R2C: Mapping Room to Chessboard to Unlock LLM As Low-Level Action Planner

Ziyi Bai[1,2], Hanxuan Li[1,2], Bin Fu[1,2], Chuyan Xiong[1,2], Ruiping Wang[1,2], Xilin Chen[1,2]
[1]Key Laboratory of AI Safety of Chinese Academy of Sciences (CAS),
Institute of Computing Technology, CAS, Beijing, 100190, China
[2]University of Chinese Academy of Sciences, Beijing, 100049, China
{ziyi.bai, hanxuan.li, bin.fu, chuyan.xiong}@vipl.ict.ac.cn, {wangruiping, xlchen}@ict.ac.cn

Abstract

*This paper explores using large language models (LLMs) as low-level action planners for embodied tasks. While LLMs excel as the robot's "brain" for high-level planning, they face challenges in directly controlling the "body" by generating precise low-level actions. This limitation arises from LLMs' strength in high-level conceptual understanding but their inability to handle spatial perception effectively, restricting their potential in embodied tasks. To address this, we bridge the gap by enabling LLMs to not only comprehend complex instructions but also produce actionable, low-level plans. We introduce Room to Chessboard (R2C), a novel semantic representation that maps environmental states onto a grid-based chessboard, empowering LLMs to generate specific low-level coordinates and guide the robot in a manner akin to playing a game of chess. To further enhance decision-making, we propose the Chain-of-Thought Decision (CoT-D) paradigm, which improves LLMs' interpretability and context-awareness in spatial reasoning. By jointly training LLMs for high-level task decomposition and low-level action generation, we create a unified "**brain-body**" system capable of handling complex, free-form instructions while producing precise low-level actions, allowing the robot to flexibly control its movements and adapt to varying tasks. We validate R2C using both fine-tuned open-source LLMs and GPT-4, demonstrating effectiveness on the challenging ALFRED benchmark. Results show that with our R2C framework, LLMs can effectively act as low-level planners, generalizing across diverse settings and open-vocabulary robotic tasks. The code and demonstrations are available at: https://vipl-vsu.github.io/Room2Chessboard.*

1. Introduction

The pursuit of general embodied agents focuses on developing robust systems capable of understanding natural language commands to meet diverse human requirements. Traditional robotic learning methods have succeeded in executing complex tasks but struggle to generalize to new environments or tasks due to their reliance on task-specific training and rigid planning. Recently, Large Language Models (LLMs) have emerged as promising embodied agents.

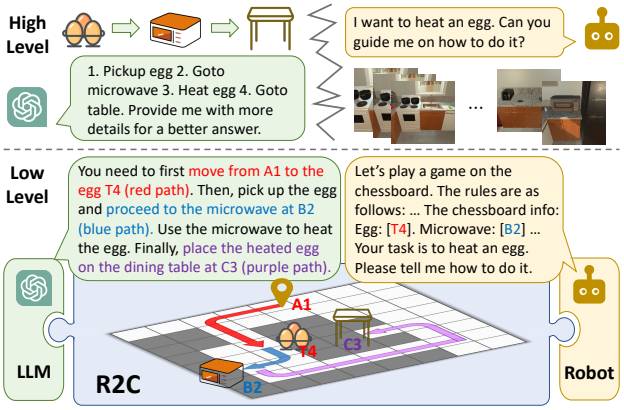

Figure 1. Comparison of LLM as a high-level versus low-level planner for the task of "heating eggs". High-level planner provides subgoals while low-level planner directly provides exact path.

Several pioneering works [8, 13, 14, 37] have explored LLMs as the "brain" of embodied systems, leveraging their strong generalization to manage diverse tasks effectively. However, most LLM-based agents [1, 13, 37] focus mainly on high-level task planning, decomposing tasks like "bring me an egg" into subgoals (e.g., "go to egg" → "take it back"). The execution of these subgoals is still delegated to low-level policy networks [5, 16, 18] or deterministic algorithms [7, 31]. While LLMs have extensive world knowledge, they lack spatial awareness of real environments, making it difficult to predict if these plans can be grounded to the physical world accurately. This often leads to frequent re-planning [1, 37], reducing efficiency and success rates, thereby limiting the potential of LLMs as embodied

agents in real-world applications.

The primary barrier [37, 44] to unlocking the full potential of LLMs lies in the ineffective communication between the LLM (the "brain") and the robot (the "body"). The robot struggles to convey the updated spatial information about environment, leaving the LLM without a basis for decision-making. As a result, the LLM can only generate high-level subgoal plans, while controlling the robot's movements relies on external navigation APIs. To overcome this challenge, a "common language" is needed — a platform that delivers sufficient environmental states, empowering the LLM to generate precise low-level action plans and directly control the robot.

We propose a novel Room to Chessboard (R2C) framework, which realizes a unified "brain-body" system that integrates high-level task understanding with direct low-level action decisions. As shown in Fig. 1, R2C utilizes a grid-based chessboard as an effective communication platform between the robot and the LLM. On one side, the robot can continuously map its observations onto the chessboard, which contains essential information while avoiding information overload. On the other side, the LLM is unlocked as a low-level action planner, directly guiding the robot "moves" on the chessboard to complete the task. Specifically, the LLM decomposes long-horizon tasks into subgoals, which are then addressed through precise low-level planning. At this stage, an Environment Filter maintains task-aware environmental states on a compact chessboard. Since the chessboard grid size is calibrated to match the robot's step length, the LLM can perform low-level planning on the chessboard by predicting the robot's next position. Additionally, the Chain-of-Thought Decision (CoT-D) paradigm is proposed to enhance LLM's spatial reasoning and decision-making capabilities.

