Semantically-Driven Object Search Using Partially Observed 3D Scene Graphs

Isaac Remy
Department of Aeronautics and Astronautics
University of Washington
iremy@uw.edu

Abhishek Gupta
School of Computer Science and Engineering
University of Washington
abhgupta@cs.washington.edu

Karen Leung
Department of Aeronautics and Astronautics
University of Washington
kymleung@uw.edu

Abstract

Object search is a fundamental task for service robots aiding humans in their daily lives. For example, a robot must locate a cup before pouring coffee, or locate a sponge before cleaning up a spill. As such, robots performing object search across many different and potentially unseen environments must reason about uncertainty in both environment layout and object location. In this work, we frame object search as a Partially Observable Markov Decision Process (POMDP), and propose a generalizable planner that combines the structured representations afforded by 3D scene graphs with the semantic knowledge of language models. Specifically, we introduce (i) 3DSG-POMDPs, which are POMDPs defined over 3D scene graphs that reduce the dimensionality of object search, and (ii) PROPHE-C, a sampling-based planner for solving 3DSG-POMDPS. We demonstrate the efficacy of PROPHE-C in a partially observable household environment, revealing that additional online inference leads to more efficient and exploratory search plans, compared to solely relying on language models for decision-making.

1 Introduction

Autonomous mobile service robots that assist humans in their everyday lives or work environments are required to autonomously navigate and execute their tasks in various, potentially unseen, environments. In many tasks, the robot is required to locate objects necessary to complete the task, e.g., locating a sponge to clean up a spill. However, since each deployment environment is different and objects' locations vary over time and are not rigorously tracked, the exact location of objects required to accomplish the task is unknown. As such, the problem of *object search* is a critical piece in the development of fully autonomous service robots.

The problem of object search can be framed as a Partially Observed Markov Decision Process (POMDP), a general modeling framework for sequential decision-making with partial observability. However, computing POMDP policies suffers from the curse of dimensionality unless problem-specific structure is exploited. Alternatively, learning-based methods such as reinforcement learning have demonstrated impressive performance, but are very data-hungry and prone to distribution shifts. In this work, we propose a scalable and generalizable approach to solving the object search problem by leveraging the power of *semantics*, both for the environment representation and the prior for an object's location.

Motivated by the massive success of language models and their potential in sequential decision-making, we investigate the effectiveness of using language models in aiding object search problems that are characterized by partial observability. Specifically, we leverage 3D scene graphs (3DSGs)

Foundation Models for Decision Making Workshop at NeurIPS 2023.

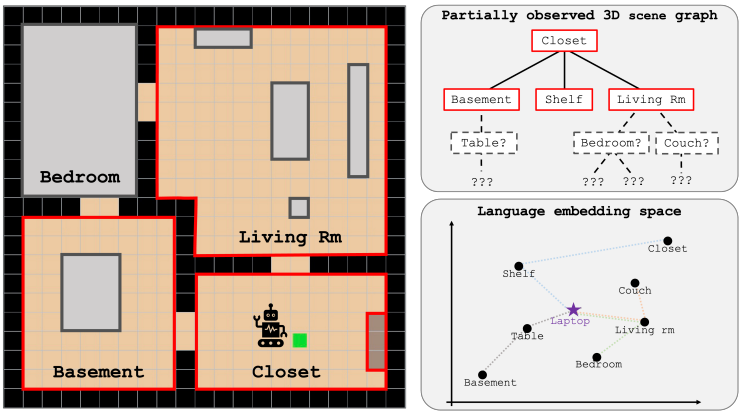


Figure 1: A partially observed household environment (left) and its corresponding 3D scene graph (top right). Our work explores performing search tasks over 3D scene graphs by harnessing the semantic richness of language models (bottom right).

because they ground an environment's semantic features with spatial information; we feed their nodes into a language model, which contains massive amounts of semantic knowledge, and are provided strong priors for potential object locations. Our experiments reveal that language models that are fed the 3DSG's nodes with no additional planning provide good guesses for where an object could be, but lack spatial awareness when searching for it, often resulting in unnecessarily long paths to find the object. On the other hand, by hallucinating future nodes in the 3DSG via domain-specific data, extracting priors from a language model, and then sampling for promising trajectories that weigh both travel cost and object location probabilities, our method harnesses both the "common-sense" reasoning present in language models and spatial information held within 3DSGs. This incentivizes a better balance of exploitative and exploratory behavior under partial observability.

Our contributions are three-fold: **(i)** We introduce 3DSG-POMDP, a POMDP defined over 3DSGs that is well-suited for general object search problems. **(ii)** We propose PROPHE-C (PRobabilistic Online Planner with Hierarchical Exploration Capabilities), a sampling-based online planner that combines priors obtained from a language model embedding space and online inference to calculate spatially efficient search plans. **(iii)** We evaluate our proposed approach against baseline methods and quantify the benefits of using language models of various capacities as priors.

2 Related Work

First, we briefly introduce 3D scene graphs and their use in sequential decision-making. Then, we discuss various approaches for robot planning problems across three broad categories: reinforcement learning, large language models (LLMs), and model-based methods.

