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

Supplementary Material

The supplemental material is organized as follows:

- Experimental details of real-world deployment and open-vocabulary tasks in Sec. 1.
- Additional failure case studies in Sec. 2.
- Additional experiments on fine-grained chessboards in Sec. 3.
- Model implementation details in Sec. 4.
- Model inference visualizations in Sec. 5.

Task/Room Size	7×7	5×9	5×11	Overall
Specific Object	60.0	60.0	80.0	66.7
Specific Location	100.0	60.0	60.0	73.3
Nearest Corner	40.0	40.0	80.0	53.3
Center Between	100.0	80.0	60.0	80.0
Overall	75.0	60.0	70.0	68.3

Table 1. Performance of R2C-GPT-4 on open-vocabulary tasks.

1. Experimental Details

1.1. Real-World Deployment

We deploy the R2C-GPT-4 model on LoCoBot, a popular open-source mobile robot based on Kobuki and equipped with a RealSense D435i camera. We build a $6m \times 6m$ office scene with various objects in the laboratory. Due to the significant differences between the camera parameters of the robot and those of the AI2THOR simulator, directly using the RGB-D images captured by the robot's camera for 3D point cloud generation results in significant loss of environmental information, which poses challenges for low-level planning of the R2C models. Therefore, following [7], we use handheld scanning with a Microsoft Azure Kinect DK camera to obtain relatively complete 3D point clouds for downstream tasks. The results are shown in Fig. 1. We then apply an off-the-shelf 3D semantic segmentation model, SAI3D [8], to process semantic mapping. The chessboard construction process remains the same as in the main experiments. Finally, we obtain a 22×23 chessboard with a grid size of $0.25m$.

We choose three types of open-vocabulary tasks for evaluation, including: 1) Go to the middle of the shelf and box, 2) Circle around the bookshelf, and 3) Go to a chair and back. The initial position of the robot is random. The robot's step length, action space are all set to be the same as the settings in the main experiments. We show third-view trajectories and the corresponding chessboards in Fig. 2. The results indicate that our R2C-GPT-4 model demonstrates strong generalization not only to novel tasks but also to unseen scenes, fully showcasing the potential of applying our methods to real-world scenarios.

1.2. Open-Vocabulary Tasks

Tasks in benchmarks like ALFRED are typically limited to specific objectives, such as "find a specific object" or "interact with a specific object", which can be easily grounded without spatial reasoning. In contrast, real-world scenarios involve high-level spatial reasoning, with human instructions referencing abstract concepts like "the center of a room" or "the nearest corner". To address this, we design an open-vocabulary mini benchmark that challenge embodied planners by incorporating such abstract spatial concepts.

1.2.1. Benchmark Construction

We utilize GPT-4 to construct a mini evaluation benchmark. First, we prompt GPT-4 to randomly generate rooms of varying sizes (7×7, 5×9, and 5×11) with diverse layouts. Each room contains objects of different types: table, chair, armchair, sofa, bed, and desk. The sizes of the objects vary, and the layouts are designed to resemble realistic room structures with adequate walking space. For each room size, five unique layouts are generated, resulting in a total of 15 distinct rooms. All generated rooms are shown in Fig. 3. Based on these rooms, we then prompt GPT-4 to generate four types of tasks for each room. These tasks simulate common embodied tasks encountered in daily life while requiring both generalized spatial concepts understanding ability and various spatial reasoning ability combing specific environmental states.

1) Find Specific Object (**Specific Object**): Tasks involve identifying a particular object among multiple identical ones. These tasks evaluate the model's spatial grounding ability.

2) Find Specific Location (**Specific Location**): Tasks include abstract spatial concepts such as "center" and "corner", designed to assess the model's global spatial understanding ability.

3) Go to the Nearest Corner (**Nearest Corner**): Tasks require the model to identify the nearest corner based on the distance estimation.

4) Go to the Center Between Two Objects (**Center Between**): Tasks require calculating the midpoint between two objects, which test precise arithmetic spatial reasoning.

To ensure accuracy, we manually evaluate task completion. The game rules are similar to those of the main exper-

Figure 1. 3D point cloud of the room.