To construct a chessboard representation that fully captures the complex environmental states while remaining manageable for LLMs, we introduce an Environment Filter to abstract the environment states. Initially, new observations are converted into a detailed 2D semantic map. However, such a high-dimensional representation can overwhelm LLMs. To mitigate this, we apply kernel filters to down-sample the map, then flatten it into a 2D chessboard, organized according to the priority of the current subgoal. The object occupancy on the chessboard is further abstracted into object coordinate sets. This representation strikes a balance between semantic richness and simplicity, effectively capturing critical information such as object semantics, object size and scene layout, while enabling efficient low-level action planning.

In R2C framework, LLM needs robust long-context understanding capabilities to comprehend the object coordinate sets and spatial reasoning abilities for low-level planning. Despite advancements in LLM capabilities, achieving this remains a huge challenge. To enhance LLM's low-level action planning abilities, we design a fine-tuning paradigm and formalize the CoT-D fine-tuning task. CoT-D comprises four subtasks: key information extraction, direction determination, target prediction, and selection analysis. LLM is required to link these subtasks together into a coherent logical chain [42]. These tasks can not only strengthen the long-context understanding of LLM but also enhance its spatial reasoning to generate more interpretable low-level plans.

We evaluate our R2C framework on the challenging ALFRED [34] benchmark, which features a diverse set of long-horizon tasks. We test both GPT and fine-tuned open-source LLMs using our novel CoT-D. R2C achieves state-of-the-art (SoTA) performance among LLM-based methods. Additionally, LLMs trained with CoT-D exhibit strong spatial reasoning and action planning capabilities. Our LLM-based low-level action planner also demonstrates impressive generalization across open-vocabulary tasks and real-world deployment, highlighting the versatility and robustness of the R2C framework in various application scenarios.

The main contributions of this paper are as follows:

- We propose Room to Chessboard (R2C), a unified "brain-body" system that enables LLMs to simultaneously understand free-form instructions and generate specific, environment-aware low-level actions.
- We develop the Chain-of-Thought Decision (CoT-D) fine-tuning mechanism, which enhances the spatial reasoning capabilities of LLMs, leading to a significant improvement of 11.37%.
- We achieve SoTA performance among LLM-based methods using limited robotic data and demonstrate strong generalization to unseen environments, including real-world scenarios.

2. Related Works

2.1. Task Planning In Robotics

Task planning in robotics [28, 34] involves generating a sequence of actions for robots to execute in the environments to achieve a specific goal. In real-world applications, the instructions are typically complex, resulting in long-horizon tasks that encompass a variety of embodied activities, such as navigation [2, 9] and object interaction [20]. Early approaches [27, 38] simply integrate visual and textual inputs to generate contextually appropriate action sequences. Besides, one widely used approach is reinforcement learning (RL) [11, 24, 39, 41]. Although the above methods have good performance on some specific tasks, they are trained end-to-end using expert trajectories and low-level instructions, resulting in poor generalization to unknown environments [19, 25, 37].

Therefore, some approaches [3, 15, 19, 23, 25] suggest explicitly breaking down the complex task into mul-

tiple functional modules. FILM [25] proposes a framework with four submodules including language processing, semantic mapping, semantic search and deterministic policy. ThinkBot [23] presents an additional object localizer based on multimodal transformers to predict the position of target objects. Though they provide more interpretable frameworks, each of their modules need to be fine-tuned with corresponding data, which make them struggle in adapting to novel robotic tasks.

2.2. Task planning with LLMs

To address the above generalization challenges, incorporating Large Language Models (LLMs) into robotic task planning presents a promising pathway [1, 10, 13, 14]. Early attempts [13] adopt LLMs to help with free-form human instruction following by decomposing the task into reasonable subgoals. However, due to LLM's inability to perceive the complex physical world, the generated goals often fail to execute in the environments [1].

Recent research has delved into integrating environmental states to ground the output of the LLM-based planner. LM-Nav [32] leverages a pre-trained Vision-Language Model (VLM) to generate captions of viewpoints and ground them with landmarks to enhance the executability of navigation plan. Additionally, structured representations such as object-centric scene graphs and topology graphs [6, 10, 29, 46] are commonly employed to improve the LLM's comprehension of relationships among objects and the overall room layout within the environments. Beyond directly translating visual feedback into inputs for LLMs, Say-Can [1] trained a vision-based value function to judge the affordance of the LLM's plan in the environments. Furthermore, some other works [21, 35] involve text-form Planning Domain Definition Language (PDDL) descriptions of scenes, which are easier for LLMs to handle.

Although the above works address some cases where LLM-generated plans are not executable, they all use LLMs solely as high-level planners. Most works still rely on low-level controllers to translate these high-level plans into executable action sequences. Some recent works [12, 33, 47] train end-to-end models to directly map robot observations to actions on large amounts of robotic trajectory data to achieve generalization. Unlike these approaches, we unlock LLMs to function as both high-level and low-level planners by introducing an interpretable textual representation of the environment.

3. Methods

We aim to unlock LLMs as low-level action planners that can directly guide robots to solve long-horizon tasks. We first introduce our newly designed Room to Chessboard framework in Sec. 3.1, and then illustrate the Chain-of-Thought Decision (CoT-D) fine-tuning task formulation in Sec. 3.2.

3.1. Room to Chessboard

We introduce the Room to Chessboard (R2C) framework (Fig. 2), which enables LLMs to perform both High-Level Planning (HLP) and Low-Level Planning (LLP) within a unified framework. First, the LLM conducts HLP, processing step-by-step instructions into a sequence of subgoals. Each subgoal is then handled at the LLP stage. At each timestep, an egocentric RGB-D image is processed by an Environment Filter to generate a chessboard abstraction. The updated chessboard is subsequently converted into object occupancy coordinates, which are fed to the LLM. Acting like a chess player, the LLM predicts the robot's next move on the chessboard, guiding it to navigate or interact with the environment.

