Integrating Common Sense and Planning with Large Language Models for Room Tidying

Zhanxin Wu, Bo Ai, and David Hsu
National University of Singapore
{zhanxinwu, bo.ai, dcsdavid}@u.nus.edu

***Abstract*—Do you want a personal housekeeper robot? This project seeks to endow robots with the capability of tidying up messy rooms with brief natural language descriptions of the environment. We address three key challenges: (i) incomplete map information in the description, (ii) commonsense understanding of object locations, and (iii) long-horizon planning and acting to achieve the objective. To tackle these challenges, we leverage Large Language Models' (LLMs) understanding of typical layouts of human-living environments and object locations, as well as programming and control skills for action execution. Specifically, we prompt ChatGPT to reconstruct complete map representations from partial descriptions, then generate a high-level action plan in the form of Python functions, and finally refine the plans with atomic actions executable by the robot. We show that our framework enables effective room tidying with limited human instruction guidance. On simulation and real-world maps, it is able to find a place missing out from human description within three interactions with humans. In the simulation environment, it is capable of putting more than 80% household objects in their desired place. This study provides preliminary evidence that LLMs have common sense about the spatial layout of human-living environments and object arrangements, and this work connects this knowledge to robotics tasks.**

I. Introduction

Imagine a robot housekeeper that can operate in any house in the world, and you invite it to your messy home to put things back in place. The robot has no idea of the layout of your room, though it would be better if it knew how to perform the task once you *briefly* describe your house. However, this seemingly easy task has a plethora of challenges, such as:

i. Your description might not contain the complete information for the robot to navigate between two locations. For example, you may miss out bathroom in your description, then the robot has to *infer* the location of the bathroom in order to put a toothbrush in place. This common sense is absent in many existing robotic systems.
ii. The robot may not know where the misplaced objects *should* be, as this involves understanding the functionality of objects and human living habits. For instance, typical robots should know that a cup cannot float in the air, but they may not comprehend why the cup should not be placed on a sofa.
iii. The robot needs to reason about the actions to put the objects pack in place. This involves both navigation and manipulation at a potentially long horizon.

In this project, we examine these three key issues by exploiting the commonsense and planning capabilities that Large Language Models (LLMs) embody. Specifically, our approach involves exploiting the knowledge from LLMs at three levels, *i.e.*, spatial layout understanding, commonsense reasoning, and programming and control skills. First, LLMs are utilized to construct comprehensive map representations from incomplete textual descriptions of the environment. Then, we use LLMs to generate high-level task plans for room tidying. Finally, these high-level plans are refined to atomic actions that are executable by a robot. Our system is illustrated in Figure 1.

Experimentally, our approach shows promising results in typical human-living environments. To evaluate LLMs' understanding of spatial layout, we designed a task where LLMs are asked to recover a full map representation from a partial human language description. We show that LLMs have a decent understanding of typical layouts and can recover map information with a small amount of human guidance. Then, we evaluate LLMs' capability on object re-arrangement in a simulated household environment. We discovered that our framework enables it to put more than 80% of objects back in the correct places. These findings demonstrate that LLMs have the common sense knowledge that is critical to household tasks but has not been fully explored by existing work.

II. Related Work

Task and Motion Planning. Task and Motion planning [15] [7] is a principled framework that allows robots to perform long-horizon tasks. Previous work utilizes symbolic planning and pre-defined domains as underlying action languages [3] [10] [4] [6]. The development of LLMs has exhibited significant potential in planning and reasoning for simple tasks [14] [5] [9] [8]. Previous works studies fine-tune LLMs with additional environment inputs [12] or directly output actions [16]. However, these studies solely depend on language instructions using an open-world vocabulary, which may potentially result in ungrounded plans. Recent work used programmatic LLMs prompts for multi-task planning [13]. Our framework adopts the Pythonic program representation from [13] to interact with LLMs. However, our task has a much longer horizon as it involves both navigation and manipulation, and we not only exploit LLM's actionable knowledge but also their common sense understanding of spatial layout and object locations.

