SayPlan: Grounding Large Language Models using 3D Scene Graphs for Scalable Task Planning

Anonymous Author(s)
Affiliation
Address
email

Abstract:
Large language models (LLMs) have demonstrated impressive results in developing generalist planning agents for diverse tasks. However, grounding these plans in expansive, multi-floor, and multi-room environments presents a significant challenge for robotics. We introduce SayPlan, a scalable approach to LLM-based, large-scale task planning for robotics using 3D scene graph (3DSG) representations. To ensure the scalability of our approach, we: (1) exploit the hierarchical nature of 3DSGs to allow LLMs to conduct a *semantic search* for task-relevant subgraphs from a smaller, collapsed representation of the full graph; (2) reduce the planning horizon for the LLM by integrating a classical path planner and (3) introduce an *iterative replanning* pipeline that refines the initial plan using feedback from the scene graph simulator, correcting infeasible actions and avoiding planning failures. We evaluate our approach on two large-scale environments spanning up to 3 floors, 36 rooms and 140 objects, and show that our approach is capable of grounding large-scale, long-horizon task plans from abstract, and natural language instruction for a real-world mobile manipulator robot to execute. We provide anonymised video and code on our project page sayplan.github.io

Keywords: robot task planning, large language models, semantic search, 3D Scene Graphs

1 Introduction

"Make me a coffee and place it on my desk" – The successful execution of such a seemingly straightforward command remains a daunting task for today's robots. The associated challenges permeate every aspect of robotics, encompassing navigation, perception, manipulation as well as high-level task planning. Recent advances in Large Language Models (LLMs) [1, 2, 3] have led to significant progress in incorporating common sense knowledge for robotics [4, 5, 6]. This enables robots to plan complex strategies for ambiguously articulated tasks that require a substantial amount of background knowledge and semantic comprehension.

For LLMs to be effective planners in robotics, they must be grounded in reality, that is, they must adhere to the constraints presented by the physical environment in which the robot operates, including the available affordances, relevant predicates, and the impact of actions on the current state. Furthermore, in expansive environments, the robot must additionally understand where it is, locate items of interest, as well comprehend the topological arrangement of the environment in order to plan across the necessary regions. To address this, recent works have explored the utilization of vision-based value functions [4], object detectors [7, 8], or Planning Domain Definition Language (PDDL) descriptions of a scene [9, 10] to ground the output of the LLM-based planner. However, these efforts are primarily confined to small-scale environments, typically single rooms with pre-encoded information on all the existing assets and objects present. The challenge lies in scaling these models. As the environment's complexity and dimensions expand, and as more rooms and entities enter the