3.1.1. High-Level Planning (HLP)

At the beginning of an episode, the LLM processes the natural language instruction L into a sequence of high-level subgoals, denoted as $\mathcal{G} = [G_1, G_2, ..., G_K]$. Each subgoal G_k is a tuple (`Action`, `Object`), where `Action` $\in A_H$ is a primitive action chosen from the set of navigation action (`GOTO`) or interaction actions (e.g., `PICKUP`, `PUT`). The `Object` refers to the semantic class of the interacted object (e.g., `SAFE`, `CD`). These subgoals are executed sequentially during the LLP stage.

3.1.2. Environment Filter

Since the environment is partially observed, the agent has never explored the entire environment and must construct a map based on its own observations online. At each timestep, the Environment Filter $F(o_t, G_k)$ takes the new observation o_t as input to filter out the goal-related environment information conditioned on the current subgoal G_k. This filtered information is then added to a dynamically updated semantic chessboard $\mathcal{B}$ to represent the current state of the environment. The computation of F involves two parts: semantic mapping and chessboard building.

Semantic mapping. We first build an online semantic map of the room, inspired by prior work [25]. At each timestep t, the agent receives an egocentric RGB-D observation $o_t = I_{rgb}, I_{dpt}$. An off-the-shelf instance segmentation model processes I_{rgb}, and, combined with I_{dpt}, the observation is converted into a point cloud, with each point labeled with predicted semantic categories. This 3D point cloud is then voxelized and flattened along the Z-axis to form the 2D semantic map $\mathcal{M}$. The resulting semantic map is represented as $(C + 2) \times M \times M$ binary grid, where C is the number of object categories, and two additional channels denote obstacles and explored areas in each cell. Here, M represents the map resolution.

Chessboard building. We further abstract the semantic map into a compact chessboard, *i.e.*, map to chessboard (*M2C*). The *M2C* function consists of two cascaded kernels.

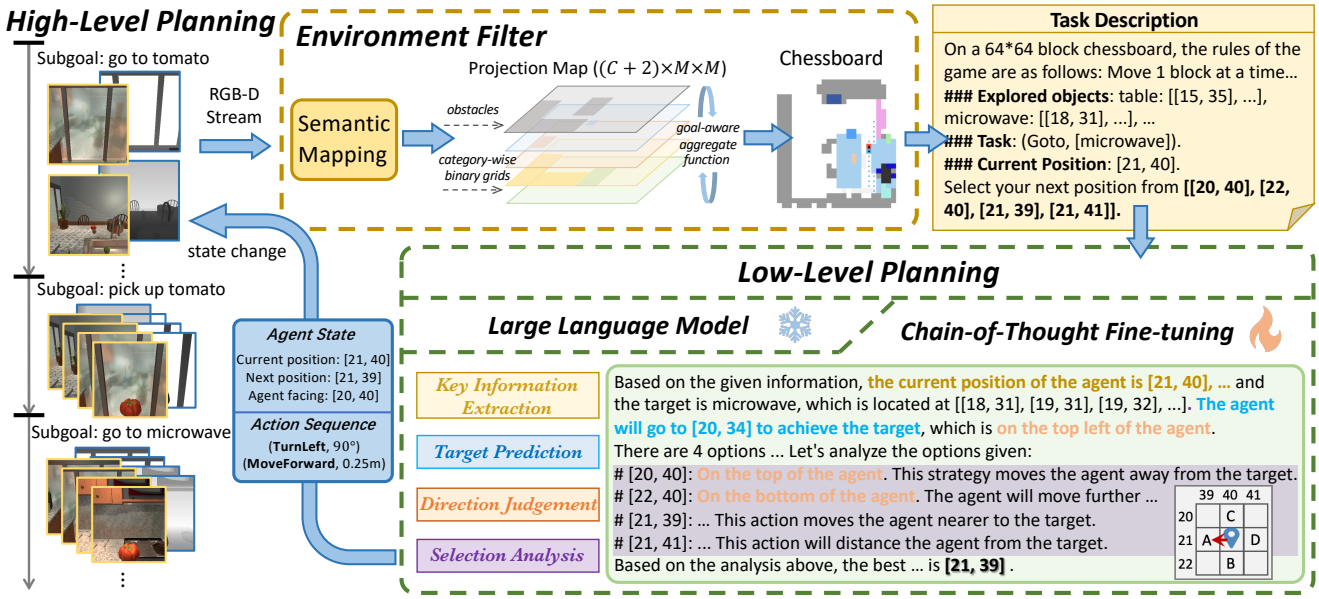

Figure 2. Overview of R2C framework: The LLM begins by generating high-level subgoals for the task. For each subgoal, the environment filter translates RGB-D observations into an updated grid-based chessboard representation of the environments. The LLM then uses chain-of-thought analysis to integrate this chessboard representation along with game rules, determining the next optimal position, which is finally converted into executable robot actions.

The first kernel is a dilation kernel $\mathcal{K}_{dila}$, designed to prevent collisions between the agent and obstacles. Similar to the dilation algorithm in image processing [30], it expands all occupied pixels of obstacles (including objects) outward by δ in the map. The second kernel is a max pooling kernel $\mathcal{K}_{pool}$, which performs max pooling on the map. Then, the map is down-sampled from size M to size W, where $W = \lceil \frac{M}{\omega} \rceil$ and ω is the grid size, calibrated to match the length of the agent's step.

Next, we aggregate this map with multiple object layers into a unified single-layer chessboard. To address the many-to-one projection problem, where overlapping objects might appear in the same grid, e.g., "apples on a table", we introduce a *Goal-aware Aggregate* function $\mathcal{A}(\cdot)$. This function prioritizes both the relevance to the current subgoal, i.e., the target object `Object` in current subgoal G_k, and the size of the object. The most relevant and smallest objects are placed at the top layer, and then we merge object layers from bottom to top to produce the final chessboard. The overall formulation of the *M2C* is:

$$\begin{aligned} M2C &= \mathcal{A} \circ \mathcal{K}_{dila} \circ \mathcal{K}_{pool} \\ \mathcal{B} &= M2C(\mathcal{M}, G_k). \end{aligned} \tag{1}$$

3.1.3. Low-Level Planning (LLP)

The chessboard filtered out from the Environment Filter provides compact yet sufficient environmental state information for LLM. We then simulate a "chess game" between the LLM and the robot to perform low-level planning. The LLP task is formulated as a single-step generation task for the LLM. As shown in Fig. 2, various state information is collected, including current subgoal G_k, chessboard state $\mathcal{U}$, action history $\mathcal{Q}$ and game rules R. Our prompt system $\mathcal{P}$ organizes such data and feeds them into the LLM to generate the next position prediction w. Note that LLP is invoked only for navigation subgoals, as interaction subgoals in ALFRED can be handled by predefined low-level actions once the agent reaches the visible range of the target object.