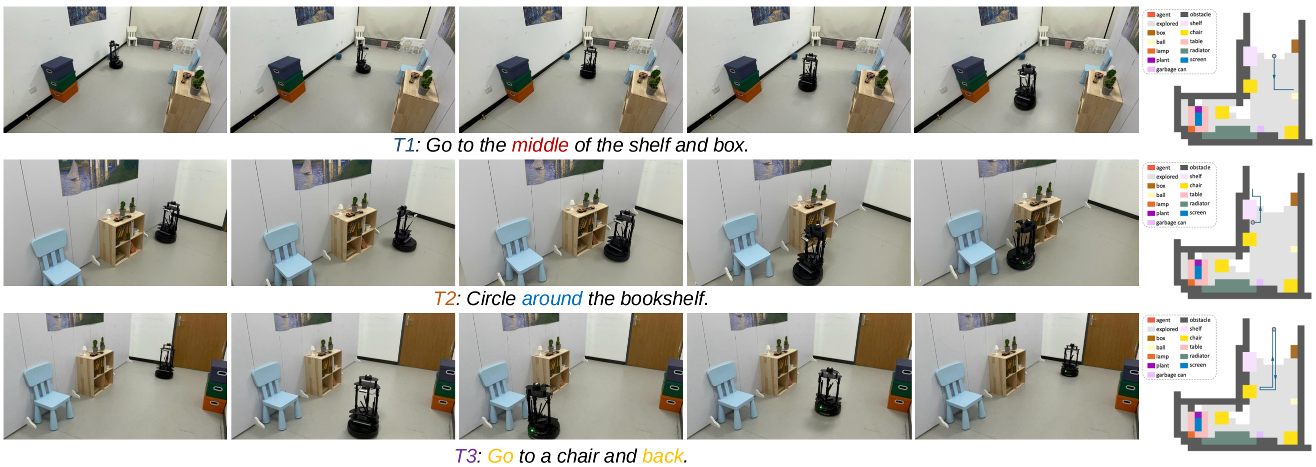


Figure 2. Third-view trajectories and chessboard visualizations of the real-world experiments.

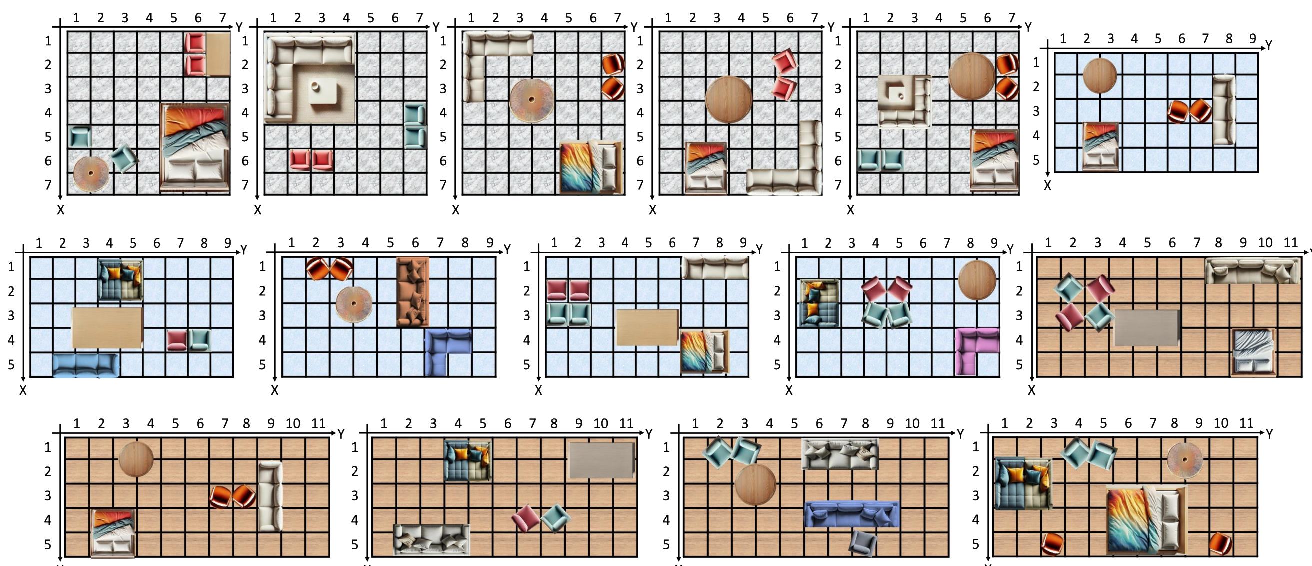


Figure 3. Generated rooms in three sizes (7×7, 5×9, and 5×11), each with five different layouts.

iments. However, due to the smaller chessboard size, tasks are considered successful if they are completed within 50 steps and with no more than two collisions.

1.2.2. Detailed Result Analysis

Sec. 1.2.1 presents the detailed experimental results for all tasks of GPT-4 (90°), measured by task success rate across different room layouts. As shown in Sec. 1.2.1, our R2C-based LLM can generalize well across different tasks and rooms. Overall, it performs relatively strong spatial reasoning and path planning, achieving 68.3% success rate. The model performs best in smaller, square rooms (7×7), achieving a success rate of 75%, while rooms with other aspect ratios exhibit similar performance levels. This suggests that the LLM has the potential to handle various room layouts effectively. Tasks involving distance estimation, like "Nearest Corner", have the lowest success rates, indicating difficulties in abstract spatial reasoning. On the other hand, the "Center Between" task demonstrates relatively high performance, showcasing the model's ability to calculate precise distances between two objects.