Reasoning over 3D Scene Graphs. 3D scene graphs (3DSG) are directed graph representations that provide semantically-aware hierarchical relationships between objects/containers within a 3D environment [3, 19]. For example, house (building) $\rightarrow$ kitchen (room) $\rightarrow$ shelf (furniture) $\rightarrow$ cup (item). 3DSGs have become increasingly popular for decision-making problems due to their ability to condense a wealth of semantic information into a compact graph data structure that retains vital spatial information [1]. In particular, the semantic information and graph structure of 3DSGs makes it amenable for neural message-passing algorithms to determine object location likelihoods [13, 14], but these methods have been limited to fully-observed 3DSGs. Recent extensions consider the partially observed case, but the focus is on computing likelihoods of object location but not on policy construction [12]—while the likelihoods are known, there is still the question of how to find the object while striking a balance between exploration and exploitation.

Reinforcement learning for search. Reinforcement learning (RL) has been used widely to solve object search, or target-driven, problems [30, 5]. Typically a neural network policy is learned coupled with visual representation learning, and actions consist of moving the robot in a direction (i.e., left, right, forward, stop). The use of semantic map representations and 3DSGs has been shown to improve policy performance [5, 18]. While RL methods have succeeded in solving a number of search problems, they are incredibly data-hungry and often rely on high-fidelity simulators which are faced with sim-to-real transfer challenges. As such, pure RL-based solutions suffer from scalability challenges when deploying in diverse real-world environments. Moreover, in terms of high-level

decision-making with respect to choosing which room or object to search, the action space varies over time depending on what options are available. Unfortunately, RL with varying action spaces is still an open challenge.

Planning with large language models. Large language models (LLMs) have exploded over the last few years [6, 4, 23] and there has been an abundant body of work leveraging semantic understanding from LLMs to aid in robot planning and decision-making [9, 26, 27]. There is huge potential for LLMs to help robots operate in novel environments and reason beyond domain-specific data collection methods used previously. However, a significant limitation of LLMs is contextual grounding since LLMs lack a physical embodiment and spatial awareness of where and how they will be used by a robot. Very recently, LLMs have been combined with 3DSGs [17] which combines language and scene semantics to help robots more efficiently plan in 3D environments. While promising, the assumption is that the 3DSG is fully known, and their focus is directed at developing an efficient navigation planner in a 3D environment given a language command. In this work, we assume a partially known environment and the challenge is finding a robot navigation policy that balances between exploration and exploitation.

Belief-space planning. Planning under partial observability has been a widely studied problem in robotics, from autonomous exploration [16], human-robot interaction [15], and aircraft collision avoidance [11]. POMDPs are a popular modeling framework but suffer from the curse of dimensionality, and hence are limited to relatively small problems or for cases where problem-specific structure can be exploited. For example, [10] utilizes local and global information to construct a hierarchical value function to synthesize a policy for large-scale autonomous exploration in unknown environments, [7] utilizes hybrid belief states for task and motion planning, [24] proposes a novel object-oriented POMDP formulation to reduce the exponential growth of the belief space, and [29] propose a factored state-space POMDP framework where target object locations are correlated with more observable items. Another similar work is [2], which uses scene graphs for reasoning under partial observability but does not explicitly reason over the hierarchical relationships between nodes. While these methods have demonstrated impressive results in their respective applications, these techniques are tailored towards a specific problem formulation and are difficult to generalize or transfer to other domains. In this work, we aim to leverage two general-purpose tools, namely 3DSGs and language models, to formulate a general POMDP formulation amenable to problems that can be framed as object search in partially known environments. We must also note that PROPHE-C's design draws heavy inspiration from successful POMDP solution techniques such as POMCP [20], which uses Monte Carlo Tree Search to approximate a value function solution.

3 Problem Formulation

The problem of object search can be framed as a POMDP since the location of the target object is unknown. In Section 3.1 we briefly describe the canonical POMDP definition, and then describe our object search problem framed as a POMDP leveraging 3D scene graphs in Section 3.2.

3.1 POMDP Preliminaries

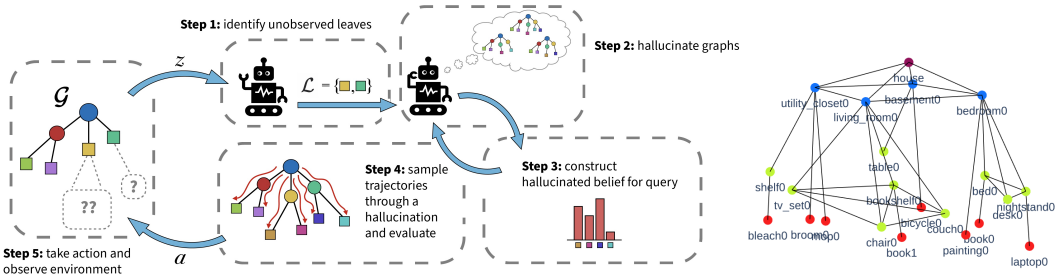
POMDPs are defined with a tuple: $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{Z}, \mathcal{O}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is a set of partially observable states, $\mathcal{A}$ is the set of actions, $\mathcal{T}$ is the state-action transition function where $\mathcal{T}_a(s, s') = p(s'|s, a)$ for $a \in \mathcal{A}$ and $s, s' \in \mathcal{S}$, and $\mathcal{Z}$ is the set of observations, $\mathcal{O}$ represents the observation-function $\mathcal{O}_a(s', z) = p(z|s, a)$ where $a \in \mathcal{A}, s, s' \in \mathcal{S}, z \in \mathcal{Z}$, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, and γ is a discount factor. The goal is to find a policy π (i.e., $a = \pi(s)$) that maximizes the expected discounted sum of rewards $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$. However, since the state is partially observed, a *belief* distribution b over states s is used since the true value of the state is unknown. The belief is updated after an observation is made using the following update rule,