LLMs for Robotics Tasks. Previous research has integrated commonsense knowledge into robot tasks [5] [1] [19]. Some work extracts commonsense knowledge on object configurations

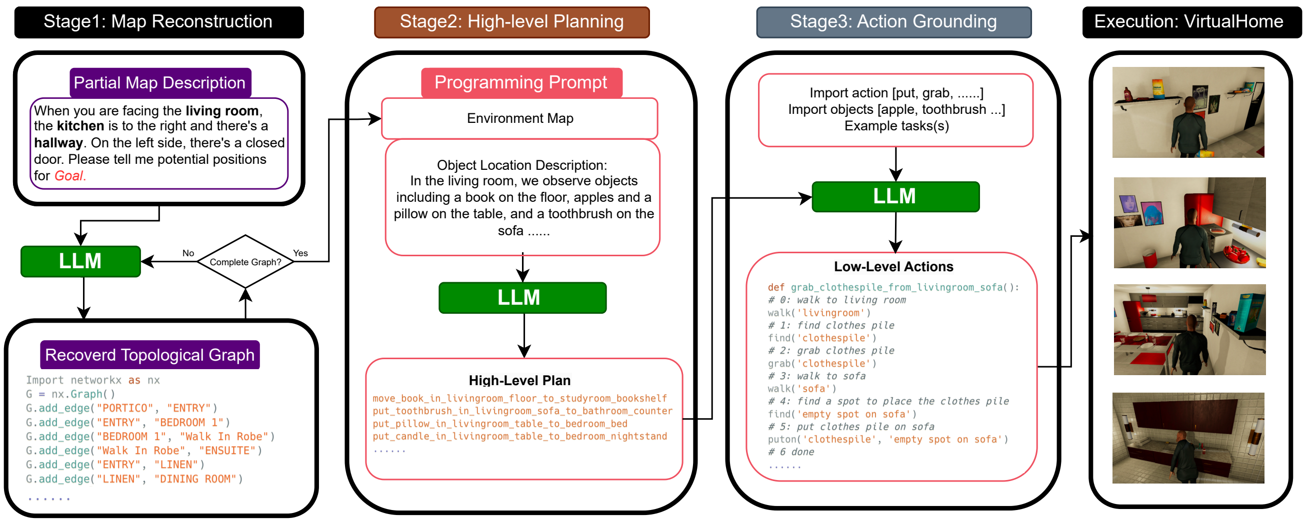


Fig. 1: The proposed system architecture. We exploit LLM's commonsense reasoning capability to construct maps from partial descriptions, identify misplaced objects in the first two stages, and eventually use LLM to complete the action plan for execution.

from LLMs and applies it to various scene geometries for object rearrangement [2] [9], whereas others focus on the common sense of object relationships in the environment for efficient object-goal navigation [18]. Another recent work [17] seeks to learn user preference with LLMs for object re-arrangements. Our work is different from these prior works in that, we study LLMs' common sense knowledge of spatial layouts and object placements in typical human-living environments, and integrate it with planning to achieve both navigation and object re-arrangement with partial description of the environment.

III. Methodology

A. Problem Formulation

In this task, we focus on putting misplaced objects back to their typical locations with a partial prior map, *e.g.*, returning a book on the sofa to the shelf, and we assume only partial map information of the environment and flawless execution of low-level actions. Ideally, the task consists of three stages.

i. Model initialization. The user provides a description of the environment's layout, which could only be partial, as well as descriptions of the objects in the room.
ii. Planning and model update. The agent generates a high-level plan for re-arranging the objects, *e.g.*, "move the book on the living room sofa to the study room bookshelf." If the target location is not in the given partial map description, the LLM would propose a potential position for the study room and add it to the current map representation.
iii. Repeat the previous stage for every misplaced object.

However, this formulation makes it hard to evaluate a system's performance, as the model update and planning are intervened. Here we make a simplifying assumption that the agent should recover the complete map information before planning. The actual formulation is as follows.

i. Model initialization. This step remains the same.
ii. Map recovery. Given the partial description of the environment, the agent interacts with the user to recover a full map representation. The challenge is that the agent should have commonsense understanding of the typical spatial layouts of household environments to minimize interactions with the user.
iii. Planning. Given the description of objects in the scene from the user and the map representation, the agent returns misplaced objects to their desired locations. The challenges are to identify the misplaced objects and their desirable locations, and to generate executable plans.
iv. Repeat the previous stage for every misplaced object.

B. Proposed Framework

Given the partial human description of the map T, language description of scene S, and description of locations of all movable objects O in the scene, our framework consists of three stages.