On the contrary, the above tasks pose significant challenges for traditional models [4, 5] or current high-level LLM-based planners [1, 6], as these tasks require a combination of both high-level semantic understanding and low-level spatial reasoning and action planning abilities. On one hand, only LLMs are capable of generalizing to abstract spatial semantics, such as "the center of a room", "the nearest corner", or "the center between objects". Traditional models [4, 5] can only obtain policies of these tasks after having observed corresponding trajectories. On the other hand, while current LLM-based planners [1, 6] understand these abstract spatial semantics, they lack access to specific environmental states, such as the precise spatial states of objects in the environment. They rely solely on limited object descriptions from caption models, which are insufficient for directly guiding a low-level planner to reach a target location.

Fig. 6 illustrates a case study of GPT-4's open vocabulary experiment, where the task "Move to the nearest corner in the room" is completed successfully in 5 steps. GPT-4 is guided to first analyze the state and generate a Chain-of-Thought Decision (CoT-D), followed by determining the optimal move. Finally, GPT-4 independently decides whether to terminate the task. We highlight the different sub-tasks within the CoT-D using distinct colors in GPT-4's responses. As shown in its responses, GPT-4 accurately interprets the semantics of "corner" and correctly identifies the nearest one. During the "Selection Analysis", it evaluates each possible move on the chessboard, including the **occupation**, *e.g.*, (2,5) is occupied by the *table*, making the move impossible; and the **distance**, *e.g.*, the shortest path to a corner. Then it ultimately selects the optimal position for the next step.

2. Failure Case Studies

We further analyze the failure cases of R2C by categorizing them into segmentation, language processing, object exploration, interaction. It explores how these issues affect the model's performance. By examining the impact of each failure mode, the study identifies key bottlenecks, such as segmentation and language processing. Below is a summary of the key failure modes identified in the R2C-Mistral-7B on ALFRED seen and unseen validation sets.

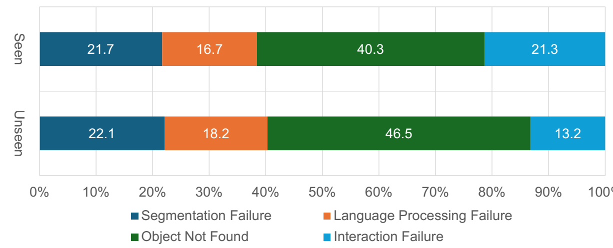


Figure 4. Failure Types. The percentage of different failure types of R2C on the validation set in ALFRED.

- **Segmentation Failure**: The system struggles to correctly segment objects, making it hard to identify or interact with them. This is a key bottleneck in visual perception, particularly due to the domain gap between the simulator and real-world scenes.
- **Language Processing Failure (High-Level Planning Failure)**: Failures occur when the system misinterprets language instructions, often because of the ambiguity in natural language. For example, it may confuse object categories like *desk lamps* and *floor lamps*, causing cascading errors.
- **Object Not Found**: The system often cannot locate small or poorly visible objects, especially from certain viewpoints, which challenges low-level planning and is worsened by limited segmentation and field of view.
- **Interaction Failure**: This happens when the system cannot interact with objects, either due to them being out of view, too far away, or poorly segmented. It can also occur when the target object is inside a closed receptacle, like a drawer or a cabinet.

The proportion of different types of errors is shown in Fig. 4. Among them, "Segmentation Failure" and "Object Not Found" are the two main error types. "Object Not Found" and "Interaction Failure" are partly due to the limitations of the simulator but also stem from low-level planning issues within R2C. Due to the fact that ALFRED is a long-horizon planning task, the accumulation of these different types of errors will pose significant challenges to the model, resulting in a low success rate of the current model.

3. Additional Experiments: More Fine-grained Chessboard

We further conduct experiments with the grid's size set to be $0.1m$. Then the chessboard becomes more fine-grained (160×160). We use the R2C-Mistral-7B model trained on 64×64 chessboard data to infer with such more fine-grained chessboard (zero-shot). To simulate the cluttered setting, we select a kitchen with messy objects. The visualized chessboards and agent's trajactories are shown in Fig. 5. We find that the model can easily generalize to this more fine-grained chessboard. Our model successfully completes the task with 111 steps. Note that the target object, an *apple*, is relatively small, but its position is accurately represented on the chessboard.

4. Implementation Details

We provide additional model implementation details, including model settings and the pseudocode of the R2C framework. We also compare the computational cost of our models with two representative methods.