$$b'(s') = \eta \mathcal{O}_a(s', z) \sum_{s \in \mathcal{S}} \mathcal{T}_a(s, s') b(s) \tag{1}$$

where η is the normalization constant. Unfortunately, given that we only have the belief and not the true state, solving for an optimal policy for a POMDP is generally computationally intractable since the size of the belief space grows exponentially with $|\mathcal{S}|$. Indeed, developing approximate solutions that scale tractable is an area of active research [21].

3.2 3DSG-POMDPs: POMDPs with 3D Scene Graphs

First we define a 3D scene graph and explain the appeal of this specific class of POMDPs, and then present our 3DSG-POMDP formulation for decision-making over 3D scene graphs with unknown goal locations.



(a) A schematic of the PROPHE-C algorithm. (b) 3D scene graph.

Figure 2: (a) Each step is contained within the dashed boxes. The set of leaves (squares) changes as we explore. We identify the leaves in the graph, hallucinate a full graph from those leaves, and sample for the most valuable leaf in that hallucination using a belief vector for that entire hallucinated graph, repeating this process many times. Then, the agent takes an action a whose expected value is highest. We reveal more nodes in the scene graph and repeat. (b) Example of a 3D scene graph on a household environment. The robot uses a 3D scene graph to reason about objects in its environment.

3.2.1 3D Scene Graphs

For ease of exposition, consider an indoor environment. The node-class layers of the 3D scene graph include, from highest to lowest: buildings, rooms, structures/furniture, and objects. For a 3D scene graph $\mathcal{G}$, the items of those classes comprise its nodes $\mathcal{V}$, and their relationships are described by edges $\mathcal{E}$. The members of $\mathcal{V}$ and $\mathcal{E}$ also have attributes; nodes have names, classes, spatial locations, and more, and edges encode semantic relationships (such as `inside` or `next to`). Since $\mathcal{G}$ is a tree, it also has a set of leaves, denoted by $\mathcal{L} \subseteq \mathcal{V}$.

There are four reasons we use 3DSGs for environment representation: **(i)** Defining our state and action spaces over the nodes in 3DSGs induces small state and action spaces. **(ii)** 3DSGs bind semantic information to states and actions, allowing easy value assessment of states and actions via language models. **(iii)** Their hierarchical structure enables strong contextual reasoning since child-nodes are always contextualized by their parents, which is useful when extracting priors over object locations from a language model. **(iv)** The nodes' spatial attributes capacitate path planning.

3.2.2 3DSG-POMDPs

Our justifications for defining a POMDP class specific to 3DSGs are two-fold: **(i)** In problems such as object search where agent states are known, goal locations are not, and hidden state transitions are deterministic, belief is only over potential goal locations. **(ii)** State and action spaces in a partially observed 3DSG are *dynamic*, as the agent will continuously reveal new actions and states; this means the belief size also fluctuates. We need a formalism that addresses and exploits these characteristics.

We define a 3DSG-POMDP with the tuple $\langle \mathcal{G}, \mathcal{T}, \mathcal{Z}, \mathcal{O}, \mathcal{R}, \gamma \rangle$. Let us first break down each term. $\mathcal{G}$ represents the environment graph represented as a scene graph as described in Section 3.2.1, with nodes $\mathcal{V}$ and edges $\mathcal{E}$. The environment graph $\mathcal{G}$ implicitly defines the states, actions, and object location of our problem. The nodes $\mathcal{V}$ describe the state and action space of the agent; the agent's state refers to which node s they are at, whereas taking an action $a \in \mathcal{V}$ corresponds to traveling to that node ($s' = a$). Notably, the state transition is assumed to be deterministic, i.e. $p(a|s, a) = 1$. For notational simplicity, we use $\mathcal{T}(s, a) = a$ to denote the deterministic transition from s to a if taking action a. Let ℓ denote the (unknown) location of the object. The leaves $\mathcal{L}$ of $\mathcal{G}$ represent the (final) goal nodes that could contain the object of interest. The belief state b over the object of interest's location is defined as a vector of probabilities over each $\ell \in \mathcal{L}$.