1) Stage 1: Construct Map Representation from Partial Information: We iteratively recover a full map representation G from the partial language description T with an LLM, *i.e.*,

$$G_0 = LLM(T, P_{1,0}) \tag{1}$$

$$G_i = LLM(G_{i-1}, P_{1,i}), i = 1, 2, 3, ... \tag{2}$$

where G_{i-1} is an intermediate map representation, and $P_{1,i}$ is a prompt provided by the human indicating the correctness of the map so that the LLM can update it in the next round. The iteration stops when the complete graph has been obtained. The final map G is the last map generated by the LLM, *i.e.*,

$$G = G_N \tag{3}$$

where N is the number of iterations taken. Here we use a Pythonic topological graph representation for G where each node is a subspace of the environment with a distinct semantic meaning, *e.g.*, living room. We use such a representation for two reasons: (i) Topological structure is sufficient to enable the agent to navigate across places, assuming low-level navigation skills are provided by the simulator; (ii) Pythonic programming language offers a concise and structured description of the

TABLE I: Results on the map reconstruction task.

Environment	# Places	Left-Out Places	# Interaction Rounds	
			Ours	Random Guess
VH Apartment	5	Bathroom	**1.20** $\pm$ 0.45	2.82 $\pm$ 1.50
		Bedroom	**1.60** $\pm$ 0.55	3.32 $\pm$ 1.43
Real Apartment	15	Bathroom	**3.20** $\pm$ 1.30	8.00 $\pm$ 4.56
		Bedroom	**2.40** $\pm$ 0.55	7.20 $\pm$ 4.01
Hospital	20	Nurse's Station	**1.40** $\pm$ 0.55	7.60 $\pm$ 5.64
		Bathroom	**2.20** $\pm$ 2.17	5.60 $\pm$ 2.93
School	17	IT Service	**3.40** $\pm$ 3.13	6.60 $\pm$ 3.39
		Bathroom	**3.60** $\pm$ 1.34	5.00 $\pm$ 5.10
Airport	25	Immigration	**1.80** $\pm$ 0.45	7.20 $\pm$ 6.85
		Bathroom	**1.60** $\pm$ 0.55	6.20 $\pm$ 5.23
		Info Desk	**1.60** $\pm$ 1.34	8.20 $\pm$ 3.31
Mall	18	Bathroom	**5.80** $\pm$ 0.83	7.40 $\pm$ 3.38

topological graph, compared to free-form natural language.

To illustrate, an example is presented in Appendix VI-C.

2) Stage 2: Generate High-level Action Plan: In this stage, we generate a high-level action plan ζ from map representation G and the description of the location of objects O, *i.e.*,

$$\zeta = LLM(G, O, P_2) \tag{4}$$

where P_2 is a prompt stating the objective of generating a high-level plan.

Here ζ is in the form of a list of Pythonic function skeletons that achieve the objective at the high level, *e.g.*, `move_object_from_placeA_to_placeB`. The challenge here is identifying misplaced objects and providing a sketch solution accordingly. The high-level plans will be grounded into executable actions in the next stage. We disentangle stage 2 and stage 3 because we found that this progressive-style query allows the LLM to produce more sensible outputs. This is similar to humans, who may better solve a problem if guidance is given along the way.

3) Stage 3: Generate Executable Actions: Finally, we generate an executable action plan ξ from ζ, *i.e.*,

$$\xi = LLM(\zeta, P_3) \tag{5}$$

where P_3 is a prompt specifying the set of executable actions and asking for a concrete action plan. Here, LLM needs to understand the capabilities of our agent and complete the skeletal program ζ with actions executable by our agent.

The full system architecture is illustrated in Figure 1.

IV. Experiments

In this subsection, we seek to verify the following hypotheses for our framework:

H_1. LLMs embody commonsense knowledge about spatial layouts of typical human-living environments, enabling map construction with only partial descriptions.

H_2. LLMs have commonsense knowledge about the typical placement of common objects.

H_3. Hierarchical planning enables LLMs to better reason at a long horizon, compared to "end-to-end" planning.

As H_1. focuses on map representation construction while H_2. and H_3. are related to object manipulation, we will use separate experiments to verify the hypotheses. In all experiments, we use GPT-3.5-turbo, 2023 March 23 version from OpenAI.

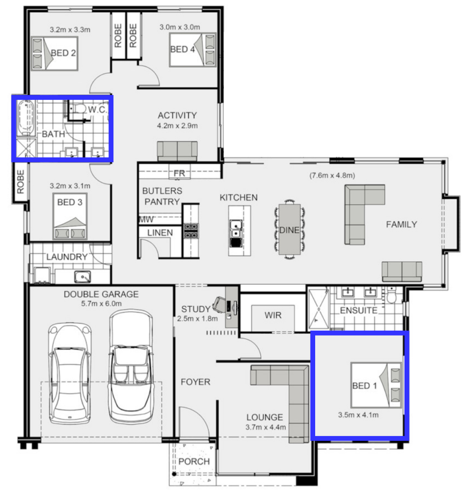


Fig. 2: Example of a real-world map. The regions marked by the blue boxes represent the areas missing from the user descriptions, which are subsequently recovered by the LLM.