4.1. Model Settings

In all our GPT-4 experiments, we use the official API without any in-context examples. The specific model we used is gpt-4-turbo-2024-04-09. The hyperparameters for the Chain-of-Thought fine-tuning of Mistral-7B are detailed in Tab. 2. We use Mistral-7B-Instruct-v0.2 as the base model and perform full parameter fine-tuning. The learning rate is set to 2e-5. During training, the batch size per device is 32, while for inference, it is reduced to 1.

Hyper-parameter	Values
base model	Mistral-7B-Instruct-v0.2
fine-tune mode	full fine-tune
deep speed	ds_z3
cutoff length	3000
preprocessing num workers	16
per device train batch size	32
per device eval batch size	1
gradient accumulation steps	1
lr scheduler type	cosine
logging steps	10
warmup steps	375
save steps	500
eval steps	500
evaluation strategy	steps
learning rate	2e-5
num train epochs	1.0
max samples	1000000
val set ratio	0.01
ddp timeout	1800000

Table 2. Hyper-parameters in Chain-of-Thought fine-tuning of Mistral-7B.

4.2. Pseudocode of R2C Framework

The pseudocode for the complete R2C framework is provided in Algorithm 1.

Algorithm 1 Room to Chessboard Planning Framework

Input: Instruction L, Game rules R, Initial chessboard information $\mathcal{U}_0$,
    Initial position u_0
Output: Sequence of coordinates $[u_t = (x_t, y_t)]$ and action sequence $\mathbf{a}$
Initialize current position (x_0, y_0) based on u_0
Initialize history steps $\mathcal{Q}_0 = []$
$t \leftarrow 0$, $k \leftarrow 1$, $errors \leftarrow 0$
High-Level Planning (HLP):
Parse instruction L to generate sequence of sub-goals $\mathbf{G} = [G_1, G_2, ..., G_K]$, where $G_k = (Action_k, Object_k)$
while $t < T$ **and** $k \leq K$ **and** $errors < E$ **do**
    Get current subgoal $G_k = (Action_k, Object_k)$
    Low-Level Planning (LLP):
    Get new observations: $o_t = \{I_{rgb}, I_{dpt}\}$
    Update chessboard: $\mathcal{U}_t = F(o_t, G_k)$
    if $Action_k$ is navigation **then**
        $u_t = \pi(P(G_k, \mathcal{U}_t, R, \mathcal{Q}_t))$
        Convert $u_t = (x_t, y_t)$ to executable actions
        $\mathbf{a} = \{a_1, a_2, ..., a_n\}$ in $\mathbf{A}_{nav}$
        Execute actions $\mathbf{a}$
        $t \leftarrow t + n$
        if Move to (x_t, y_t) is successful **then**
            $\mathcal{Q}$.append((x_t, y_t))
        else
            $errors \leftarrow errors + 1$
        end if
        if $Object_k$ is visible in $\mathcal{U}_t$ **then**
            $k \leftarrow k + 1$ ▷ Move to next subgoal if visible
        end if
    else if $Action_k$ is interaction **then**
        Execute predefined low-level interaction actions
        $\mathbf{a} = \{a_1, a_2, ..., a_n\}$ in $\mathbf{A}_{int}$
        $t \leftarrow t + n$
        $k \leftarrow k + 1$ ▷ Directly move to next subgoal
    end if
end while

4.3. Comparison of the Computational Cost

We provide a comparison of the computational cost among the R2C models, the traditional baseline FILM [4], and the LLM-based baseline, LLM-P [6]. Note that inference speed refers to the time required for each model call, while inference frequency refers to the average number of times the model is called per episode.

As shown in Tab. 3, though the textualized chessboard is relatively long, we only keep the task-related object coordinates and remove the unexplored coordinates. The prompt length of R2C in each turn is about 1k tokens, which is similar to the LLM-P. The proposed CoT-D mainly affects the