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

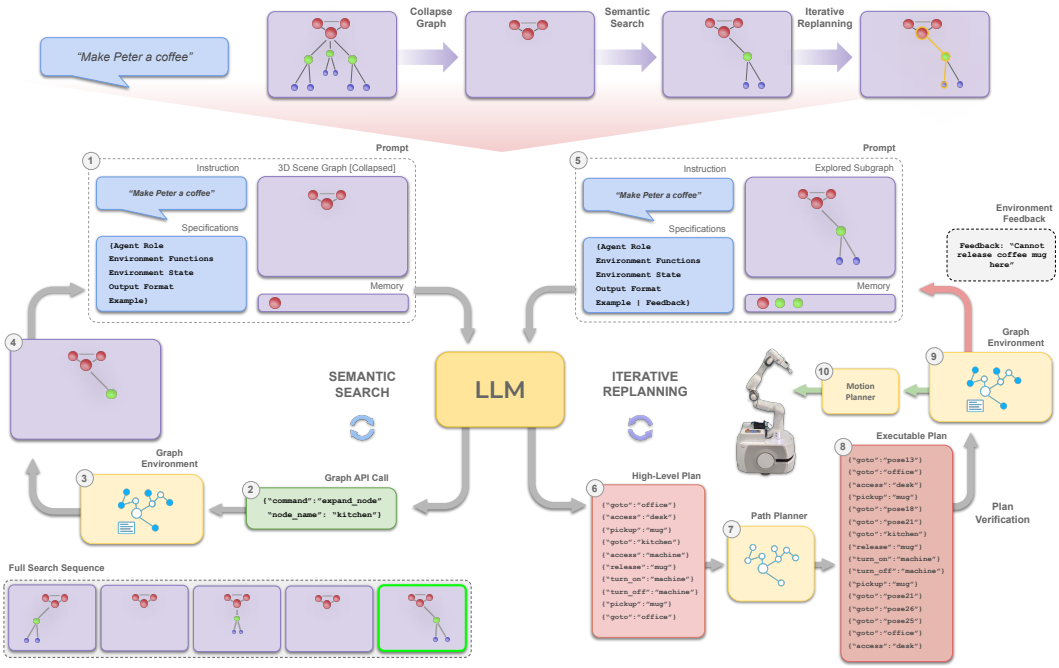

Figure 1: **SayPlan Overview.** SayPlan operates across two stages to ensure scalability: (left) Given a collapsed 3D scene graph and a task instruction, *semantic search* is conducted by the LLM to identify a suitable subgraph that contains the desired items for the task; (right) The explored subgraph is then used by the LLM to generate a high-level task plan, where a classical path planner completes the navigational component of the plan; finally, the plan goes through an *iterative replanning* process with feedback from a scene graph simulator until an executable plan is identified. Numbers at the corners of the modules represent an example order of operation.

scene, pre-encoding all the necessary information within the LLMs context becomes increasingly infeasible.

To this end, we present a scalable approach to ground LLM-based task planners across environments spanning multiple rooms and floors. We achieve this by exploiting the growing body of 3D scene graph (3DSGs) research [11, 12, 13, 14, 15, 16]. 3DSGs capture a rich topological and hierarchically-organised semantic graph representation of an environment with the versatility to encode the necessary information required for task planning including object state, predicates, affordances and attributes using natural language – suitable for parsing by an LLM. We can leverage a JSON representation of this graph as input to a pre-trained LLM, however, to ensure the *scalability* of the plans to larger scenes, we present three key innovations.

Firstly, we present a mechanism that enables the LLM to conduct a *semantic search* for a task-relevant subgraph $\mathcal{G}'$ by manipulating the nodes of a 'collapsed' graph $\mathcal{G}$ through `expand` and `contract` API function calls – thus making it feasible to plan over increasingly large-scale environments. In doing so, the LLM maintains focus on the small, informative subgraph, $\mathcal{G}'$ during planning, without exceeding its token limit. Secondly, as the horizon of the task plans across such environments tend to grow with the complexity and range, there is an increasing tendency for LLMs to hallucinate or produce infeasible action sequences [17, 18, 7]. We counter this by relaxing the need for the LLM to generate the navigational component of the plan, and instead leverage an existing optimal path planner such as Dijkstra [19] to connect high-level nodes generated by the LLM. Finally, to ensure the feasibility of the proposed plan, we introduce an *iterative replanning* pipeline that verifies and refines the initial plan using feedback from a scene graph simulator in order to correct for any unexecutable actions, e.g., missing to open the fridge before putting something into it – thus avoiding planning failures due to inconsistencies, hallucinations, or violations of the physical constraints imposed by the environment.

Our approach SayPlan ensures feasible and grounded plan generation for a mobile manipulator robot operating in large-scale environments spanning multiple floors and rooms. We evaluate our framework across a range of 90 tasks organised into four levels of difficulty. These include semantic search tasks such as (*"Find me something non-vegetarian."*) to interactive, long-horizon tasks with

ambiguous multi-room objectives that require a significant level of common-sense reasoning (*"Let's play a prank on Niko"*). These tasks are assessed in two expansive environments, including a large office floor spanning 36 rooms and 150 interactable assets and objects, and a three-storey house with 32 rooms and 121 objects. Our experiments validate SayPlan's ability to scale task planning to large-scale environments while conserving a low token footprint. By introducing a semantic search pipeline, we can reduce full large-scale scene representations by up to 82.1% for LLM parsing and our iterative replanning pipeline allows for near-perfect executability rates, suitable for execution on a real mobile manipulator robot.[1]

2 Related Work

Task planning in robotics aims to generate a series of actions to achieve a goal within an environment. Conventional methods employ domain-specific languages such as PDDL [20, 21, 22] and ASP [23] together with search techniques [24, 25] and complex heuristics [26] to arrive at a solution. The requirement to specify tasks and the environment via these languages as well as the need for such complex heuristics limits the versatility of the approach when scaling to larger environments and more complex tasks. Learning-based alternatives [27, 28], hierarchical and reinforcement learning [29], face challenges with data demands and scalability. Our work leverages the in-context learning capabilities of LLMs to generate feasible long-horizon task plans across 3D scene graphs, a scalable approach to representing large-scale scenes. Tasks, in this case, can be naturally expressed using natural language, and search heuristics are derived by the LLM based on its internet scale knowledge, the task's semantics and the information within the scene graph environment.