A. Experiments on Map Construction

This subsection focuses on H_1.. We use simulation environments and real-world scenes to test the model's capability of recovering the full map representation from partial information. For simulation environments, we will be using maps from the VirtualHome (VH) [11]. The platform simulates diverse household scenes, which suits our intended application scenarios. Moreover, we use real-world maps of diverse places to better understand LLM's capability. We choose common public and private places, including hospitals, malls, airports, and apartments, to cover diverse human-living environments.

1) Metrics:

i. Number of Interaction Rounds: The number of rounds required to recover one missing place on average. This number quantifies the amount of human effort required to exploit such common sense for map recovery.

2) Results: Results are presented in Table I. It can be observed that LLMs are capable of suggesting the correct location for unseen places within three interaction rounds. However, our method performs less well on a mall layout, due to a non-typical layout in one of the test cases: a bathroom is next to a health store. Nevertheless, compared to the random guesser, our framework reduces interactions by up to 80% and has a much smaller variance in performance, supporting H_1..

One example of a real-world map is presented in Figure 2. More examples are given in Appendix VI-C.

B. Experiments on Room Tidying

This subsection focuses on H_2. and H_3.. As object relocation involves actions and executions, we perform our evaluation in purely simulation environments, *i.e.*, VirtualHome [11]. The simulator contains models for approximately 130 kinds of common household objects, which makes the task challenging.

TABLE II: Results on the room arrangement task. The full table with standard deviations is in Appendix VI-D.

Room	Method	Number of Misplaced Objects											
		2			4			8			12		
		SRC	ER	GCR	SRC	ER	GCR	SRC	ER	GCR	SRC	ER	GCR
Living Room	Our Method	**1.00**	**1.00**	**1.00**	**0.80**	0.76	**0.95**	**0.60**	0.83	**0.73**	**0.40**	0.70	**0.69**
	ProgPrompt	0.60	**1.00**	0.70	0.40	**0.92**	0.70	0.00	**0.98**	0.20	0.00	**0.79**	0.15
Kitchen	Our Method	**0.60**	**1.00**	**0.70**	**0.60**	0.90	**0.83**	**0.60**	0.93	**0.85**	**0.20**	0.76	**0.78**
	ProgPrompt	**0.60**	0.96	**0.70**	0.20	**0.97**	0.65	0.20	**0.98**	0.50	0.00	**0.94**	0.17
Bathroom	Our Method	**1.00**	**1.00**	**1.00**	**0.60**	**1.00**	**0.90**	**0.40**	**1.00**	**0.90**	**0.40**	**0.96**	**0.57**
	ProgPrompt	0.40	0.89	0.50	0.20	0.93	0.45	0.00	0.89	0.35	0.00	0.81	0.20
Bedroom	Our Method	**0.80**	0.90	**0.90**	**0.80**	**0.96**	**1.00**	0.40	**0.96**	**0.93**	**0.60**	**0.98**	**0.65**
	ProgPrompt	0.40	**0.91**	0.60	0.20	0.82	0.35	0.00	**0.96**	0.25	0.00	0.94	0.22

To initialize the environment, we randomly place k movable objects at locations where they typically do not appear, *e.g.*, we could place a cup on a sofa. Then we execute our framework as described in Section III-B. However, as we now focus on room tidying only, we will ensure that the recovered map representation is correct.

1) Metrics:

i. Success Rate (SR): One success is an indicator variable that takes the value one only if all objects are returned to their desired places, as deemed appropriate by humans. The success rate is the fraction of success among all trials, reflecting the overall system performance.
ii. Execution Rate (ER): ER is defined as the fraction of atomic actions in the action plan executable by the agent, regardless of whether or not it contributes to the task objective (return objects to their desired places). This metric reflects the LLM's ability to understand the agent's capability and perform planning and control.
iii. Goal Condition Rate (GCR): GCR is the fraction of the objects returned to their desired places over the total number of misplaced objects. This metric measures the extent to which the agent can complete the task and can be viewed as a more granular metric compared to SR.

2) Results: Full results are presented in Table II. Across all environments and the number of misplaced objects, our method can return most of the objects (GCR > 60%). In environments with fewer misplaced objects, our SR and GCR are close to 1. This shows that LLMs have common sense knowledge about object placement, supporting H_2..