The observation space $\mathcal{Z}$ is the set $\{0, 1\}$, where success (observing the object as a child of a leaf) yields 1 and 0 otherwise. We also assume perfect observations, so we say $\mathcal{O}_a(s, z) = z$. Since our belief is over leaves rather than agent states, the standard belief update in (1) is unsuitable for our purposes; we discuss our alternative in Section 4.5.2. Lastly, we have the reward $\mathcal{R}$ and discount factor γ, which are defined similarly to the standard POMDP formulation. The reward function should incentivize the agent to find the object as efficiently as possible.

4 PROPHE-C: Solving 3DSG-POMDPs

In this section, we describe our proposed planning algorithm, PROPHE-C. Broadly speaking, we make use of *hallucination* to add foresight about potential layouts of the map, and then sample for

promising trajectories through a hallucination to identify which next available leaf action is the best; sampling is especially effective due to its computational speed combined with the small and discrete nature of our state space. We will first outline the high-level components of the algorithm, and then follow with descriptions of their finer details.

4.1 Method Overview

At any time, the agent will have observed some set of 3DSG nodes. For instance, in Figure 1, the robot in the `closet` sees a `shelf`, and doorways to the `basement` and `living room`. All of these nodes, regardless of their type, are classified as leaves; these leaves form the action space for this current timestep. Our method also assumes access to a trained model that can correctly construct 3DSGs (i.e., it correctly segments and labels the nodes for the environment's 3DSG) given some visual input. As illustrated in Figure 2a, our planner is comprised of five main components:

Step 1. Leaf identification. We identify the set of leaves that are known but whose children have yet to be observed, a prerequisite for graph hallucination and belief construction. See Section 4.2.

Step 2. Scene graph hallucination. Given the current 3DSG and leaves, the agent hallucinates what a complete 3DSG for the environment might look like. See Section 4.3.

Step 3. Belief state creation. After hallucinating a 3DSG, we compute the belief vector over the leaves potentially containing the target object. See Section 4.4.

Step 4. Trajectory sampling. For each hallucinated 3DSG, we sample many possible trajectories in that 3DSG to calculate the most valuable action to take. See Section 4.5.

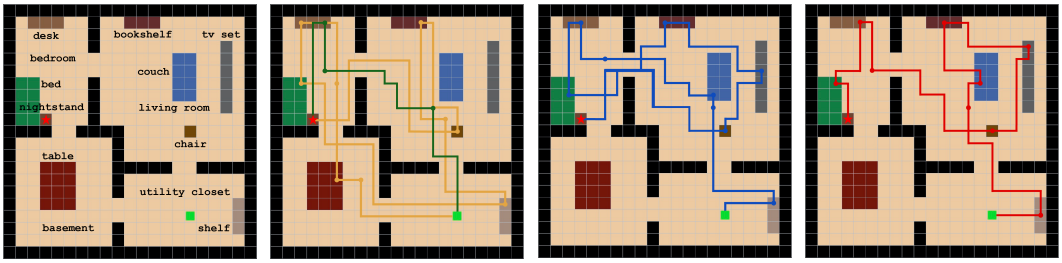
Step 5. Action execution. Execution of the selected action (i.e., target node) from Step 4 requires computing a navigation plan to the desired location. See Section 4.6.

Algorithm 1 PROPHE-C

initialize $\mathcal{G}$ ▷ observed scene graph
initialize m, n ▷ samples, horizon length
initialize g ▷ number of graphs to hallucinate
function PROPHE-C($s_0, \text{query}, m, n, g, \mathcal{G}$)
    tried $\leftarrow \{\}$
    $s_t \leftarrow s_0$
    while true **do**
        best_actions $\leftarrow \{\}$
        for i < g **do** ▷ sample scene graphs
            hallucinate $\mathcal{G}'$ from $\mathcal{G}$ and $\mathcal{L}$
            $b \leftarrow \mathcal{Q}(\mathcal{G}', \text{query}, \text{tried})$
            $T, V \leftarrow \{\}$ ▷ trajectories, values
            for j < m **do** ▷ sample trajectories
                traj $\leftarrow \{s_t\}$
                for k < n **do**
                    $a \sim b$
                    traj.INSERT(a)
                end for
                T.INSERT(traj)
                $v \leftarrow$ROLLOUT(traj, b, 1)
                V.INSERT(v)
            end for
            $\text{traj}^* \leftarrow \underset{T}{\text{argmax}}(V)$
            best_actions.INSERT($\text{traj}^*[1]$)
        end for
        $a_t \leftarrow \text{mode}(\text{best\_actions})$
        $\mathcal{G}, s_t, z_t \leftarrow$EXECUTEACTION($a_t, \text{query}, \mathcal{G}$)
        if $z_t = 1$ **then** ▷ termination condition
            break
        end if
        tried.INSERT(a_t)
    end while
end function

4.2 Leaf Identification

Identifying the initial leaves is critical because (i) it provides start points for graph hallucination, and (ii) helps us prune our belief vector to only include leaves we have not checked, keeping it smaller. We use a set of "tried" leaves, explicitly keeping track of previously checked nodes.



(a) Ground-truth, with labeled rooms and furniture pieces.