Chessboard state. To translate the chessboard state for the LLM, we convert it into textual object occupancy coordinate sets. The chessboard coordinate system originates from the upper left corner, with the X-axis pointing down and the Y-axis to the right. For each object on the chessboard, we gather a set of occupancy coordinates $\mathcal{U}_t^c = \{x_i, y_i\}$, where $c \in [1, C+2]$ and $x_i, y_i \in [1, W]$. The initial agent position w_0 is predefined in the benchmark.

Game rules. To assist the LLM in planning, we define key game rules R: 1) Basic inputs, including chessboard dimensions W, maximum steps T, and maximum errors E; 2) Action space, simplified to adjacent grids in four directions (up, down, left, right), i.e., the robot is only allowed to move 1 block at a time, due to the limited spatial reasoning ability of current LLMs; 3) Collision rules, where grids occupied by objects or obstacles are considered illegal moves.

Action history. We maintain a queue $\mathcal{Q}$ with a fixed length τ to store recent successful movement coordinates, providing the LLM with contextual information about previous actions. At each step t, the prompt system combines

all such information and feeds them to the LLM, which simulates a chess player to decide the next move. The LLP models a policy π, defined as:

$$u_t = \pi(\mathcal{P}(G_k, \mathcal{U}_t, R, \mathcal{Q}_t)), \tag{2}$$

where $u_t = (x_t, y_t)$ is the predicted next position on the chessboard. According to the defined action space, the available next positions are restricted to the adjacent grids in four directions $x_{t+1} \in [x_t - 1, x_t + 1]$ and $y_{t+1} \in [y_t - 1, y_t + 1]$, with $x_{t+1}, y_{t+1} \in [1, W]$.

Since the chessboard grid size is calibrated to match the robot's step length, the predicted adjacency target position u_t can be directly converted to a sequence of executable low-level actions $\mathbf{a} = \{a_i\}, a_i \in \mathbf{A}_{nav}$ in the real-world environments. For example, in the ALFRED benchmark, $\mathbf{A}_{nav}$ includes (`MoveForward`, $0.25m$), (`TurnLeft`, $90°$) and, (`TurnRight`, $90°$). Based on the predicted target position, the movement in any of the four directions can be converted into the corresponding action sequence (e.g., `TurnLeft` and `MoveForward` to move to the left adjacent grid). Additionally, if the output w_t is within the visible range of the target object `Object` in the current subgoal, the system moves on to the next subgoal. See more details of the R2C framework in the pseudocode provided in the supplementary materials.

3.2. Chain-of-Thought Fine-tuning Paradigm

To improve the decision-making capabilities of LLMs in our chess game, we introduce an interpretable fine-tuning paradigm with two core features: 1) simultaneous training of high-level task decomposition and low-level Chain of Thought Decision (CoT-D) tasks, enabling the integration of High-Level Planning (HLP) and Low-Level Planning (LLP) within a single LLM, and 2) structuring the CoT-D process into four distinct parts to enhance the LLM's comprehension of game rules and spatial reasoning.

3.2.1. Joint Training of HLP and LLP

Long-horizon planning often involves compositional tasks that require hierarchical planning capabilities. Therefore, rather than only exploiting the LLM as a high-level planner or training an end-to-end low-level action generation model, we jointly train the model on both HLP and LLP tasks simultaneously. Given the data imbalance between HLP and LLP, we apply data balancing techniques to ensure the model effectively learns skills at both levels.

3.2.2. Chain of Thought Decision (CoT-D)

LLM should thoroughly comprehend our chessboard coordinate system and predict the next position based on the semantic information of the chessboard and its inherent common sense. For the given prompt $\mathcal{P}$ with chessboard state $\mathcal{U}$, the LLM needs to generate answer sentence $\mathcal{S}$ (including next position w) with probabilistic language model p_{LM}. If LLM is asked to directly provide coordinates, this process can be formalized as:

$$p(\mathcal{S} \mid \mathcal{P}) = \prod_{i=1}^{|\mathcal{S}|} p_{LM}\left(s_i \mid \mathcal{P}, s_{<i}\right). \tag{3}$$

However, such a complex task is much more challenging than the task decomposition. This requires the LLMs have strong capabilities in long-text comprehension and spatial reasoning. However, the current models are not particularly skilled in these abilities. Consequently, we design the Chain of Thought Decision (CoT-D) tasks to strengthen LLM's rationale $\mathcal{R}$ in these aspects. The task consists of four subtasks, requiring LLMs to output the result of key information extraction $\mathcal{R}_E$, direction judgment $\mathcal{R}_D$, target prediction $\mathcal{R}_T$, and selection analysis $\mathcal{R}_S$, respectively. We link these sub-tasks sequentially in natural language to construct a coherent logical chain. The entire task can be represented as:

$$\begin{aligned} p(\mathcal{S} \mid \mathcal{P}) &= p(\mathcal{S} \mid \mathcal{P}, \mathcal{R}) \cdot p(\mathcal{R} \mid \mathcal{P}) \\ p(\mathcal{R} \mid \mathcal{P}) &= \prod_{r_i \in \{\mathcal{R}_E, \mathcal{R}_D, \mathcal{R}_T, \mathcal{R}_S\}} p_{LM}\left(r_i \mid \mathcal{P}, r_{<i}\right) \\ p(\mathcal{S} \mid \mathcal{P}, \mathcal{R}) &= \prod_{j=1}^{|\mathcal{S}|} p_{LM}\left(s_i \mid \mathcal{P}, \mathcal{R}, s_{<j}\right) \end{aligned} \tag{4}$$

The specific content of the four sub-tasks will be introduced below. Examples can be found in Fig. 2, with details available in the supplementary materials.