The GCR of our method exceeds 60% for all setups, and sometimes 80% for scenarios with a small number of misplaced objects, such as 2 or 4. It shows the ability of LLMs to identify misplaced objects and their proper locations, supporting H_2..

Compared to ProgPrompt, our approach exhibits significant improvements in SR and GCR. In particular, when there are a large number of misplaced objects, *e.g.*, 8 and 12, the baseline ProgPrompt almost never completes a trial (SR $\approx$ 0). This shows that our hierarchical planning is effective in enabling LLMs to reason about action plans. It is interesting to see that ProgPrompt has a high ER even when the SR is zero, suggesting that ProgPrompt tends to generate executable but irrelevant actions for the task, such as picking up an object and then putting it back in the same location. This supports H_3. that our hierarchical planning enables LLMs to better reason about action plans. In fact, this result is rather intuitive, just like humans who write better articles with an outline. We hope this finding to be transferable to other tasks.

However, one common failure case of our method is its tendency to put objects in one place regardless of their functionality and desirable locations. For example, when there are clothes, fruits, and plates on a sofa, LLMs may move all the items inside the kitchen cabinet. We conjecture that this is due to its limitation in understanding the physical and functional structure of human-living environments.

V. Discussion

In this paper, we presented a framework that enables an agent to perform room tidying with a partial map description. Specifically, we exploit LLMs' common sense about spatial layout and object arrangements to recover map information and generate high-level plans to return misplaced objects. Then, we exploit its capability of planning to ground the high-level task plans into low-level actions for execution. We show that our framework enables successful map recovery and object re-arrangement with limited human guidance.

Apart from the results, there are several key takeaways. First, LLMs have a common-sense understanding of typical human-living environments, which could be useful for robot navigation tasks, especially when the map information is incomplete. Second, LLMs understand typical object placement, which is important for housekeeping tasks. Third, hierarchical planning allows LLMs to better reason about action plans. We hope these insights to be useful in future research.

While our approach shows promising results, our system has some limitations. First, it does not have a recovery mechanism for cases where the objects are returned to the wrong place. Second, this framework requires humans to be in the loop, which might be inconvenient for users in practice. Moreover, common sense is only applicable to typical scenarios, but there might be variance in user preference in reality. We acknowledge the limitations, but we hope this work provides insights into exploiting LLMs for household tasks.

References

[1] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. Comet: Commonsense transformers for automatic knowledge graph construction. *arXiv preprint arXiv:1906.05317*, 2019.

[2] Yan Ding, Xiaohan Zhang, Chris Paxton, and Shiqi Zhang. Task and motion planning with large language models for object rearrangement. *arXiv preprint arXiv:2303.06247*, 2023.

[3] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[4] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 440–448, 2020.

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

[6] Yu-qian Jiang, Shi-qi Zhang, Piyush Khandelwal, and Peter Stone. Task planning in robotics: an empirical comparison of pddl-and asp-based systems. *Frontiers of Information Technology & Electronic Engineering*, 20: 363–373, 2019.

[7] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, pages 1470–1477. IEEE, 2011.

[8] Yash Kant, Arun Ramachandran, Sriram Yenamandra, Igor Gilitschenski, Dhruv Batra, Andrew Szot, and Harsh Agrawal. Housekeep: Tidying virtual households using commonsense reasoning, 2022.

[9] 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.

[10] Dana Nau, Yue Cao, Amnon Lotem, and Hector Munoz-Avila. Shop: Simple hierarchical ordered planner. In *Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2*, pages 968–973, 1999.

[11] 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.

[12] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[13] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. ProgPrompt: Generating situated robot task plans using large language models. 2022.

[14] 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, 2023.

[15] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 639–646. IEEE, 2014.

[16] 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.

[17] Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, and Thomas Funkhouser. Tidybot: Personalized robot assistance with large language models. *arXiv preprint arXiv:2305.05658*, 2023.

[18] Bangguo Yu, Hamidreza Kasaei, and Ming Cao. L3mvn: Leveraging large language models for visual target navigation, 2023.

[19] Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. From recognition to cognition: Visual commonsense reasoning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6720–6731, 2019.

VI. Appendix

A. Example of Map Construction

As an illustration, the following conversation history between a user and ChatGPT shows how we construct a complete Pythonic topological map representation of the environment from a partial natural language description in stage 1.

`---- The conversation begins ----`

User: I am in my apartment. Upon stepping into the living room, the kitchen is on my right-hand side and one study room is on my left-hand side. Please tell me the potential position for a bedroom.
Please utilize Python to create a topological graph of the apartment, with nodes representing rooms and edges indicating the connections between them.