(b) Agent trajectories for random search (in yellow) and greedy search with domain knowledge (in green).

(c) Agent trajectory using max likelihood greedy via language embeddings.

(d) Agent trajectory using PROPHE-C.

Figure 3: An example of our household environment encoded by a 3D scene graph. The agent is shown in bright green, and the hidden object (in this case a `laptop`) located on the `nightstand` is denoted by the red star. Circles denote actions/observations taken to reveal a room or furniture item.

4.3 3DSG Hallucination

As the 3DSG is partially observed, the full state and action spaces are unknown, making planning for the future difficult. Hallucination addresses this by filling an existing 3DSG with *possible* new nodes until it has reached some constraint, such as an expected maximum number of rooms, or furniture items. Since generative graph models, and specifically, conditional graph generation, are themselves open research problems that have gained interest in recent years [28, 25, 22], we consider it outside the scope of this work. Instead, we use the link probabilities between rooms and furniture from our dataset to obtain priors for what possible nodes could be linked to currently observed nodes, and then sample from them.

4.4 Belief State Creation

Every time we hallucinate a complete 3DSG, as done on Line 10 of Algorithm 1, we construct a belief vector for all of the leaves in that environment. This is where we leverage language models to provide the initial belief distribution.

4.4.1 Language Model Querying

Our query function $\mathcal{Q}$, used on Line 11 of Algorithm 1, takes the target object string, such as `apple`, `cup`, `book`, and computes a prior belief of that object being a child of leaf nodes (of the complete/hallucinated 3DSG) such as `fridge` or `cupboard`. Additionally, it accesses the set of tried nodes to prevent those nodes from being added to the new belief vector. We leverage existing language models, particularly the English language Tok2Vec values from the SpaCy library's small, medium, and large models [8], to consider the similarity between the target object string and each leaf node string in the word embedding space. To address the issue that the same furniture type existing in different rooms could contain very different items (e.g., `bathroom shelf` versus `living room shelf`), we instead consider the *context vector* $\mathbf{c}(\ell) = [\ell, \text{Parent}(\ell), \text{Parent}(\text{Parent}(\ell)), ..., \text{Root}]$ of each leaf node, which consists of all nodes connecting the leaf and root nodes. To measure the similarity between the target object q and context vector $\mathbf{c}(\ell)$ of leaf node ℓ, we use the geometric mean,

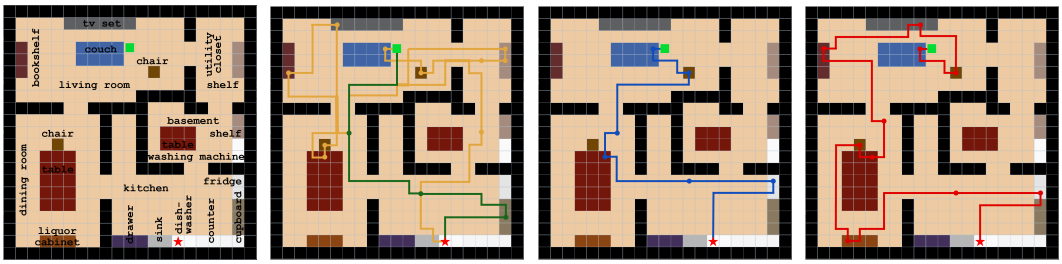
$$\text{similarity}(q, \mathbf{c}(\ell)) = \left(\prod_{i}^{|\mathbf{c}(\ell)|} \langle e(q), c_i(\ell) \rangle \right)^{\frac{1}{|\mathbf{c}(\ell)|}} \tag{2}$$

where $e(\cdot)$ is the embedding vector from the language model and $\langle \cdot, \cdot \rangle$ is the cosine similarity between two vectors. A softmax is applied to normalize the probabilities.[1]

[1] A large language model could be prompted to produce probabilities, but using similarities in the embedding space provides direct numerical values.

4.5 Sampling-based Planning

With a hallucinated graph and an initial belief distribution over the leaves, we sample many trajectories, i.e., action sequences, over a pre-determined horizon. We evaluate each trajectory using a reward function R.



(a) Ground-truth, with labeled rooms and furniture pieces.

(b) Agent trajectories for random search (in yellow) and greedy search with domain knowledge (in green).

(c) Agent trajectory using max likelihood greedy via language embeddings.

(d) Agent trajectory using PROPHE-C.

Figure 4: Example of a slightly larger household environment. In this case, the hidden object is a `cup`, located in the `dishwasher`. Using PROPHE-C, the agent actually checks more items before finding the `cup` than max likelihood, deciding to fully explore a room before checking the next one.

4.5.1 Reward Function

Given the robot's current state s, action a, and belief b, the reward function is the current belief over a leaf a offset by a's distance from the current state s, $R(s, a, b) = b(a) - \eta\|s - a\|$ where η trades off being greedy (choosing max likelihood) and lazy (choosing the closest object)[2]. The distance between s and a is computed using Dijkstra's algorithm.