Key information extraction. Due to the intricate information contained within the chessboard, when textualized, it becomes a lengthy document with substantial information. Considering that LLM often encounters issues such as context loss when comprehending long texts, this task is designed to train LLM in processing task-related long texts and extracting key information. This task requires LLM to extract relevant information about the current coordinates and target objects based on the chessboard information inputs. The form used in data annotation is as follows: `The current position of the agent is [COORDS], and the target is [OBJECT NAME], which is located at [COORDS SET].`

Direction judgment. The conversion from a chessboard grid image to text is based on a two-dimensional Cartesian coordinate system, and all spatial relationship understanding relies on a good coordinate system understanding. Despite providing clear instructions to establish the coordinate system using prompts like `Establish a coordinate system with the top-left grid as (1,1)`, experiments have shown that the LLM still frequently misunderstands the setup. This could be attributed to the length of the text and the scarcity of spatial reasoning tasks incorporated during the pre-training of contemporary LLM. To ensure LLM's accurate comprehension

Method	Training Mode	Val Seen		Val Unseen		Δ SR ↑	Δ GC ↑
		SR ↑	GC ↑	SR ↑	GC ↑		
Specialists, only for ALFRED tasks							
E.T. [27]	from scratch	46.59	52.92	7.32	20.87	-39.27	-32.05
HiTUT [45]	from scratch	25.24	34.85	12.44	23.71	-12.80	-11.14
M-TRACK [36]	from scratch	26.70	33.21	17.29	28.98	-9.41	-4.23
FILM [25]	from scratch	24.63	37.20	20.10	32.45	-4.53	-4.75
LEBP [22]	from scratch	27.63	35.76	22.36	29.58	-5.27	-6.18
Generalists, based on LLMs							
SayCan [1]	few-shot	12.30	24.52	9.88	22.54	-2.42	-1.98
LLM-P (GPT) [37][1]	few-shot	16.45	30.11	15.36	29.88	-1.09	-0.23
R2C-GPT-4 (ours)	zero-shot	20.00	28.46	**24.00**	28.24	**+4.00**	-0.22
R2C-Llama-7B (ours)	fine-tune	20.83	29.60	18.99	29.69	-1.84	**+0.09**
R2C-Mistral-7B (ours)	fine-tune	**22.31**	**32.40**	22.35	**31.97**	+0.04	-0.43

Table 1. Main results on the ALFRED benchmark. SR and GC are short for success rate and goal-conditioned success rate. Δ SR and Δ GC represent the performance drops in generalizing from the seen to the unseen environments.

Task Type	Val Seen		Val Unseen	
	SR	GC	SR	GC
Overall	48.18	55.13	53.33	58.18
Examine	75.86	79.31	83.33	87.96
Pick & Place	69.57	69.57	63.33	63.33
Stack & Place	29.03	37.00	45.45	46.46
Clean & Place	63.89	73.87	38.89	50.45
Cool & Place	36.84	49.60	61.11	69.23
Heat & Place	23.53	41.38	28.57	41.13
Pick 2 & Place	33.33	54.41	37.50	53.70

Table 2. Performance of R2C on different tasks. The R2C model excels in "Examine" tasks, but faces challenges in complex "Heat & Place" tasks, where the "HeatObject" subgoal alone involves seven interactive steps, increasing the risk of error accumulation.

of the chessboard coordinate system, this task requires LLM to judge the direction between the target and the current coordinates. The format employed in data annotation is as follows: `the target is at..., which is on the [DIRECTION] of the agent.`

Target prediction. The locations where different categories of objects appear often have priors. For example, *sofas* are likely to appear opposite the *television*. Therefore, we aim to enable LLM to develop the capability of predicting target locations. This can minimize the ineffective or inefficient exploration process for the robot to find target objects, thereby improving the efficiency of the system in accomplishing embodied tasks. The expression utilized in data annotation is: `Based on the chessboard analysis, the target is likely located near [COORDS].`

Selection analysis. Based on the above comprehension, we require the LLM to further analyze all potential next positions according to the chessboard rules. During this analysis, the LLM evaluates each possible move, providing a rationale for each choice. The specific format used in the annotation is: `[COORDS]: This position is on the [DIRECTION] of the agent, [Reason].` The `[Reason]` component is generated by GPT-4 based on the agent's current state and the relationship between the chosen position and the correct orientation. For instance: `Objects in this direction have already been discovered; we should head to areas that have not been explored.`

These four sub-tasks comprehensively enhance the understanding of chessboard and spatial reasoning ability of the LLMs from different perspectives.

4. Evaluation

4.1. Experiment Settings

Benchmark. We evaluate our method on the challenging ALFRED [34] benchmark, which features long-horizon tasks (7 types in total) involving both navigation and interaction across 207 unique environments and 115 different object types. In our experiments, we adhere to the benchmark's settings, including low-level action space, maximum agent steps, and failure limits.

Method	Val Seen		Val Unseen	
	SR	GC	SR	GC
Base Model (R2C-Mistral-7B)	22.31	32.40	22.35	31.97
+ GT Seg.	37.92	45.83	35.24	43.88
+ GT Seg., GT Goal	**48.18**	**55.13**	**53.33**	**58.18**
+ GT Seg., GT Goal, - SA	45.97	51.75	47.45	53.03
+ GT Seg., GT Goal, - CoT	41.22	49.35	41.96	47.88

Table 3. Ablation of R2C. "GT Seg." represents the model using ground-truth segmentation. "GT Goal" represents using ground-truth subgoals. "- SA" is the model without Selection Analysis part of CoT. "- CoT" is the model without all CoT tasks.