ChatGPT: Certainly! I can help you visualize the apartment layout using Python and create a topological graph for it. To represent the graph, we can use the NetworkX library in Python. Here's an example code that creates the topological graph for your apartment layout:

```
import networkx as nx
```

```
# Create a directed graph
G = nx.DiGraph()

# Add nodes representing the rooms
G.add_node('Living Room')
G.add_node('Kitchen')
G.add_node('Study Room')
G.add_node('Bedroom')

# Add edges representing the
# connections between rooms
G.add_edge('Living Room', 'Kitchen')
G.add_edge('Living Room', 'Study Room')
G.add_edge('Living Room', 'Bedroom')
```

In this code, we create a graph from NetworkX. We add nodes for each room ('Living Room', 'Kitchen', 'Study Room', and 'Bedroom') using the add_node method, and we add edges to represent the connections between the rooms using the add_edge method. You can modify the code to add more rooms or adjust the connections as per your apartment layout.

User: Hi, I explored the living room and discovered that there is no bedroom connected to it. Could you please suggest a potential position for the bedroom and update the topological graph accordingly?

ChatGPT: Certainly! If there is no bedroom connected to the living room, we can consider adding the bedroom adjacent to the kitchen. Here's the updated code that reflects this potential position for the bedroom:

```
import networkx as nx

# Create a directed graph
G = nx.DiGraph()

# Add nodes representing the rooms
G.add_node('Living Room')
G.add_node('Kitchen')
G.add_node('Study Room')
G.add_node('Bedroom')

# Add edges representing the
# connections between rooms
G.add_edge('Living Room', 'Kitchen')
G.add_edge('Living Room', 'Study Room')
G.add_edge(''Kitchen', 'Bedroom'')
```

In this updated code, I added a new node for the 'Bedroom' using the add_node method, and I also added a new edge from the 'Kitchen' to the 'Bedroom' using the add_edge method. This represents the potential position for a bedroom in your apartment layout.
Running this code will generate a topological graph that includes the 'Bedroom' as a node connected to the 'Kitchen'.

User: I explored the kitchen and discovered the bedroom. Thanks for your suggestions. Now, I am looking for the bathroom. Could you please suggest a potential position for bathroom and update the topological graph accordingly?

ChatGPT: Certainly! Based on the information you provided, we can consider adding the bathroom adjacent to the bedroom. Here's the updated code that reflects this potential position for the bathroom:

```
import networkx as nx

# Create a directed graph
G = nx.DiGraph()

# Add nodes representing the rooms
G.add_node('Living Room')
G.add_node('Kitchen')
G.add_node('Study Room')
G.add_node('Bedroom')

# Add edges representing the
# connections between rooms
G.add_edge('Living Room', 'Kitchen')
G.add_edge('Living Room', 'Study Room')
G.add_edge('Kitchen', 'Bedroom')
G.add_edge('Bedroom', 'Bathroom')
```

In this updated code, I added a new node for the 'Bathroom' using the add_node method, and I added a new edge from the 'Bedroom' to the 'Bathroom' using the add_edge method. This represents the potential position for the bathroom adjacent to the bedroom.
Running this code will generate a topological graph that includes the 'Bathroom' as a node connected to the 'Bedroom'.

`---- The conversation ends ----`

Remarks: In the above conversation, ChatGPT took two rounds for the bedroom and one round for the bathroom to complete the map construction.

B. *Example of Room Tidying*

As an illustration, the following conversation history between a user and ChatGPT shows how we generate high-level plans and ground these to executable actions in stage 2 and 3.

`---- The conversation begins ----`

User: Imagine yourself as a housekeeper in my apartment. At the moment, you find yourself in the living room where you observe a cupcake and a clothes pile resting on the sofa. Additionally, on the table, there is a mug, a book and a toothbrush on the table. In this situation, what steps would you take?