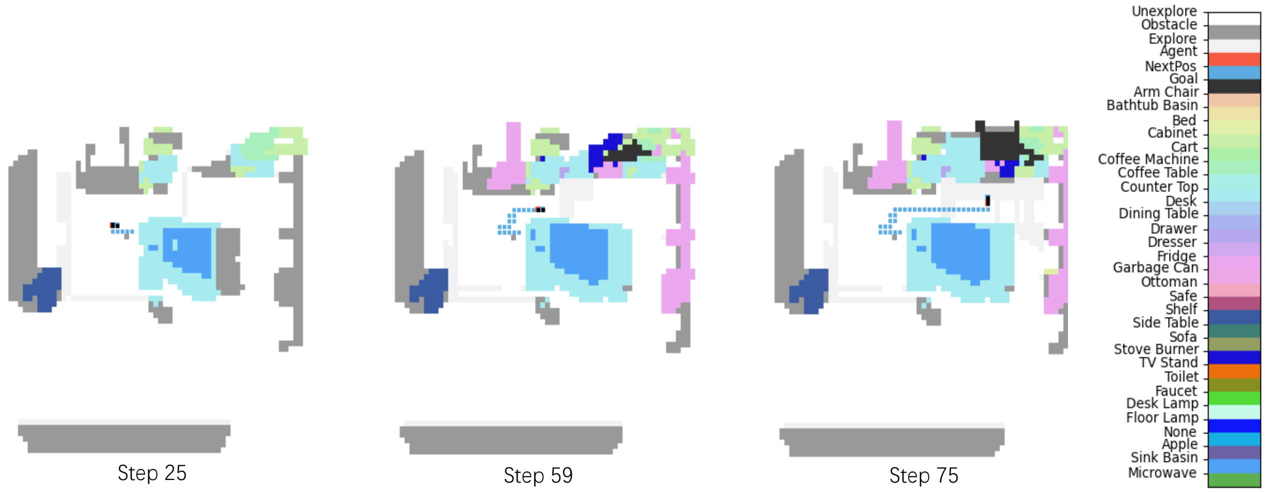


Figure 5. Cluttered setting demo. The chessboard size is 160 × 160. The task is "Get an apple from the sink and heat it up in the microwave". The blue dashed line represents the trajectory.

Model	FILM [4]	LLM-P (GPT) [6]	R2C-GPT-4	R2C-Mistral-7B
Input Prompt Length (tokens)	/	∼1k	∼1k	∼1k
Output Seq. Length (tokens)	/	∼20	∼1.5k	∼1.5k
Training Data / Examples (ep)	21k	100	0	21k
Training Time (GPU*Day)	1.5[3090]	/	/	4[A100]
Inference Speed (s/times)	357[3090]	5	12	0.53 (100 processes) / 3.79 (1 process)
Inference Freq. (times per ep)	1	<10	∼50	∼50

Table 3. Comparison of computational cost across different models.

length of the output of LLMs. Therefore, the inference time will become relatively longer. However, since LLM-P can only generate high-level plans, besides the inference time of GPT, its inference time depends on the motion planning model, HLSM [2], which is not short either, see the inference speed of FILM [4] (357s).

Besides, the training data we used is similar to FILM. We just split the complete episode data into single steps to train the model's single-step prediction ability. Since the number of parameters is much larger than traditional models, the training time of R2C is greater than FILM. However, FILM has 4 sub-models to train. The combination of each sub-models will also bring significant time costs.

As for the inference cost, the average inference time for FILM is about 357 seconds per episode, as it generates the entire action sequence for the episode in at once. In contrast, our R2C-Mistral-7B performs single-step reasoning, outputting the next target coordinate at each step. The average inference time for an episode is comparable between FILM and R2C-Mistral-7B (357s vs. 8×50s = 400s). Note that the model and the task can be run in parallel using the vLLM [3] speed-up technique (multi-process) to realize much faster inference. Using 4 A40 GPUs and running tasks in parallel (100 process) can speed up the single step inference time to 0.53s.

5. Model Inference Examples

We present inference examples for both GPT-4 and the open-source Mistral-7B models, each containing high-level and low-level plannings. Zero-shot R2C-GPT-4: Fig. 7 and Fig. 8 in the task "Look at a mug in lamp light." Fine-tuned R2C-Mistral-7B: Fig. 9 and Fig. 10 in the task "Turn on a lamp while holding a baseball bat."

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. 3

[2] 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. 5

[3] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023. 5

[4] 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. 3, 4, 5

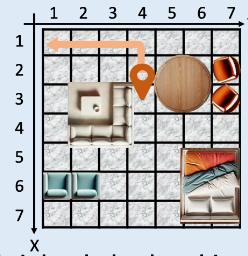
[5] Alexander Pashevich, Cordelia Schmid, and Chen Sun. Episodic transformer for vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15942–15952, 2021. 3

[6] 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. 3, 4, 5

[7] Shun-Cheng Wu, Johanna Wald, Keisuke Tateno, Nassir Navab, and Federico Tombari. Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7515–7525, 2021. 1

[8] Yingda Yin, Yuzheng Liu, Yang Xiao, Daniel Cohen-Or, Jingwei Huang, and Baoquan Chen. Sai3d: Segment any instance in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3292–3302, 2024. 1

On a 7x7 block chessboard, the rules of the game are as follows:
Establish a coordinate system with the top left grid as (1,1). Each block can be represented by coordinates.
For instance, the block in the 3rd row and 5th column is denoted as (3,5). In the chessboard, there are the following objects:
- Sofa: [[3,2],[3,3],[4,2],[4,3]]
- Bed: [[5,6],[5,7],[6,6],[6,7]]
- Table: [[2,5],[2,6],[3,5],[3,6]]
- Chair: [[6,1],[6,2]]