Task planning with LLMs, that is, translating natural language prompts into task plans for robotics, is an emergent trend in the field. Earlier studies have effectively leveraged pre-trained LLMs' in-context learning abilities to generate actionable plans for embodied agents [4, 10, 9, 8, 30, 7, 31]. These plans have been further grounded using feedback from object detectors [8, 7] or value functions [4]. However, these methods are predominantly constrained to single-room environments and either scale poorly with the number of objects [4] or lack a full environment-level prior [7, 4]. Another line of work explored the use of PDDL with LLMs [10, 9, 32] for planning, thus being inherently prone to the limitations of planning using such declarative languages [33]. We instead represent a large-scale environment through a hierarchical 3D scene graph, which can be directly represented as text-serialised JSON, while enabling subgraph exploration for token-efficient search of task-relevant nodes.

Integrating external knowledge in LLMs has been a growing line of research combining language models with external tools to improve the reliability of their outputs. In such cases, external modules are used to provide feedback or extra information to the LLM to guide its output generation. This is achieved either through API calls to external tools (e.g., LLM-Augmenter [34] and Toolformer [35]) or as textual feedback from the operating environment (e.g., Mind's eye [36] and Inner Monologue [8]). As an iterative feedback approach, CLAIRIFY [37] used compiler error feedback to prompt and verify code, thus ensuring syntactically valid task plans in robot programming. Building on these ideas, we propose an iterative plan verification process with feedback from a scene graph-based simulator to ensure all generated plans adhere to the constraints and specific states captured by the pre-constructed scene graph. This ensures the direct executability of the plan on a mobile manipulator robot, operating in the corresponding real-world environment.

3 SayPlan

3.1 Problem Formulation

We aim to address the challenge of long-range planning for an autonomous agent, such as a mobile manipulator robot, in a large-scale environment based on natural language instructions. This requires the robot to comprehend abstract and ambiguous instructions, understand the scene and generate task plans involving both navigation and manipulation of a mobile robot within an environment. Existing approaches lack the ability to reason over scenes spanning multiple floors and rooms. Our focus is on integrating large-scale scenes into planning agents based on Language Models (LLMs) and solving

[1] sayplan.github.io

Algorithm 1: SayPlan

Given: scene graph simulator ψ, classical path planner ϕ, large language model LLM
Inputs: prompt $\mathcal{P}$, scene graph $\mathcal{G}$, instruction $\mathcal{I}$

$\mathcal{G}' \leftarrow \texttt{collapse}_{\psi}(\mathcal{G})$ ▷ collapse scene graph
Stage 1: Semantic Search ▷ search scene graph for all relevant items
while command != *terminate* **do**
  command $\leftarrow LLM(\mathcal{P}, \mathcal{G}', \mathcal{I})$
  if command == 'expand' **then**
    $\mathcal{G}' \leftarrow \texttt{expand}_{\psi}$(node_id) ▷ expand node to reveal objects and assets
  else if command == 'contract' **then**
    $\mathcal{G}' \leftarrow \texttt{contract}_{\psi}$(node_id) ▷ contract node if nothing relevant found
Stage 2: Causal Planning ▷ generate a feasible plan
while feedback != *success* **do**
  plan $\leftarrow LLM(\mathcal{P}, \mathcal{G}', I, \text{feedback})$ ▷ high level plan
  full_plan $\leftarrow \phi(\text{plan}, \mathcal{G}')$ ▷ compute optimal path between nodes
  feedback $\leftarrow \texttt{verify_plan}_{\psi}$(full_plan) ▷ forward simulate the full plan
return full_plan ▷ executable plan

the scalability challenge. We aim to tackle two key problems: 1) representing large-scale scenes within LLM token limitations, and 2) mitigating LLM hallucinations and erroneous outputs when generating long-horizon plans in large-scale environments.

3.2 Preliminaries

Here, we describe the 3D scene graph representation of an environment and the components of scene graph API which we leverage throughout our approach.