ChatGPT: Here are the function names for the actions I would take in this scenario:

1. put_cupcake_in_livingroom_sofa_to_kitchen_counter
2. put_clothespile_in_livingroom_sofa_to_bedroom_closet
3. put_plate_in_livingroom_table_to_kitchen_sink
4. put_book_in_livingroom_table_to_bedroom_nightstand
5. put_toothbrush_in_livingroom_table_to_bathroom_counter

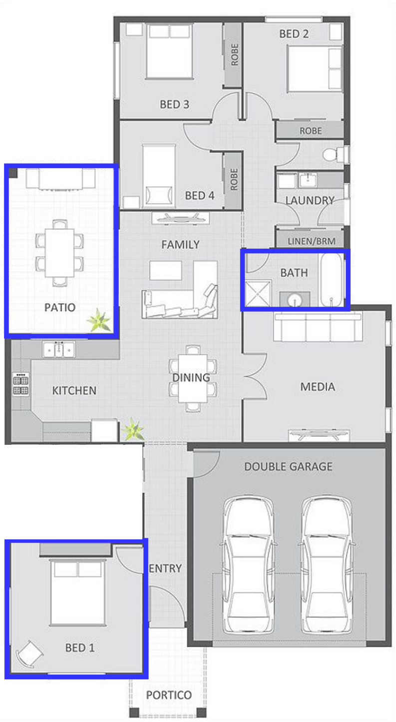
User: Please modify the given function, "put_toothbrush_in_livingroom_sofa_to_bathroom_counter" as demonstrated in the example function below.
[ProgPrompt Sample tasks]

ChatGPT:

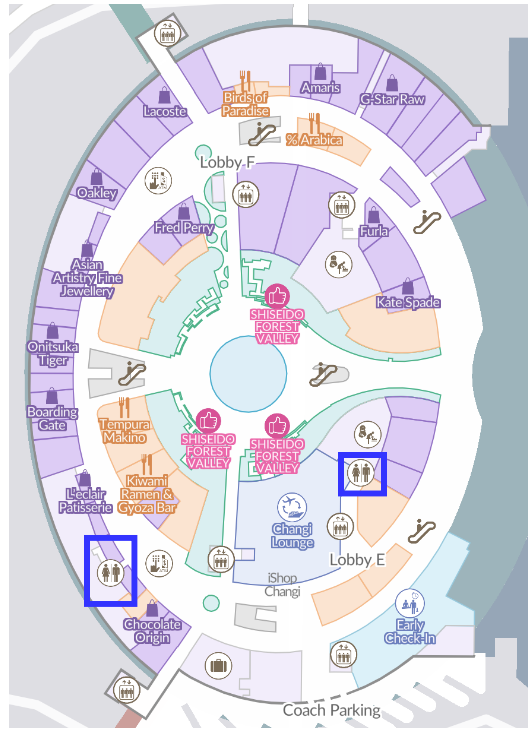
```
def put_toothbrush_in_livingroom_sofa_to_
bathroom_counter():
    # 0: walk to living room
    walk('livingroom')
    # 1: find sofa
    find('sofa')
    # 2: look for toothbrush on the sofa
    lookat('sofa')
    find('toothbrush')
    # 3: grab toothbrush
    assert('close' to 'toothbrush')
    grab('toothbrush')
    # 4: walk to bathroom
    walk('bathroom')
    # 5: find bathroom counter
    find('bathroomcounter')
    # 6: put toothbrush on counter
    assert('toothbrush' in 'hands')
    puton('toothbrush', 'bathroomcounter')
    # 7: Done
```

---- The conversation ends ----

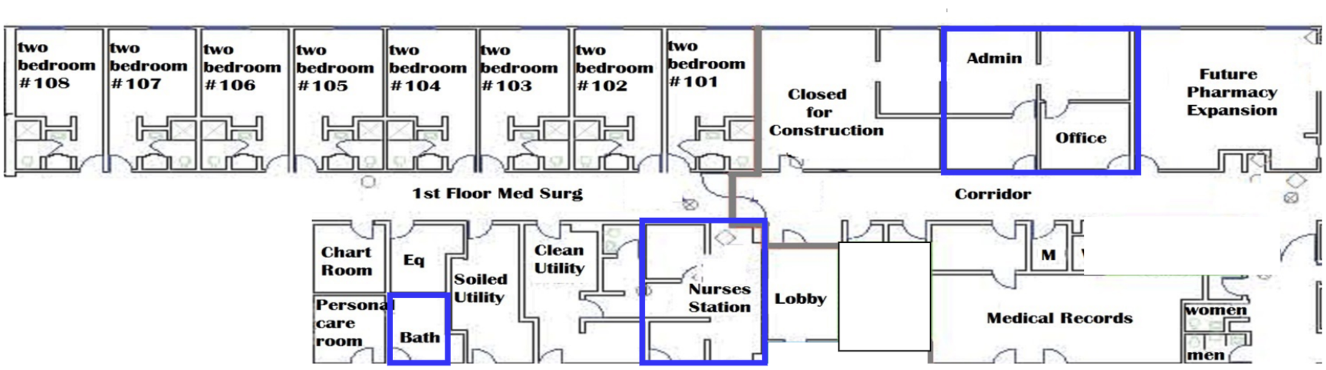
C. Examples of Real-World Map Reconstruction



(*a*) Real-world apartment



(*b*) Real-world mall



(*c*) Real-world hospital

Fig. 3: Examples of real-world maps. The regions marked by the blue boxes represent the areas missing from the user descriptions, which are subsequently recovered by the LLM.

