent work like [28] proposes true few-shot setting which does not use an additional validation set to choose prompt templates or hyperparameters.

5.3. Prompting for VLN

The use of large language model for decision making has been a increasingly popular topic due to the amazing capability of the large language models to perform multi-task generalization with a limited number of training data. Previous work has either pre-trained a large Transformer model on image-text data [30] or apply additional re-ranking or alignment to the generated plan [23, 31].

JARVIS [39] applies few-shot learning by fine-tuning BART with free-form dialogues and previous high-level plans to generate the next high-level plan. PROG-PROMPT [13] utilizes the strengths of LLM (e.g., GPT-3 [4] and Codex [5]) in understanding programming language to format the instructions as program-like prompt. LM-Nav [30] prompts LLM with raw navigation instructions and 3 examples to generate a list of landmarks for a vision-language model to infer a joint probability distribution over landmarks and images [14] asks sufficient large language model (GPT-3) to generate a intermediate plan given a prompt with an in-context example and a goal, and the intermediate plan, by BERT-style LLM, is converted to an admissible action, which is later appended to the prompt for the future plan. Saycan [1] ranks the score of a list of pre-defined admissible action from LLM by prompting in-context examples, combining with an affordance function which assigns higher weights to the objects appearing in the current scene. However, none of them dynamically insert the object information retrieved from the environment to the prompts during the exploration nor plans for more than a single time step forward. On the other hand, LLM-Planner grounds the environment while dynamically planning long-horizon subgoals for a given natural language instruction.

With appropriate prompt design and in-context example retrieval, we show that LLM can generate complete high-level plans grounded on the environment without any post-filtering mechanisms to perform similarly compare to the given methods with only fraction of data.

6. Conclusion

We demonstrate that large language models (LLMs) can be used as a planner for embodied agents and can dramatically reduce the amount of human annotations needed for learning the instruction following task. Our work opens a new door for developing extremely sample-efficient embodied agents by harnessing the power of large lan-

guage models and enhancing them with physical grounding. Promising future directions include exploring other LLMs such as Codex [5], better prompt design, and more advanced methods for grounding and dynamic replanning.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022. 8

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sunderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, pages 3674–3683, 2018. 1, 2, 8

[3] Valts Blukis, Chris Paxton, Dieter Fox, Animesh Garg, and Yoav Artzi. A persistent spatial semantic representation for high-level natural language instruction execution. In *Conference on Robot Learning*, pages 706–717. PMLR, 2022. 2, 3, 5, 6, 8

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. 2, 3, 6, 8

[5] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, S. Arun Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021. 8, 9

[6] G.N. Desouza and A.C. Kak. Vision for mobile robot navigation: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237–267, 2002. 2

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. 4, 8

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 8

[9] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online, Aug. 2021. Association for Computational Linguistics. 2, 4

[10] Jing Gu, Eliana Stefani, Qi Wu, Jesse Thomason, and Xin Wang. Vision-and-language navigation: A survey of tasks, methods, and future directions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7606–7623, 2022. 1

[11] Bernal Jiménez Gutiérrez, Nikolas McNeal, Clay Washington, You Chen, Lang Li, Huan Sun, and Yu Su. Thinking about gpt-3 in-context learning for biomedical ie? think again. *arXiv preprint arXiv:2203.08410*, 2022. 2, 4

[12] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. A recurrent vision-and-language bert for navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1643–1653, June 2021. 8

[13] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. *arXiv preprint arXiv:2210.05714*, 2022. 8

[14] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022. 8

[15] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022. 8

[16] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Ab-

hinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017. 1, 2

[17] Xiujun Li, Chunyuan Li, Qiaolin Xia, Yonatan Bisk, Asli Celikyilmaz, Jianfeng Gao, Noah A. Smith, and Yejin Choi. Robust navigation with language pretraining and stochastic sampling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1494–1499, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. 8

[18] Hao Liu, Yang Liu, Hong He, and Hang Yang. Lebp - language expectation & binding policy: A two-stream framework for embodied vision-and-language interaction task learning agents. *ArXiv*, abs/2203.04637, 2022. 6, 8

[19] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online, May 2022. Association for Computational Linguistics. 2, 8

[20] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*, 2021. 2

[21] Xiaotian Liu, Hector Palacios, and Christian Muise. A planning based neural-symbolic approach for embodied instruction following. *Interactions*, 9(8):17, 2022. 8

[22] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Ro{bert}a: A robustly optimized {bert} pretraining approach, 2020. 7, 8

[23] Lajanugen Logeswaran, Yao Fu, Moontae Lee, and Honglak Lee. Few-shot subgoal planning with language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5493–5506, Seattle, United States, July 2022. Association for Computational Linguistics. 8

[24] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32, 2019. 8

[25] Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. Improving vision-and-language navigation with image-text pairs from the web. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI*, pages 259–274, 2020. 8

[26] So Yeon Min, Devendra Singh Chaplot, Pradeep Kumar Ravikumar, Yonatan Bisk, and Ruslan Salakhutdinov. FILM: Following instructions in language with modular methods. In *International Conference on Learning Representations*, 2022. 2, 3, 5, 6, 8

[27] Alexander Pashevich, Cordelia Schmid, and Chen Sun. Episodic Transformer for Vision-and-Language Navigation. In *ICCV*, 2021. 6, 8

[28] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models, 2021. 2, 3, 4, 8

[29] Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, 2021. 2

[30] Dhruv Shah, Błażej Osiński, Sergey Levine, et al. Robotic navigation with large pre-trained models of language, vision, and action. In *6th Annual Conference on Robot Learning*, 2022. 8

[31] Pratyusha Sharma, Antonio Torralba, and Jacob Andreas. Skill induction and planning with latent language. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1713–1726, Dublin, Ireland, May 2022. Association for Computational Linguistics. 1, 3, 8

[32] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 3, 5, 8

[33] Chan Hee Song, Jihyung Kil, Tai-Yu Pan, Brian M. Sadler, Wei-Lun Chao, and Yu Su. One step at a time: Long-horizon vision-and-language navigation with milestones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15482–15491, June 2022. 3, 6

[34] Alessandro Suglia, Qiaozi Gao, Jesse Thomason, Govind Thattai, and Gaurav Sukhatme. Embodied bert: A transformer model for embodied, language-guided visual task completion, 2021. 8

[35] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999. 1, 3

[36] Peter West, Chandra Bhagavatula, Jack Hessel, Jena Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. Symbolic knowledge distillation: from general language models to commonsense models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4602–4625, Seattle, United States, July 2022. Association for Computational Linguistics. 2

[37] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods*

in Natural Language Processing: System Demonstrations, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics. 6

[38] Yichi Zhang and Joyce Chai. Hierarchical task learning from language instructions with unified transformers and self-monitoring. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4202–4213, Online, Aug. 2021. Association for Computational Linguistics. 6

[39] Kai Zheng, KAI-QING Zhou, Jing Gu, Yue Fan, Jialu Wang, Zonglin Li, Xuehai He, and Xin Eric Wang. Jarvis: A neuro-symbolic commonsense reasoning framework for conversational embodied agents. *ArXiv*, abs/2208.13266, 2022. 8