Scene Representation: The 3D Scene Graph (3DSG) [11, 12, 14] has recently emerged as an actionable world representation for robots [13, 15, 16, 38, 39, 40], which hierarchically abstracts the environment at multiple levels through spatial semantics, object relationships, affordances, attributes, as well as 3D pose information. Formally, a 3DSG is a hierarchical multigraph $G = (V, E)$ in which the set of vertices V comprises $V_1 \cup V_2 \cup \ldots \cup V_K$, with each V_k signifying the set of vertices at a particular level of the hierarchy k. Edges stemming from a vertex $v \in V_k$ may only terminate in $V_{k-1} \cup V_k \cup V_{k+1}$, i.e. edges connect nodes within the same level, or one level higher or lower. We assume a pre-constructed 3D scene graph representation of a large-scale environment generated using existing techniques [15, 13, 11]. The entire 3D scene graph can be represented as a NetworkX `Graph` object [41] and text-serialised into a JSON data format which can be parsed directly by a pre-trained LLM. An example of a single asset node from the 3D scene graph is represented as: {`name: coffee_machine, type: asset, location: kitchen, affordances: [turn_on, turn_off, release], state: off, attributes: "red", position: [2.34, 0.45, 2.23]`}. The 3D Scene Graph (3DSG) is organized in a hierarchical manner with four primary layers: floors, rooms, assets, and objects. The top layer contains floors, each of which branches out to several rooms. These rooms are interconnected through pose nodes to represent the environment's topological structure. Within each room, we find assets (immovable entities) and objects (movable entities). Both asset and object nodes encode particulars such as state, affordances (possible interactions), attributes (like color or weight), and 3D pose. The graph also incorporates a dynamic agent node, denoting a robot's location within the scene.

Scene Graph API: The LLM is given access to an external API which provides it with a set of tools required to manipulate and operate over 3DSGs. It enables the LLM to manipulate scene graphs through `expand` and `contract` functions, revealing connected nodes in a lower layer, or reversing the process respectively. Furthermore, generated plans can be verified through a task-agnostic `scene graph simulator` which consists of a set of rules which verify if actions performed on the nodes adhere to the physical constraints, predicates and affordances present in the corresponding environment.

3.3 Approach

Using 3D scene graph representations $\mathcal{G}$ and by defining a task instruction $\mathcal{I}$ in natural language, we can view our framework SayPlan as a high-level task planner $\pi(\boldsymbol{a}|\mathcal{I}, \mathcal{G})$, capable of generating long-

horizon plans $\boldsymbol{a}$ grounded in the large scale environment within which a mobile manipulator robot operates. The plan can then be fed to a low-level visually grounded motion planner for real-world execution. An overview of the SayPlan pipeline is illustrated in Figure 1 and the corresponding pseudo-code is given in Algorithm 1.

We address the challenges that arise when planning across these large-scale scenes by decomposing the planning pipeline into two key stages: *semantic search* and *iterative replanning*. During semantic search, the LLM explores from a collapsed representation of the full scene graph for a suitable *subgraph* $\mathcal{G}'$ that contains the necessary items required to fulfil the task. This relaxes the need for providing the entire scene to the LLM, which would typically exceed the token limits of current models. Once the desired subgraph is identified, the LLM switches to the iterative planning phase. Here, we reduce the planning horizon for an LLM by integrating a classical path planner to fill in optimal navigation paths within the high-level plan generated by the LLM. To ensure the plan adheres to the specific constraints imposed by the environment, it iteratively goes through a plan verification stage where it is executed within a scene graph environment. If the verification fails, a feedback message is appended to the output plan and fed back to the LLM for replanning. This process is repeated until an executable plan is achieved and can be passed to a visually grounded motion planner for execution in the real-world environment. We provide more detail on each component in the following sections.

Semantic Graph Search: The semantic search phase begins with a collapsed representation of the full 3D scene graph $\mathcal{G}$, exposing only the highest level of the hierarchy to the LLM. Given a natural language task description $\mathcal{I}$, the goal of this phase is to conduct a search, for a sub-graph $\mathcal{G}'$ which contains all the nodes necessary for solving the task. The search is governed by the LLM's common sense reasoning capabilities and in-context learning from a set of input-output examples [42, 43]. We leverage Chain-of-Thought (CoT) [18] reasoning to help the LLM decompose complex tasks into intermediate steps to facilitate its ability to decide on the appropriate nodes to `expand` or `contract` using the available API calls. At each step, the subgraph $\mathcal{G}'$ in the LLM's previous input is updated and passed again to the LLM until a suitable $\mathcal{G}'$ is identified. The ability to `contract` nodes not required for solving the task reduces the token footprint over the course of long search sequences (see Fig. 2). To avoid expanding already-contracted nodes, we maintain a list of expanded nodes, passed as an additional *memory* input to the LLM. This leads to a fully Markovian decision-making process, where the current subgraph $\mathcal{G}'$ and the history of expanded nodes are the only state inputs required for the LLM to make its next decision. This allows it to scale to long search sequences, unlike [5] which has to maintain the full history of interactions. Once the LLM agent identifies that the current subgraph has visibility over all the assets and objects required to solve the task, it autonomously switches to the planning phase. An example of the LLM-scene graph interaction during semantic search is provided in Appendix I.