D. Additional Results

Room	Method	Number of Misplaced Objects											
		2			4			8			12		
		SRC	ER	GCR	SRC	ER	GCR	SRC	ER	GCR	SRC	ER	GCR
Living Room	Our Method	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00	**0.80** $\pm$ 0.40	0.76 $\pm$ 0.21	**0.95** $\pm$ 0.10	**0.60** $\pm$ 0.49	0.83 $\pm$ 0.09	**0.73** $\pm$ 0.16	**0.40** $\pm$ 0.49	0.70 $\pm$ 0.15	**0.69** $\pm$ 0.26
	ProgPrompt	0.60 $\pm$ 0.49	**1.00** $\pm$ 0.00	0.70 $\pm$ 0.40	0.40 $\pm$ 0.49	**0.92** $\pm$ 0.07	0.70 $\pm$ 0.24	0.00 $\pm$ 0.00	**0.98** $\pm$ 0.04	0.20 $\pm$ 0.17	0.00 $\pm$ 0.00	**0.79** $\pm$ 0.39	0.15 $\pm$ 0.18
Kitchen	Our Method	**0.60** $\pm$ 0.49	**1.00** $\pm$ 0.00	**0.70** $\pm$ 0.40	**0.60** $\pm$ 0.49	0.90 $\pm$ 0.15	**0.83** $\pm$ 0.21	**0.60** $\pm$ 0.49	0.93 $\pm$ 0.09	**0.85** $\pm$ 0.20	**0.20** $\pm$ 0.40	0.76 $\pm$ 0.21	**0.78** $\pm$ 0.09
	ProgPrompt	**0.60** $\pm$ 0.49	0.96 $\pm$ 0.06	**0.70** $\pm$ 0.24	0.20 $\pm$ 0.40	**0.97** $\pm$ 0.06	0.65 $\pm$ 0.25	0.20 $\pm$ 0.40	**0.98** $\pm$ 0.05	0.50 $\pm$ 0.38	0.00 $\pm$ 0.00	**0.94** $\pm$ 0.07	0.17 $\pm$ 0.21
Bathroom	Our Method	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00	**0.60** $\pm$ 0.49	**1.00** $\pm$ 0.00	**0.90** $\pm$ 0.20	**0.40** $\pm$ 0.49	**1.00** $\pm$ 0.00	**0.90** $\pm$ 0.09	**0.40** $\pm$ 0.49	**0.96** $\pm$ 0.08	**0.57** $\pm$ 0.37
	ProgPrompt	0.40 $\pm$ 0.49	0.89 $\pm$ 0.20	0.50 $\pm$ 0.45	0.20 $\pm$ 0.40	0.93 $\pm$ 0.09	0.45 $\pm$ 0.33	0.00 $\pm$ 0.00	0.89 $\pm$ 0.10	0.35 $\pm$ 0.24	0.00 $\pm$ 0.00	0.81 $\pm$ 0.11	0.20 $\pm$ 0.15
Bedroom	Our Method	**0.80** $\pm$ 0.40	0.90 $\pm$ 0.20	**0.90** $\pm$ 0.20	**0.80** $\pm$ 0.40	**0.96** $\pm$ 0.08	**1.00** $\pm$ 0.00	0.40 $\pm$ 0.49	**0.96** $\pm$ 0.05	**0.93** $\pm$ 0.06	**0.60** $\pm$ 0.49	**0.98** $\pm$ 0.04	**0.65** $\pm$ 0.43
	ProgPrompt	0.40 $\pm$ 0.49	**0.91** $\pm$ 0.11	0.60 $\pm$ 0.27	0.20 $\pm$ 0.40	0.82 $\pm$ 0.30	0.35 $\pm$ 0.37	0.00 $\pm$ 0.00	**0.96** $\pm$ 0.06	0.25 $\pm$ 0.31	0.00 $\pm$ 0.00	0.94 $\pm$ 0.08	0.22 $\pm$ 0.27

TABLE III: Results on the room arrangement task (full table).