Your task is: Move to the nearest corner in the room.
Your current position is (3,4).
The movement is FORBIDDEN on the object blocks. You can ONLY move 1 block at a time.
Tell me what position you would move. You should first tell me your reason and then give your optimal next position, finally, judge whether the task is finished.
ALL you OUTPUT is below, DO NOT use code block or other formats:
{
"reason": "...",
"next_position": "(x, y)",
"task_done": "True/False"
}

GPT Response: {
"reason": "**Starting from position (3,4), the nearest corners would be (1,1), (1,7), (7,1), and (7,7).** Looking at direct routes, moving vertically doesn't interfere with any objects until border/corner. Your position (3,4) is open for a direct **move to the up** **leading toward corner (1,1)** without encountering any objects.",
"next_position": **"(2, 4)"**,
"task_done": "False"
}

GPT Response: {
"reason": "**Starting from the current position (2,4), to move towards the nearest corner, which is (1,1), (1,7), (7,1), or (7,7),** movement can only proceed to open spaces not occupied by other objects. The immediate adjacent spaces to (2,4) are (2,3),(1,4) and (2,5). However, (2,5) is occupied by the table, making the move impossible. An unoccupied direction is (2,3), which is clear for movement and moves closer to the **corner at (1,1)**.",
"next_position": **"(2, 3)"**,
"task_done": "False"
}

GPT Response: {
"reason": "**From the current position (2,3), the nearest corner positions are (1,1), (1,7), (7,1), and (7,7).** Visually observing the grid, moving either left or right from the current point leads into blocked sofa and table areas. The only clear path initially is downward to (3,3), but it is blocked by a sofa. **Moving one point upward to (1,3)** avoids any objects and is **a step towards corner (1,1)** which seems closest without blockages directly above.",
"next_position": **"(1, 3)"**,
"task_done": "False"
}

GPT Response: {
"reason": "**The nearest corners from the current position (1,3)** on the 7x7 board **are (1,1), (1,7), (7,1), and (7,7).** The shortest path to a corner, considering unblocked straight movements, **is towards (1,1)** by **moving directly left**, as there are no obstacles directly left of the starting position. Moves are one block at a time, so the next move from (1,3) would be to (1,2).",
"next_position": **"(1, 2)"**,
"task_done": "False"
}

GPT Response: {
"reason": "**The nearest corners to the current position (1,2) are (1,1) and (1,7).** Both can be reached by **moving horizontally**. Moving from (1,2) to (1,1) is closer and **requires a single leftward step**, which is clear of any objects.",
"next_position": **"(1, 1)"**,
"task_done": "True"
}

Figure 6. Case study of GPT-4's open-vocabulary experiment. Task: Move to the nearest corner in the room.

Please translate the subgoal instruction into subgoal expression like (GotoLocation,["Soapbar"]).
Each subgoal can be translated to ONLY ONE subgoal expression!For example, if you have 5 subgoal, you should only output 5 subgoal expression!
Please ONLY use the following subgoal verb:
['GotoLocation','PickupObject','PutObject','CoolObject','HeatObject','CleanObject','SliceObject','ToggleObject']
When the task means clean something, use CleanObject instead of ToggleObject something like faucet
When the task means heat or cook something, use HeatObject instead of PutObject into microwave etc.

Your task is Look at a mug in lamp light.
And the subgoal instruction are : ['Go to the desk in front of you, then look up to the shelves.', 'Pick up the mug from the shelf.', 'Look down to the lamp on the desk.', 'Turn on the lamp.']
Please ONLY use the following object name:
['Sink Basin', 'Arm Chair', 'Bathtub Basin', 'Bed', 'Cabinet', 'Cart', 'Coffee Machine', 'Coffee Table', 'Counter Top', 'Desk', 'Dining Table', 'Drawer', 'Dresser', 'Fridge', 'Garbage Can', 'Microwave', 'Ottoman', 'Safe', 'Shelf', 'Side Table', 'Sofa', 'Stove Burner', 'TV Stand', 'Toilet', 'Faucet', 'Floor Lamp', 'None', 'Mug', 'Desk Lamp']
ONLY answer with DIRECTLY like the example below , notice that an subgoal except PutObject only have one object and this example's task is "put a cooked slice of bread on to the counter":

(GotoLocation,["Countertop"])|(PickupObject,["Knife"])|(GotoLocation,["Bread"])|(SliceObject,["Bread"])|(GoToLocation,["Drawer"])|(PutObject,["Knife","Drawer"])|(GoToLocation,["Dining Table"])|(PickupObject,["Bread"])|(GoToLocation,["Countertop"])|(PutObject,["Bread","Microwave"])|("HeatObject",["Bread"])|(GoToLocation,["Countertop"])|(PutObject,["Bread","Countertop"])

GPT Response:
(GotoLocation,["Shelf"])|(PickupObject,["Mug"])|(GotoLocation,["Desk Lamp"])|(ToggleObject,["Desk Lamp"])

Figure 7. GPT-4's high-level planning prompt and answer.