Iterative Re-planning: Given the subgraph $\mathcal{G}'$, we generate correct and feasible long-horizon task plans, via two key mechanisms. First, we shorten the LLM's planning horizon by delegating pose-level path planning to an optimal path planner, such as Dijkstra. For example, a typical plan output such as `[goto(meeting_room), goto(pose13), goto(pose14), goto(pose8), ..., goto(kitchen), access(fridge), open(fridge)]` is simplified to `[goto(meeting_room), goto(kitchen), access(fridge), open(fridge)]`. The path planner handles finding the optimal route between high-level locations, allowing the LLM to focus on essential manipulation components of the task. Secondly, we utilise the `scene graph simulator` to evaluate if the generated plan complies with the scene graph's predicates, state, and affordances. For instance, a `pick(banana)` action might fail if the robot is already holding something, if it is not in the correct location or if the fridge was not opened beforehand. Such failures are transformed into textual feedback (e.g., *"cannot pick banana"*), appended to the LLM's query, and used to generate an updated, executable plan. This iterative process, involving planning, validation, and feedback integration, continues until a feasible plan is obtained. This plan is then passed to a low-level motion planner for robotic execution. An example of the LLM-scene graph interaction during iterative re-planning is provided in Appendix J.

Implementation Details: We utilise `GPT-4` [3] as the underlying LLM agent unless otherwise stated. We follow a similar prompting structure to Wake et al. [5] as shown in Appendix H. We define the agent's role, details pertaining to the scene graph environment, the desired output structure and a set of input-output examples which together form the static prompt used for in-context learning. This static prompt is both task- and environment-agnostic and takes up approximately 3900 tokens

Instruction Family	Num	Explanation	Example Instruction
		Semantic Search	
Simple Search	30	Queries focussed on evaluating the basic semantic search capabilities of SayPlan	Find me a ripe banana.
Complex Search	30	Abstract semantic search queries which require complex reasoning	Find the room where people are playing board games.
		Causal Planning	
Simple Planning	15	Queries which require the agent to perform search, causal reasoning and environment interaction in order to solve a task.	Refrigerate the orange left on the kitchen bench.
Long-Horizon Planning	15	Long Horizon planning queries requiring multiple interactive steps	Tobi spilt soda on his desk. Help him clean up.

Table 1: **List of evaluation task instructions.** We evaluate SayPlan on 90 instructions, grouped to test various aspects of the planning capabilities across large-scale scene graphs. The full instruction set given in Appendix B.

of the LLMs input. During semantic search, both the *3D Scene Graph* and *Memory* components of the input prompt get updated, while during iterative planning only the *Feedback* component gets updated with information from the `scene graph simulator`.

4 Experimental Setup

We design our experiments to evaluate the 3D scene graph reasoning capabilities of LLMs with a particular focus on high-level task planning pertaining to a mobile manipulator robot. The plans adhere to a particular embodiment consisting of a 7-degree-of-freedom robot arm with a two-fingered gripper attached to a mobile base. We use two large-scale environments, shown in Figure 3, which exhibit multiple rooms and multiple floors which the LLM agent has to plan across. To better ablate and showcase the capabilities of SayPlan, we decouple its semantic search ability from the overall causal planning capabilities using the following two evaluation settings:

Semantic Search: Here, we focus on queries which test the semantic search capabilities of an LLM provided with a collapsed 3D scene graph. This requires the LLM to reason over the room and floor nodes and their corresponding attributes in order to aid its search for the relevant assets and objects required to solve the given task instruction. We evaluate against a human baseline to understand how the semantic search capabilities of an LLM compare to a human's thought process. Furthermore, to gain a better understanding of the impact different models have on this graph-based reasoning, we additionally compare against a variant of SayPlan using GPT-3.5.

Causal Planning: In this experiment, we evaluate the ability of SayPlan to generate feasible plans to solve a given natural language instruction. The evaluation metrics are divided into two components: *1) Correctness*, which primarily validates the overall goal of the plan and its alignment to what a human would do to solve the task and *2) Executability*, which evaluates the alignment of the plan to the constraints of the scene graph environment and its ability to be executed by a mobile manipulator robot. We note here that for a plan to be executable, it does not necessarily have to be correct and vice versa. We evaluate SayPlan against two baseline methods that integrate an LLM for task planning: **LLM-As-Planner**, which generates a full plan sequence in an open-loop manner; the plan includes the full sequence of both navigation and manipulation actions that the robot must execute to complete a task, and **LLM+P**, an ablated variant of SayPlan, which only incorporates the path planner to allow for shorter horizon plan sequences, without any iterative re-planning.