Chessboard settings. The physical room size D is set to $16m$, which is the approximate maximum room size, and the grid size ω corresponds to the agent's step length, $0.25m$. Thus, the size of the semantic map and the chessboard is $M = 320$ and $W = 64$ separately. However, to represent more complex environments, one can choose a more fine-grained chessboard. We demonstrate this through a simple experiment in the supplementary materials.

Model settings. For the zero-shot R2C, we use the public GPT-4-turbo API [26] without any in-context examples. Considering the testing cost, we randomly sample a subset of size 100 covering all 7 types of tasks strictly according to the task distribution of ALFRED. We collected a total of 264,915 data samples for fine-tuning, including 70,000 samples for the Task Decomposition tasks and 194,915 samples for the CoT-D tasks. All data collection was based solely on the training set of ALFRED. During training,

[1]For a fair comparison, we test the performance of the GPT-4 version of LLM-P on the same selected subset as ours, achieving an SR of 16.21% on seen scenes. Since this work solely employs LLMs for task decomposition, the performance is only slightly different when compared to using GPT-3.

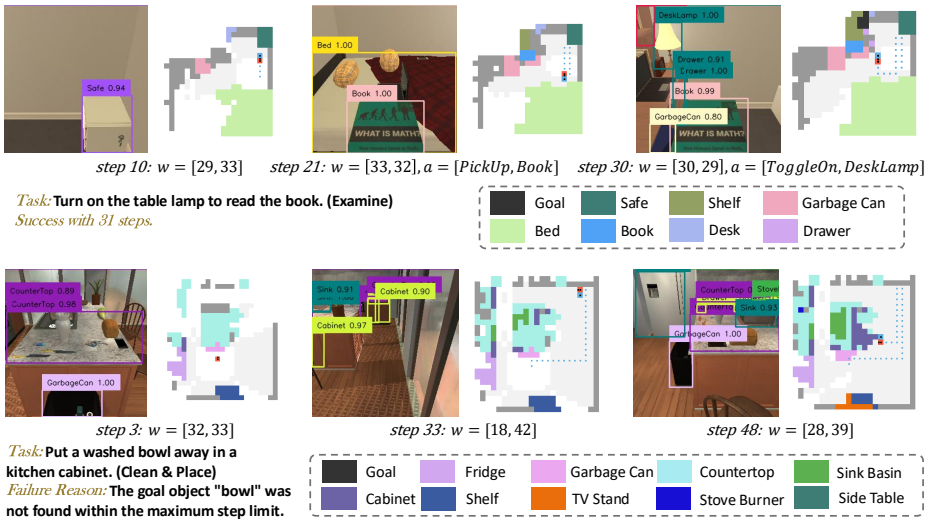

Figure 3. Case study of R2C with Mistral model: The model completes an "Examine" task in 31 steps (top) but fails to locate the small target object, a bowl, during a "Clean & Place" task in a more complex environment (bottom).

we conduct full-parameter fine-tuning on both the Mistral-7B-Instruct-v0.2 model [17] and the Llama-7B-Chat model [40] using all the data. Both models were trained for only one epoch. We conduct all fine-tuning experiments using 4 NVIDIA H100 GPUs, and all evaluations are performed on 4 NVIDIA A40 GPUs.

4.2. Main Results

Tab. 1 presents the evaluation results on the validation set of ALFRED. We compare our R2C framework with both traditional robotic learning methods (specialists) and LLM-based approaches (generalists). R2C achieves state-of-the-art performance among LLM-based methods, with the Mistral-7B excelling in seen environments and GPT-4 in unseen environments.

Compared to SayCan [1] and LLM-P [37], which use LLMs only for high-level planning, R2C integrates high-level and low-level planning, offering a more efficient, end-to-end solution. LLM-P requires 100 instruction-plan pairs for training, while R2C operates in a zero-shot setting without examples. Our CoT-D paradigm enables GPT to analyze the chessboard state comprehensively, improving decision interpretability and efficiency.

R2C fine-tuned on open-sourced LLMs achieves competitive results, with 2.31% improvement in the seen split and 1.65% drop in the unseen split compared to R2C-GPT-4. This highlights GPT-4's superior generalization to unseen environments. However, with our carefully designed fine-tuning tasks, R2C implemented on much smaller models like LLaMA and Mistral, can still deliver performance comparable to GPT-4. This demonstrates the effectiveness and efficiency of our fine-tuning approach.

Finally, R2C fine-tuned on collected data performs comparably to specialist models like FILM [25] and LEBP [22].

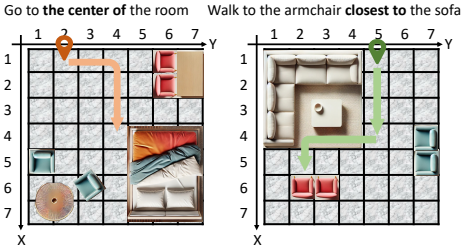

Figure 4. Cases of R2C-GPT-4 on open-vocabulary tasks.

Although these specialized models excel in seen scenes, they struggle in unseen environments due to overfitting. In contrast, R2C and other LLM-based methods demonstrate stronger generalization across both seen and unseen environments.

4.3. Ablation Study

We perform an ablation study to analyze the impact of different R2C modules. As shown in Tab. 3, using ground truth segmentation (GT Seg.) significantly improves performance, highlighting segmentation as a bottleneck, especially in simulation environments. Using ground truth subgoals (GT Goal) also boosts SR, revealing the impact of ambiguity in natural language instructions. Annotator confusion between categories like *desk lamps* and *floor lamps* can hinder task decomposition, though LLMs' strong generalization helps mitigate this.