On a 64*64 block chessboard, the rules of the game are as follows:
Establish a coordinate system with the top left grid as (1,1). Each block can be represented by coordinates.
For instance, the block in the 3rd row and 5th column is denoted as (3,5). its right is (3,6), its left is (3,4), its up is (2,5), its down is (4,5).
spatial relation:[left,right,up,down,leftup,rightup,leftdown,rightdown]
In the chessboard, there are the following objects: Bed: [[14, 29], [14, 30], [14, 31], [14, 32], [14, 33], [14, 34], [14, 35], [14, 36], [14, 37], [14, 38], [14, 39], [14, 40], [14, 41], [14, 42], [14, 43], [15, 29], [15, 30], [15, 31], [15, 32], [15, 33], [15, 34], [15, 35], [15, 36], [15, 37], [15, 38], [15, 39], [15, 40], [15, 41], [15, 42], [15, 43], [16, 29], [16, 30], [16, 31], [16, 32], [16, 33], [16, 34]]
Unexplore: [[16, 40], [16, 41], [16, 42], [16, 43], [17, 40], [17, 41], [17, 42], [17, 43], [18, 40], [18, 41], [18, 42], [18, 43], [19, 40], [19, 41], [19, 42], [19, 43], [20, 40], [20, 41], [20, 42], [20, 43], [21, 41], [21, 42], [21, 43], [22, 40], [22, 41]]
Explore: [[21, 40], [23, 32], [23, 33], [23, 34], [23, 35], [23, 36], [23, 37], [23, 38], [23, 39], [24, 30], [24, 31], [24, 32], [24, 33], [24, 34], [24, 35], [24, 36], [24, 37], [24, 38], [25, 30], [25, 31], [25, 32], [25, 33], [25, 34], [25, 35], [25, 36], [25, 37]]
Obstacle: [[24, 29], [25, 29], [26, 29], [27, 29], [28, 29], [29, 29], [29, 36], [30, 29], [30, 36], [33, 31], [33, 32], [33, 33], [33, 34], [33, 35], [33, 36], [34, 32], [34, 33], [34, 34], [34, 35]]
Desk: [[27, 39], [27, 40], [28, 38], [28, 39], [28, 40], [29, 37], [29, 38], [29, 39], [29, 40], [30, 37], [30, 38], [31, 37], [31, 38], [32, 37], [32, 38], [33, 37], [33, 38], [33, 39], [34, 36], [34, 37], [34, 38], [34, 39], [34, 40]]
Shelf: [[30, 39], [30, 40], [31, 29], [31, 30], [31, 39], [32, 29], [32, 30], [33, 29], [33, 30], [33, 40], [34, 29], [34, 30]]
Mug: [[31, 40], [32, 39], [32, 40]]
Agent: [[32, 33]]

The movememt is forbidden on the object block.
You can only move 1 block at a time.
Your task is [['GotoLocation', ['shelf']], ['PickupObject', ['mug']], ['GotoLocation', ['desklamp']], ['ToggleObject', ['desklamp']], ['NoOp', []]]
Your subgoal is (GotoLocation, [mug])
Your position is at [32, 33]
Your available position is [[31, 33], [32, 32], [32, 34]]
Tell me what position you would move. You should first tell me your reason ,which first FIRST analyze the spatial relation between target position(if you see the subgoal object, else you should explore the room as far as you can) and your position, NEXT analyze each available position(the spatial relation between your position and it AND if it makes you near the target), and judge whether the sub-goal is finished.
ALL you OUTPUT is below, DO NOT use code block or other format:
{
"reason": "...",
"next_position": "(x, y)",
"subgoal_done": "True/False"
}

GPT Response:
{
"reason": "The subgoal is to go to the location of the mug. The coordinates for the mugs are at [31, 40], [32, 39], and [32, 40]. Currently, the agent is at location [32, 33]. The shortest path initially appears to be moving directly to the right from the agent's current position in order to approach one of the mug positions most directly. Moving right from [32, 33] leads to [32, 34], moving closer in the vertical lane to the target mug coordinates ([32, 39] and [32, 40]). From [32, 34], the agent can then potentially proceed further right in subsequent moves to reach [32, 39] or [32, 40]. Moving left to [32, 32] or up to [31, 33] would not progress towards the mug's location efficiently as these moves do not reduce the vertical distance to the target mug coordinates.",
"next_position": "(32, 34)",
"subgoal_done": "False"
}

Figure 8. GPT-4's low-level planning prompt and answer.