	Office			Home		
Subtask	Human	SayPlan (GPT-3.5)	SayPlan (GPT-4)	Human	SayPlan (GPT-3.5)	SayPlan (GPT-4)
Simple Search	100%	6.6%	86.7%	100%	0.0%	86.7%
Complex Search	100%	0.0%	73.3%	100%	0.0%	73.3%

Table 2: **Evaluating the semantic search capabilities of GPT-4.** The table shows the average number of exploratory steps taken in the environment before the instruction was satisfied.

5 Results

We summarise the results for the semantic search evaluation in Table 2. SayPlan (GPT-3.5) consistently failed to reason over the input graph representation, hallucinating nodes to explore or stagnating at exploring the same node multiple times. SayPlan(GPT-4) in contrast achieved 86.7% and 73.3% success in identifying the desired subgraph across both the simple and complex search tasks respectively, demonstrating significantly better graph-based reasoning than GPT-3.5.

While as expected the human baseline achieved 100% on all sets of instructions, we are more interested in the qualitative assessment of the common-sense reasoning used during semantic search. More specifically we would like to identify the similarity in the semantic search heuristics utilised by humans and that used by the underlying LLM based on the given task instruction.

5.1 Semantic Search

We present the full sequence of explored nodes for both SayPlan (GPT-4) and the human baseline in Appendix E. As shown in the tables, SayPlan (GPT-4) demonstrates remarkably similar performance to a human's commonsense reasoning for most tasks, exploring a similar sequence of nodes given a particular instruction. For example when asked to "find a ripe banana", the LLM first explores the kitchen followed by the next most likley location, the cafeteria. In the case where no semantics are present in the instruction such as "find me object K31X", we note that the LLM agent is capable of conducting a breadth-first-like search across all the unexplored nodes.

An odd failure case in the simple search instructions involved negation, where the agent consistently failed when presented with questions such as "Find me an office that does not have a cabinet" or "Find me a bathroom with no toilet". Other failure cases noted across the complex search instructions included the LLM's failure to conduct simple distance-based and count-based reasoning over graph nodes. While trivial to a human, this does require the LLM agent to reason over multiple nodes simultaneously, where it tended to hallucinate or miscount connected nodes.

	Full Graph (Token Count)	Collapsed Graph (Token Count)	Compression Ratio
Office	4962	888	82.1%
Home	4602	1827	60.4%

Table 3: **3D Scene Graph Token Count** Number of tokens required for the full graph vs. collapsed graph.

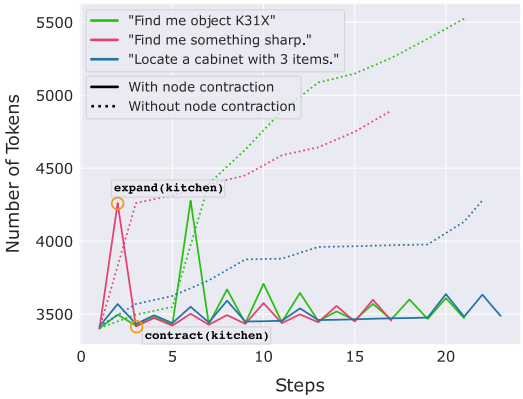

Figure 2: **Scene Graph Token Progression During Semantic Search.** This graph illustrates the scalability of our approach to large-scale 3D scene graphs

Scalability Analysis: We additionally analyse the scalability of SayPlan during semantic search. Table 3 illustrates the impact of exploiting the hierarchical nature of 3D scene graphs and allowing the LLM to explore the graph from a collapsed initial state. This allows for a reduction of 82.1% in the input tokens required to represent the Office environment and a 60.4% reduction for the Home environment. In Figure 2, we illustrate how endowing the LLM with the ability to contract explored nodes which it deems unsuitable for solving the task allows it to maintain near-constant input memory from a token perspective across the entire semantic search process. Note that the initial number of tokens already present represents the input prompt tokens described in Section 3.3.

5.2 Causal Planning

The results of causal planning across simple and long-horizon instructions (Table 4, left) reveal the effectiveness of SayPlan. We compared its performance with two baselines: LLM-As-Planner and LLM+P. All three methods displayed consistent correctness in simple planning tasks (93%).