4.5.2 Trajectory Rollouts

Given a hallucinated 3DSG, the rollout function evaluates leaf-node transitions in a sampled action sequence $\{a_0, a_1, ..., a_n\}$ by summing its rewards based off the definition in Section 4.5.1. This is also where belief updates occur, since our reward uses belief; we update by removing the previous action and renormalizing the belief vector at each step. These rollouts are computed on Line 20 of Algorithm 1. After finding the best trajectory for a hallucination, we append the first step in that trajectory to a list of best nodes; once the hallucinations are finished, we find the mode of the list, shown on Line 26 of Algorithm 1.

4.6 Intermediate Action Execution

Now, we offload execution of the most promising action to a planner such as model predictive control or A*, observe feedback from the environment, and then replan in a receding horizon fashion. In the environments introduced in Section 5.1, our function implements a deterministic path planner to calculate distance. For visualization, we use A*, but during experimentation, we instead use a Euclidean distance heuristic to reduce computation times.

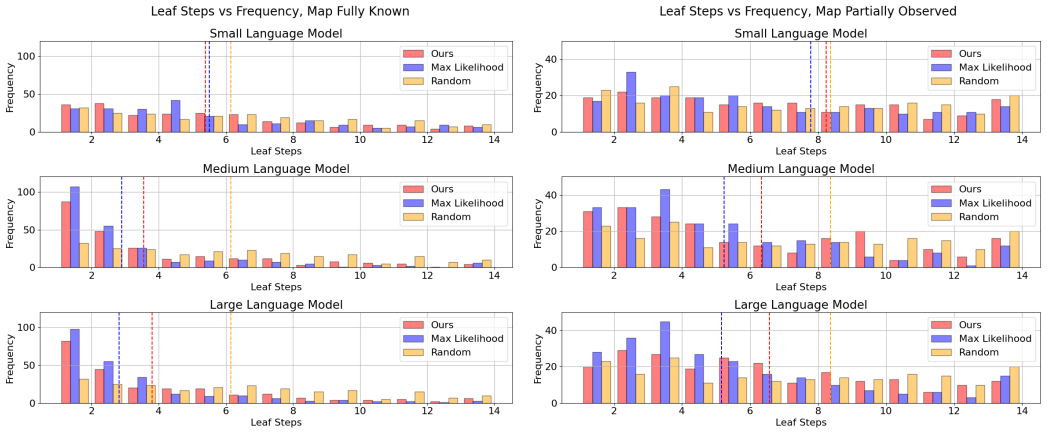
5 Experimental Setup

In this section, we describe the comparison methods and experimental setup. Our code can be found at: https://github.com/UW-CTRL/prophe-c

5.1 Household Environment

We constructed a household simulation environment complete with rooms, structures, and objects which was used to generate 3DSGs. A random household with random floor plans and objects can be generated. These household 3D scene graphs can contain up to six rooms, from: `kitchen`, `living room`, `utility closet`, `bedroom`, `garage`, `dining room`, `bathroom`, `basement`, and `pantry`. Additionally, we have over twenty furniture-like items, and forty target objects. Our dataset for evaluation contains 1000 of these generated household environments. Figure 3 shows a 2D representation of the example environment in our test set. We emphasize that the world shown by

[2]Recall that the set of available actions is the set of known leaf nodes



(a) Number of actions with fully known 3DSG. (b) Number of actions with partially observed 3DSG.

Figure 5: Histograms displaying the number leaf steps taken until search completion for each episode. The dotted line indicates the mean. Since greedy search with domain knowledge always succeeds within a few steps, we omitted it for visual clarity.

Figure 3 is solely for visualization; the planner itself relies only on 3DSGs, which can be generated from physical real-world interactions [19].

5.2 Baseline Comparisons

We compare PROPHE-C against three other methods: *random search*, *max likelihood with domain-specific priors* (we use the probabilities for generating the environments), and *max likelihood with language priors* (we utilize Tok2Vec values from SpaCy's models).

5.3 Test Setup

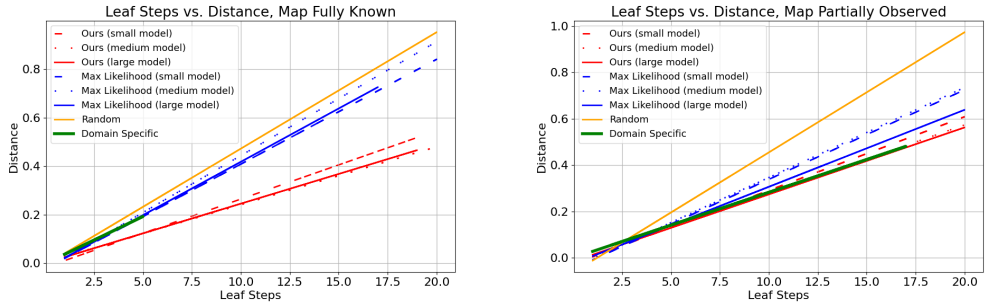
We perform experiments in two scenarios: *fully-known* where the entire 3DSG is known except the object location, and *partially-observed* where only nodes immediately visible to the agent are given initially. Episodes are terminated upon observing the target object. For both scenarios, we perform 250 object search problems. For each method, our goal is to analyze both the decisions made and how efficient the search process was by considering the total distance traveled. In both cases, PROPHE-C has a planning horizon of 3 leaf-steps, which is sufficient for our environment size. We also investigate the effect of language model complexity—we compare the performance for the small, medium, and large SpaCy models whose embedding vector lengths are 96, 300, and 300, respectively.