Removing the CoT-D framework, i.e., forcing the model to directly output position coordinates without rationale, results in a significant performance drop, especially in unseen scenes. This highlights the importance of fine-tuning with CoT-D for better spatial reasoning. Ablating individual subtasks is challenging due to their interdependence. Here, we ablate removing the Selection Analysis part (- SA), where decisions are made without analyzing options and it causes a noticeable performance decline.

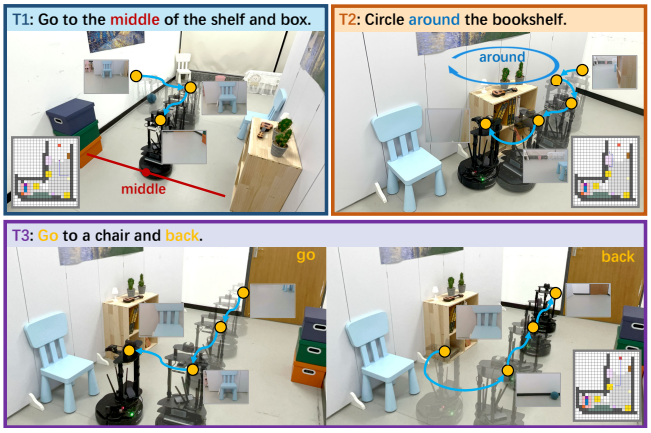

Figure 5. Visual results of R2C in real-world scenarios. Three tasks are tested in a real-world environment, with the corresponding instructions, third-person trajectories, first-person images, and chessboard visualizations shown for each task.

Task	Specific Obj.	Specific Loc.	Nearest Corner	Center Between	Overall
GPT-4 (90°)	66.7	73.3	53.3	80.0	68.3
GPT-4 (45°)	73.3	73.3	73.3	60.0	70.0
GPT-4o (90°)	40.0	53.3	33.3	46.7	43.3

Table 4. SR (%) of R2C on open-vocabulary tasks.

4.4. Evaluation Across Tasks Types

We analyze R2C's performance across task types using GT Segmentation and GT Goal settings, with task-specific results shown in Tab. 2. R2C shows SR drops on more complex long-horizon tasks, such as "Heat & Place," with average GT step lengths of 63, compared to shorter ones (e.g., Examine: 37). We provide a visualization of two case studies of R2C completing different tasks ("Examine" and "Clean & Place") in Fig. 3. More failure case analyses are included in the supplementary materials.

4.5. Exploration on Open-vocabulary Task

Real-world tasks are more diverse and flexible than the predefined, limited tasks in ALFRED. To assess R2C's ability to handle open-vocabulary tasks, we conduct a mini experiment with 60 test trials generated randomly by GPT-4 across 3 room sizes (7x7, 5x9, 5x11), five room layouts, and four task types. 1) **Specific Obj.**: Move to a specific object (e.g., nearest chair). 2) **Specific Loc.**: Move to a specific location (e.g., center or corner). 3) **Nearest Corner**: Move to the nearest corner. 4) **Center Between**: Move to the center between two objects.

Traditional closed-domain approaches struggle with these tasks, as they are limited to defined tasks in the training set. Meanwhile, high-level LLM planners face challenges in translating them into object-navigation plans. We test R2C with various settings: GPT-4 (90°), GPT-4 (45°) which allow 45-degree rotation, and GPT-4o (90°), which uses visualized chessboards. Results are summarized in Tab. 4, and two task examples are visualized in Fig. 4.

Method	SAVN [43]	S2P [4]	**R2C**
SR (%)	40.86	46.16	**52.25**
SPL (%)	16.15	28.01	**29.56**

Table 5. Performance of R2C on object navigation task.

The results demonstrate the potential of R2C in handling open-vocabulary tasks for real-world applications. GPT-4 (45°) outperforms the 90°-agent, thanks to more efficient diagonal movement, highlighting the expandability of R2C with different action spaces. Both GPT-4 (90°) and GPT-4 (45°) outperform GPT-4o, as feeding the chessboard directly to the VLM leads to poor spatial grounding and inaccurate path predictions. More results are available in the supplementary materials.

4.6. Application in Real-world Scenarios

To evaluate the real-world applicability of the R2C framework, we design three open-vocabulary tasks to test its performance using the LoCoBot robot in a set-up real-world experiment room. We evaluate R2C-GPT-4 in zero-shot setting. As shown in Fig. 5, the R2C framework, by enabling LLM to generate low-level actions, allows successful task completion in a totally different environment, showcasing its effectiveness and versatility in real-world scenarios. Detailed implementation information can be found in the supplementary materials.

4.7. Evaluation on Navigation Task

To further evaluate the model's generalization capability across tasks, we extend R2C-Mistral-7B model trained on ALFRED to the AI2THOR ObjectNav task [43] without additional fine-tuning. Results in Tab. 5 show that R2C significantly outperforms S2P [4], another LLM-based planner.

5. Conclusion

We introduce the Room to Chessboard (R2C) framework, which maps the complex room into a chessboard as a communication platform between the LLM and the robot, unlocking the LLM as low-level planner to directly guide the robot to adaptively finish the embodied tasks. To address the spatial reasoning tasks on chessboard, we design a CoT-D fine-tuning paradigm that enhances LLM's ability to make interpretable low-level decisions. Experiments show that R2C outperforms existing LLM-based high-level planners, even in zero-shot settings, and allows 7B models to surpass GPT-4. Furthermore, R2C enables LLMs to tackle open-vocabulary tasks where API-based frameworks fall short. However, R2C still faces challenges, such as handling very large scenes. Future work will focus on optimizing R2C for larger environments and exploring its potential for various open-vocabulary tasks.