	Simple		Long Horizon		Types of Errors				
	Corr	**Exec**	**Corr**	**Exec**	**Missing Action**	**Missing Pose**	**Wrong Action**	**Incomplete Search**	**Hallucinated Nodes**
LLM+P	93.3%	13.3%	33.3%	0.0%	26.7%	0.0%	10.0%	3.33%	10.0%
LLM-As-Planner	93.3%	80.0%	66.7%	13.3%	20.0%	60.0%	0.17%	0.03%	10.0%
SayPlan	93.3%	100.0%	73.3%	86.6%	0.0%	0.0%	0.0%	0.0%	6.67%

Table 4: **Causal Planning Results.** *Left:* Correctness and Executability on Simple and Long Horizon planning tasks and *Right:* Types of execution errors encountered when planning using LLMs. Note that SayPlan corrects the majority of the errors faced by LLM-based planners.

However, in long-horizon tasks, LLM-As-Planner had a significant performance drop to 33.3%. LLM+P, which incorporates a path planner, performed better at 66.7%, highlighting the planner's importance. SayPlan, benefitting from both a path planner and iterative re-planning, achieved the highest correctness rate (73.3%). The key to ensuring plan executability is iterative re-planning. Both LLM-As-Planner and LLM+P showed poor executability, whereas SayPlan achieved near-perfect executability as a result of iterative re-planning, ensuring plans are grounded to match the environment's feasibility.

Detailed task plans and errors encountered are provided in Appendix F. We summarise these errors in Table 4 (right) which shows that plans generated with LLM+P and LLM-As-Planner entailed various types of errors limiting their executability. LLM+P mitigated path planning errors as a result of the classical path planner however still suffered from errors pertaining to the manipulation of the environment. SayPlan mitigated these errors via iterative replanning, however in 6.67% of tasks, it failed to correct for some hallucinated nodes. While we believe these errors could be eventually corrected via iterative replanning, we limited the number of replanning steps to 5 throughout all experiments. We provide an illustration of the real-world execution of a generated plan using SayPlan on a mobile manipulator robot coupled with a vision-guided motion planner in Appendix G.

6 Limitations

Firstly, SayPlan is expected to inherit the limitations and biases inherent in current large language models, which are dependent on their training data, and can adversely affect the correctness of the generated plans causing misinterpretation of instructions; examples of such failures are illustrated in Appendix F. Secondly, SayPlan's current framework is constrained by the need for a pre-built 3D scene graph and assumes that objects remain static post-map generation, significantly restricting its adaptability to dynamic real-world environments. Lastly, SayPlan operates on an open-loop principle, naively expecting generated plans to be flawlessly executed by a robot without accommodating the practical necessity of online refinements through visual feedback. Integrating a closed-loop component that allows for plan adjustments in response to real-time feedback is a critical aspect that future work could explore.

7 Conclusion

SayPlan is a natural language-driven planning framework for robotics that integrates hierarchical 3D scene graphs and LLMs to plan across large-scale environments spanning multiple floors and rooms. We ensure scalability of our approach by exploiting the hierarchical nature of 3D scene graphs and the semantic reasoning capabilities of LLMs to allow the agent to explore the scene graph from the highest level within the hierarchy. In this way, we are no longer required to present all lower-level entities to the LLM, resulting in a significant reduction in the tokens required to capture larger environments. Once explored, the LLM generates task plans for an mobile manipulator robot, and we ensure the plan is feasible and grounded to the state of the environment via iterative replanning using a scene graph simulator. The framework produces the highest number of correct and executable plans that a mobile robot can follow, in comparison to existing baseline techniques. Finally, we translate a select number of the generated plans to a real-world mobile manipulator agent, capable of completing natural language-defined tasks spanning long-range, long-horizon across a wide range of rooms.

References

[1] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. E. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. J. Lowe. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155, 2022.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[3] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[4] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, pages 287–318. PMLR, 2023.

[5] N. Wake, A. Kanehira, K. Sasabuchi, J. Takamatsu, and K. Ikeuchi. Chatgpt empowered long-step robot control in various environments: A case application, 2023.

[6] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence. Palm-e: An embodied multimodal language model, 2023.

[7] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. *arXiv preprint arXiv:2212.04088*, 2022.

[8] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[9] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

[10] T. Silver, V. Hariprasad, R. S. Shuttleworth, N. Kumar, T. Lozano-Pérez, and L. P. Kaelbling. Pddl planning with pretrained large language models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

[11] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5664–5673, 2019.

[12] U.-H. Kim, J.-M. Park, T.-J. Song, and J.-H. Kim. 3-d scene graph: A sparse and semantic representation of physical environments for intelligent agents. *IEEE transactions on cybernetics*, 50(12):4921–4933, 2019.

[13] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone. Kimera: From slam to spatial perception with 3d dynamic scene graphs. *The International Journal of Robotics Research*, 40(12-14):1510–1546, 2021.