6 Experimental Results and Discussion

In this section, we discuss qualitative and quantitative results of our experiments. While our environments are small-scale, these experiments are designed to gain insights into the benefits of using language models combined with 3DSG representations and online inference.

Qualitative Results. The agent trajectories displayed in the household environments in Figure 3 show all four methods for the same object search scenario. The agent starts in the `utility closet` and is tasked with finding a `laptop` located at the `bedroom nightstand`. For the agent with domain-specific knowledge, a greedy search is still highly optimal, even under partial observability. However, for the agents whose priors come from much more general knowledge (the SpaCy embeddings), it takes noticeably longer, especially since `nightstand` has less semantic similarity to `laptop` relative to other locations. Our method, though, is more resilient to this and fully explores the `living room` before committing to the `bedroom` and then finding the `laptop`. In contrast, using max likelihood with language priors, the robot inefficiently switches between the `bedroom` and `living room` resulting in a greater distance traveled. Another example is given in Figure 4, where the agent must find a `cup`, which is located in the `dishwasher`. This location is pretty likely; as a result, the max likelihood method travels a little less than PROPHE-C, but PROPHE-C still takes the time to explore one likely room before moving to the next.

Quantitative Results. Our results indicate that max likelihood using a language model yields impressive performance in our environment but that its embeddings ultimately *lack foresight and spatial awareness*. Figures 5a and 5b show that on average, max likelihood with the medium and



(a) Search efficiency with fully known 3DSG (b) Search efficiency with partially observed 3DSG

Figure 6: Linear regression plots showing the relationship between the number of actions taken to find the object and the total distance traveled. For visual clarity, we removed the data points.

large SpaCy models will finish in one less leaf-step than our method. However, these results do not tell the full story; as shown by Figures 6a and 6b, our approach is more efficient in searching, especially where the object's location is highly uncertain, like in Figure 3. We emphasize that more leaf steps do not necessarily indicate longer search times; plotting leaf steps versus distance illustrates that our method will more thoroughly search leaves in the immediate vicinity (hence providing exploratory behavior) before moving to the next room, thus avoiding behaviors that frequently switch between two rooms. Additionally, while the medium model is marginally worse than the large one, we can boost the efficiency of the medium model greater than the large one with online inference (see Figures 6a and 6b). Also evidenced in these charts, max likelihood search results in less distance traveled under partially observed 3DSGs compared to fully known (up to the object's location) ones. We hypothesize that this is due to the lower amount of actions to choose from; as the number of available actions go up, a max likelihood search using language model embeddings may sequentially pick actions that are very spatially costly, whereas under partial observability, it is limited to a much smaller range of actions at any given point. Notice also that the gap in PROPHE-C's performance is noticeably smaller between Figures 6a and 6b.

Key Takeaways. Overall, these tests reveal insights that both build off previous works and are worth considering for future exploration: **(i)** Online inference can both enhance the performance of an existing language model and fill the gap between a smaller model and a larger one. **(ii)** These language models perform well on their own, especially in cases that line up with "common-sense", but suffer in situations where common-sense assumptions break down. **(iii)** 3D scene graphs imbue disembodied language models with spatial awareness when combined with online planning.

7 Conclusion and Future Work

In this work, we leveraged 3D scene graphs, online inference, and language model priors to solve the object search problem in partially observable domains. Our experiments imply language models indeed have remarkable semantic capabilities in novel environments, and combining these capabilities while maintaining belief over the leaves in the 3DSG elicits scalability, exploratory behavior, and travel reduction during planning. Our results indicate promising behavior, and we intend to investigate these trends on larger, more general domains. For instance, we could significantly extend the breadth and depth of the 3DSG, leverage real-world datasets, and perform real-world testing in a house or office buliding. Another avenue that we intend to explore is fully querying a large language model such as GPT-4 and using generated text to guide search through a 3DSG. For this work, we found that the raw embeddings would be easiest to work with since their values can be easily converted into a plausible belief distribution. Additionally, our 3DSG hallucination method was implemented without any learning; an interesting research direction is in conditional 3DSG generation—perhaps this could also be done via the generated output of a language model as well.

References

[1] 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 *Conf. on Robot Learning*, 2021.

[2] S. Amiri, K. Chandan, and S. Zhang. Reasoning with scene graphs for robot planning under partial observability. *IEEE Robotics and Automation Letters*, 7(2):5560–5567, 2022.

[3] 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 *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019.

[4] 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 *Conf. on Neural Information Processing Systems*, 2020.

[5] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *Conf. on Neural Information Processing Systems*, 2020.

[6] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Available at https://arxiv.org/abs/1810.04805*, 2018.