You are an indoor agent.
Please divide the TASK into some sub-steps according to the DESCRIPTION.
Each sub-step MUST be expressed in the form of (Action, [Object]), where Action is one of (GotoLocation, PickupObject, PutObject, CoolObject, HeatObject, CleanObject, SliceObject, ToggleObject, NoOp), and Object refers to one or more objects related to the Action.
TASK: Turn on a lamp while holding a baseball bat.
DESCRIPTION: ['walk forwards a bit towards the baseball bat', 'grab the baseball bat leading against the wall', 'turn and walk over to the right side of the computer desk', 'turn on the lamp on the computer desk']

R2C-Mistral-7B Response:
[('GotoLocation', ['baseballbat']), ('PickupObject', ['baseballbat']), ('GotoLocation', ['desklamp']), ('ToggleObject', ['desklamp']), ('NoOp', [])]

Figure 9. Mistral's high-level planning prompt and answer.

On a 64*64 block chessboard, the rules of the game are as follows:
Establish a coordinate system with the top left grid as (1,1). Each block can be represented by coordinates.
for instance, the block in the 5th column and 3rd row is denoted as (5,3).
In the chessboard, there are the following explored objects: obstacle: [[27, 28], [27, 29], [27, 30], [27, 31], [27, 32], [27, 33], [27, 34], [27, 35], [27, 36], [27, 37], [28, 26], [28, 30], [28, 31], [28, 32], [28, 33], [28, 34], [28, 35], [28, 36], [28, 37], [28, 39], [28, 40], [28, 41], [28, 42], [29, 26], [30, 44], [31, 29], [31, 44], [32, 43], [32, 44], [33, 42], [33, 43], [33, 44], [34, 42], [34, 43], [34, 44], [35, 42], [35, 43], [35, 44], [36, 44], [37, 43], [37, 44], [38, 42], [38, 43], [38, 44], [39, 44], [41, 35], [41, 36]]
garbagecan: [[27, 43], [27, 44], [28, 43], [28, 44], [29, 43], [29, 44]]
desk: [[28, 27], [29, 27], [29, 29], [30, 26], [30, 27], [30, 28], [30, 29], [31, 26], [31, 27], [31, 28], [32, 26], [32, 28], [32, 29], [33, 26], [33, 28], [33, 29], [34, 29]]
baseballbat: [[28, 28], [29, 28]]
desklamp: [[32, 27], [33, 27]]
sidetable: [[34, 26], [34, 27]]
drawer: [[34, 28], [39, 35], [39, 36], [39, 37], [39, 38], [40, 38]]
bed: [[35, 26], [35, 27], [35, 28], [35, 29], [35, 30], [35, 31], [35, 32], [35, 33], [35, 34], [36, 26], [36, 27], [36, 28], [36, 29], [36, 30], [36, 31], [36, 32], [36, 33], [36, 34], [37, 26], [37, 27], [37, 29], [37, 30], [37, 31], [37, 32], [37, 33], [37, 34], [38, 28], [38, 29], [38, 30], [38, 31], [38, 32], [38, 33], [38, 34], [39, 28], [39, 29], [39, 30], [39, 31], [39, 32], [39, 33], [39, 34]]
shelf: [[38, 41], [39, 41], [39, 42], [39, 43], [40, 42], [40, 43]]
dresser: [[40, 34], [40, 35], [40, 36], [40, 37], [41, 37], [41, 38], [42, 37]].
The movement is forbidden on the object block.
You can only move 1 block at a time.
Task: (GotoLocation, [baseballbat])
Current Position: [32, 31]
History Trajectory: [[32, 32], [32, 33]]
Please select your next position from [[31, 31], [33, 31], [32, 30], [32, 32]].
Please analyze the options and choose the best one to finish the task.

R2C-Mistral-7B Response:
Based on the given information, **the current position of the agent is [32, 31], and the target is baseballbat, which is located at [[28, 28], [29, 28]].**
There are 4 options for the agent to move towards the target position. Now, let's analyze the options given:
[31, 31]: This position is at the above of the agent. With this step, the agent approaches closer to the target position.
[33, 31]: This position is at the below of the agent. This action will distance the agent from the target position.
[32, 30]: This position is at the on the left side of the agent. This action will distance the agent from the target position.
[32, 32]: This position is at the on the right side of the agent. This move takes the agent away from the target position.
Based on the analysis above, the best option for the agent to move towards the target position is [31, 31]. Therefore, the answer is:
 # Next position: [31, 31]

Figure 10. Mistral's low-level planning prompt and answer.