Acknowledgement This work is partially supported by National Key R&D Program of China No. 2021ZD0111901, and Natural Science Foundation of China under contracts Nos. 62495082, U21B2025. Ziyi Bai, Hanxuan Li, and Bin Fu contributed equally to this work. Ziyi Bai led the implementation of the overall framework and the evaluation of the ALFRED benchmark and open-vocabulary tasks. Hanxuan Li proposed the R2C and CoT-D fine-tuning methods, validated their feasibility, and took the lead in the design, training, and acceleration of the LLM-based module. Bin Fu focused on the implementation of semantic mapping and chessboard construction, led the object navigation evaluation, and also contributed to the ALFRED evaluation. Bin Fu and Hanxuan Li jointly led the system deployment in real-world environments.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 1, 3, 6, 7

[2] Peter Anderson, Qi Wu, Damien Teney, J Bruce, M Johnson, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *arXiv preprint arXiv:1711.07280*, 2018. 2

[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

[4] Davide Buoso, Luke Robinson, Giuseppe Averta, Philip Torr, Tim Franzmeyer, and Daniele De Martini. Select2plan: Training-free icl-based planning through vqa and memory retrieval. *arXiv preprint arXiv:2411.04006*, 2024. 8

[5] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems*, 33:4247–4258, 2020. 1

[6] Jiaqi Chen, Bingqian Lin, Ran Xu, Zhenhua Chai, Xiaodan Liang, and Kwan-Yee Wong. Mapgpt: Map-guided prompting with adaptive path planning for vision-and-language navigation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9796–9810, 2024. 3

[7] Edsger W Dijkstra. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, pages 287–290. 2022. 1

[8] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023. 1

[9] Jing Gu, Eliana Stefani, Qi Wu, Jesse Thomason, and Xin Eric Wang. Vision-and-language navigation: A survey of tasks, methods, and future directions. *arXiv preprint arXiv:2203.12667*, 2022. 2

[10] Qiao Gu, Alihusein Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, et al. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. *arXiv preprint arXiv:2309.16650*, 2023. 3

[11] Yicong Hong, Cristian Rodriguez, Yuankai Qi, Qi Wu, and Stephen Gould. Sub-instruction aware vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1045–1054, 2020. 2

[12] Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. An embodied generalist agent in 3d world. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024. 3

[13] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR, 2022. 1, 3

[14] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022. 1, 3

[15] Yuki Inoue and Hiroki Ohashi. Prompter: Utilizing large language model prompting for a data efficient embodied instruction following. *arXiv preprint arXiv:2211.03267*, 2022. 2

[16] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022. 1

[17] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023. 7

[18] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021. 1

[19] Byeonghwi Kim, Jinyeon Kim, Yuyeong Kim, Cheolhong Min, and Jonghyun Choi. Context-aware planning and environment-aware memory for instruction following embodied agents. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10936–10946, 2023. 2

[20] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *International Journal of Robotics Research*, 37(4-5): 421–436, 2018. 2

[21] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023. 3

[22] Haoyu Liu, Yang Liu, Hongkai He, and Hangfang Yang. Lebp–language expectation & binding policy: A two-stream framework for embodied vision-and-language interaction task learning agents. *arXiv preprint arXiv:2203.04637*, 2022. 6, 7

[23] Guanxing Lu, Ziwei Wang, Changliu Liu, Jiwen Lu, and Yansong Tang. Thinkbot: Embodied instruction following with thought chain reasoning. *arXiv preprint arXiv:2312.07062*, 2023. 2, 3

[24] 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 16*, pages 259–274. Springer, 2020. 2

[25] So Yeon Min, Devendra Singh Chaplot, Pradeep Ravikumar, Yonatan Bisk, and Ruslan Salakhutdinov. Film: Following instructions in language with modular methods. *arXiv preprint arXiv:2110.07342*, 2021. 2, 3, 6, 7

[26] OpenAI. Gpt-4: Generative pre-trained transformer 4. *arXiv*, arXiv:2303.08774, 2023. 6

[27] Alexander Pashevich, Cordelia Schmid, and Chen Sun. Episodic transformer for vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15942–15952, 2021. 2, 6

[28] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8494–8502, 2018. 2

[29] Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning. In *7th Annual Conference on Robot Learning*, 2023. 3

[30] Jean Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982. 4

[31] James A Sethian. Fast marching methods. *SIAM review*, 41 (2):199–235, 1999. 1

[32] Dhruv Shah, Błażej Osiński, Sergey Levine, et al. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Conference on robot learning*, pages 492–504. PMLR, 2023. 3

[33] Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. Vint: A foundation model for visual navigation. *arXiv preprint arXiv:2306.14846*, 2023. 3

[34] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020. 2, 6

[35] Tom Silver, Varun Hariprasad, Reece S Shuttleworth, Nishanth Kumar, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddl planning with pretrained large language models. In *NeurIPS 2022 foundation models for decision making workshop*, 2022. 3

[36] 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*, pages 15482–15491, 2022. 6

[37] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009, 2023. 1, 2, 6, 7

[38] Alessandro Suglia, Qiaozi Gao, Jesse Thomason, Govind Thattai, and Gaurav Sukhatme. Embodied bert: A transformer model for embodied, language-guided visual task completion. *arXiv preprint arXiv:2108.04927*, 2021. 2

[39] Hao Tan and Mohit Bansal. Learning to navigate from disorganized instructions by self-supervised imitation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1208–1218, 2019. 2

[40] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. 7

[41] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6629–6638, 2019. 2

[42] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. 2

[43] Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6750–6759, 2019. 8

[44] Jingkang Yang, Yuhao Dong, Shuai Liu, Bo Li, Ziyue Wang, Chencheng Jiang, Haoran Tan, Jiamu Kang, Yuanhan Zhang, Kaiyang Zhou, et al. Octopus: Embodied vision-language programmer from environmental feedback. *arXiv preprint arXiv:2310.08588*, 2023. 2

[45] Yichi Zhang and Joyce Chai. Hierarchical task learning from language instructions with unified transformers and self-monitoring. *arXiv preprint arXiv:2106.03427*, 2021. 6

[46] Gengze Zhou, Yicong Hong, and Qi Wu. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7641–7649, 2024. 3

[47] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023. 3