[7] C. R. Garrett, T. andKaelbling L. P Paxton, C. andLozano-Perez, and D. Fox. Online replanning in belief space for partially observable task and motion problems. In *Proc. IEEE Conf. on Robotics and Automation*, 2020.

[8] M. Honnibal and I. Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. *Available at https://spacy.io/*, 2017. To appear.

[9] B. Ichter, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, D. Kalashnikov, S. Levine, Y. Lu, C. Parada, K. Rao, P. Sermanet, A. T. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, M. Yan, N. Brown, M. Ahn, O. Cortes, N. Sievers, C. Tan, S. Xu, D. Reyes, J. Rettinghouse, J. Quiambao, P. Pastor, L. Luu, K-H. Lee, Y. Kuang, S. Jesmonth, K. Jeffrey, R. J. Ruano, J. Hsu, K. Gopalakrishnan, B. David, A. Zeng, and C. K. Fu. Do As I Can, Not As I Say: Grounding language in robotic affordances. In *Conf. on Robot Learning*, 2022.

[10] S-K. Kim, A. Bouman, G. Salhotra, D.D. Fan, K. Otsu, J. Burdick, and A. Agha-mohammadi. PLGRIM: Hierarchical value learning for large-scale autonomous exploration in unknown environments. In *Int. Conf. on Automated Planning and Scheduling*, 2022.

[11] M. J. Kochenderfer. *Decision Making Under Uncertainty*. MIT Press, 2015.

[12] A. Kurenkov, M. Lingelbach, T. Agarwal, C. Li, E. Jin, R. Zhang, L. Fei-Fei, J. Wu, S. Savarese, and R. Martín-Martín. Modeling dynamic environments with scene graph memory. In *Proc. Int. Conf. on Autonomous Agents and Multiagent Systems*, 2023.

[13] 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 *Proc. IEEE Conf. on Robotics and Automation*, 2021.

[14] M. Lingelbach, C. Li, M. Hwang, A. Kurenkov, A. Lou, R. Martín-Martín, R. Zhang, L. Fei-Fei, and J. Wu. Task-driven graph attention for hierarchical relational object navigation. In *Proc. IEEE Conf. on Robotics and Automation*, pages 886–893, 2023.

[15] S. Nikolaidis, S. Nath, A. D. Procaccia, and S. Srinivasa. Game-theoretic modeling of human adaptation in human-robot collaboration. In *IEEE Int. Conf. on Human-Robot Interaction*, 2017.

[16] O. Peltzer, A. Bouman, S-K. Kim, R. Senanayake, J. Ott, H. Delecki, M. Sobue, M. J. Kochenderfer, M. Schwager, J. Burdick, and A. Agha-mohammadi. FIG-OP: Exploring large-scale unknown environments on a fixed time budget. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2022.

[17] K. Rana, J. Haviland, S. Gard, J. Abou-Chakra, I. Ried, and N. Suenderhauf. SayPlan: Grounding large language models using 3D scene graphs for scalable robot task planning. In *Conf. on Robot Learning*, 2023.

[18] 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 *Proc. IEEE Conf. on Robotics and Automation*, 2022.

[19] 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. *Int. Journal of Robotics Research*, 40(12–14):1510–1546, 2021.

[20] D. Silver and J. Veness. Monte-carlo planning in large POMDPs. In *Conf. on Neural Information Processing Systems*, 2010.

[21] Z. N. Sunberg and M. J. Kochenderfer. Online algorithms for POMDPs with continuous state, action, and observation spaces. In *Int. Conf. on Automated Planning and Scheduling*, 2018.

[22] A.M. Tseng, N. Diamant, T. Biancalani, and G. Scalia. GraphGUIDE: Interpretable and controllable conditional graph generation with discrete bernoulli diffusion. In *Int. Conf. on Learning Representations: Workshop on Machine Learning for Drug Discovery*, 2023. Workshop on Machine Learning for Drug Discovery.

[23] A. Vaswani, N. Shazeer, N. Parmar, J Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Conf. on Neural Information Processing Systems*, 2017.

[24] A. Wandzel, Y. Oh, M. Fishman, N. Kumar, L. Wong, and S. Tellex. Multi-object search using object-oriented POMDPs. In *Proc. IEEE Conf. on Robotics and Automation*, 2019.

[25] C. Yang, P. Zhuang, W. Shi, Luu A., and P. Li. Conditional structure generation through graph variational generative adversarial nets. In *Conf. on Neural Information Processing Systems*, 2019.

[26] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Available at https://arxiv.org/abs/2305.10601*, 2023.

[27] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *Available at https://arxiv.org/abs/2210.03629*, 2022.

[28] J. You, R. Ying, X. Ren, Hamilton W. L., and J. Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *Int. Conf. on Machine Learning*, 2018.

[29] K. Zheng, R. Chitnis, Y. Sung, G. Konidaris, and S. Tellex. Towards optimal correlational object search. In *Proc. IEEE Conf. on Robotics and Automation*, 2022.

[30] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. K. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Proc. IEEE Conf. on Robotics and Automation*, 2016.