[14] P. Gay, J. Stuart, and A. Del Bue. Visual graphs from motion (vgfm): Scene understanding with object geometry reasoning. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*, pages 330–346. Springer, 2019.

[15] N. Hughes, Y. Chang, and L. Carlone. Hydra: A real-time spatial perception engine for 3d scene graph construction and optimization. *Robotics: Science and Systems XIV*, 2022.

[16] C. Agia, K. M. Jatavallabhula, M. Khodeir, O. Miksik, V. Vineet, M. Mukadam, L. Paull, and F. Shkurti. Taskography: Evaluating robot task planning over large 3d scene graphs. In *Conference on Robot Learning*, pages 46–58. PMLR, 2022.

[17] N. Shinn, F. Cassano, B. Labash, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.

[18] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

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

[20] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. Pddl-the planning domain definition language. 1998.

[21] M. Fox and D. Long. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*, 20:61–124, 2003.

[22] P. Haslum, N. Lipovetzky, D. Magazzeni, and C. Muise. An introduction to the planning domain definition language. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(2):1–187, 2019.

[23] M. Gelfond and Y. Kahl. *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach*. Cambridge University Press, 2014.

[24] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the national conference on artificial intelligence*, pages 1194–1201, 1996.

[25] B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1-2):5–33, 2001.

[26] M. Vallati, L. Chrpa, M. Grześ, T. L. McCluskey, M. Roberts, S. Sanner, et al. The 2014 international planning competition: Progress and trends. *Ai Magazine*, 36(3):90–98, 2015.

[27] R. Chitnis, T. Silver, B. Kim, L. Kaelbling, and T. Lozano-Perez. Camps: Learning context-specific abstractions for efficient planning in factored mdps. In *Conference on Robot Learning*, pages 64–79. PMLR, 2021.

[28] T. Silver, R. Chitnis, A. Curtis, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling. Planning with learned object importance in large problem instances using graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11962–11971, 2021.

[29] F. Ceola, E. Tosello, L. Tagliapietra, G. Nicola, and S. Ghidoni. Robot task planning via deep reinforcement learning: a tabletop object sorting application. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 486–492, 2019. doi:10.1109/SMC.2019.8914278.

[30] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[31] A. Zeng, A. Wong, S. Welker, K. Choromanski, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*, 2022.

[32] Y. Xie, C. Yu, T. Zhu, J. Bai, Z. Gong, and H. Soh. Translating natural language to planning goals with large-language models. *arXiv preprint arXiv:2302.05128*, 2023.

[33] Y.-q. Jiang, S.-q. Zhang, P. Khandelwal, and P. Stone. Task planning in robotics: an empirical comparison of pddl-and asp-based systems. *Frontiers of Information Technology & Electronic Engineering*, 20:363–373, 2019.

[34] B. Peng, M. Galley, P. He, H. Cheng, Y. Xie, Y. Hu, Q. Huang, L. Liden, Z. Yu, W. Chen, et al. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*, 2023.

[35] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

[36] R. Liu, J. Wei, S. S. Gu, T.-Y. Wu, S. Vosoughi, C. Cui, D. Zhou, and A. M. Dai. Mind's eye: Grounded language model reasoning through simulation. *arXiv preprint arXiv:2210.05359*, 2022.

[37] M. Skreta, N. Yoshikawa, S. Arellano-Rubach, Z. Ji, L. B. Kristensen, K. Darvish, A. Aspuru-Guzik, F. Shkurti, and A. Garg. Errors are useful prompts: Instruction guided task programming with verifier-assisted iterative prompting. *arXiv preprint arXiv:2303.14100*, 2023.

[38] Z. Ravichandran, L. Peng, N. Hughes, J. D. Griffith, and L. Carlone. Hierarchical representations and explicit memory: Learning effective navigation policies on 3d scene graphs using graph neural networks. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9272–9279. IEEE, 2022.

[39] A. Kurenkov, R. Martín-Martín, J. Ichnowski, K. Goldberg, and S. Savarese. Semantic and geometric modeling with neural message passing in 3d scene graphs for hierarchical mechanical search. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11227–11233. IEEE, 2021.

[40] S. Garg, N. Sünderhauf, F. Dayoub, D. Morrison, A. Cosgun, G. Carneiro, Q. Wu, T.-J. Chin, I. Reid, S. Gould, et al. Semantics for robotic mapping, perception and interaction: A survey. *Foundations and Trends® in Robotics*, 8(1–2):1–224, 2020.

[41] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught, and J. Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[42] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[43] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.