

Actively Resolving Contextual Uncertainty for Underspecified Tasks in Natural Language

Zachary Ravichandran¹, Jonathan Diller¹, Fernando Cladera¹,
Varun Murali², George J. Pappas¹, and Vijay Kumar¹

¹ GRASP Laboratory, University of Pennsylvania

² Electrical & Computer Engineering, Texas A&M University
Correspondence to zacravi@seas.upenn.edu

Abstract. Foundation models provide robots with the ability to interpret natural language and reason about environmental context, yet most language-conditioned policies assume that goals are well-specified and that task-relevant information is provided upfront via a prior map. Operating in unfamiliar environments with underspecified tasks entails high contextual uncertainty—the robot must jointly infer what constitutes task success, what constitutes relevant information, and where (or whether) that information exists. We address these limitations via CLUE (Closed-Loop contextual Uncertainty rEsolution), a framework for actively resolving contextual uncertainty given underspecified tasks in natural language. CLUE uses an LLM-derived policy to hypothesize task-relevant concepts and potential plans. It then uses a language-embedded map, which is constructed online, to ground these hypotheses into actions. The policy sequentially evaluates hypotheses via closed-loop environment interaction and refines its plans as it gathers new information. We deploy CLUE on a Boston Dynamics Spot across three real indoor and outdoor environments spanning 15 tasks that require object disambiguation, functional inference, and occlusion reasoning. CLUE achieves a success rate within 7 percentage points of an oracle policy and outperforms an LLM-enabled planner without closed-loop feedback by a 4x margin. Supporting experiments demonstrate that simply building and then querying a language-enriched map is insufficient to resolve complex contextual planning tasks; these approaches achieve roughly one third the success rate of CLUE while requiring over 10x more VLM tokens. We provide additional information at <https://zacravichandran.github.io/CLUE>

Keywords: Language-driven planning, Active perception, Foundation models for robotics, Semantic mapping, Contextual uncertainty

1 Introduction

Consider a robot placed in an unfamiliar area and tasked to “locate the barrel with the hazardous content,” as illustrated in Figure 1. Resolving this specification requires addressing several coupled forms of uncertainty about semantic (what constitutes a hazard, what constitutes success), metric (where are the relevant entities), and functional (occlusions prevent inspection but can be removed) information, which we refer to as *contextual uncertainty*. A successful robot will infer subtasks and relevant entities, navigate to inspect them, and resolve unexpected

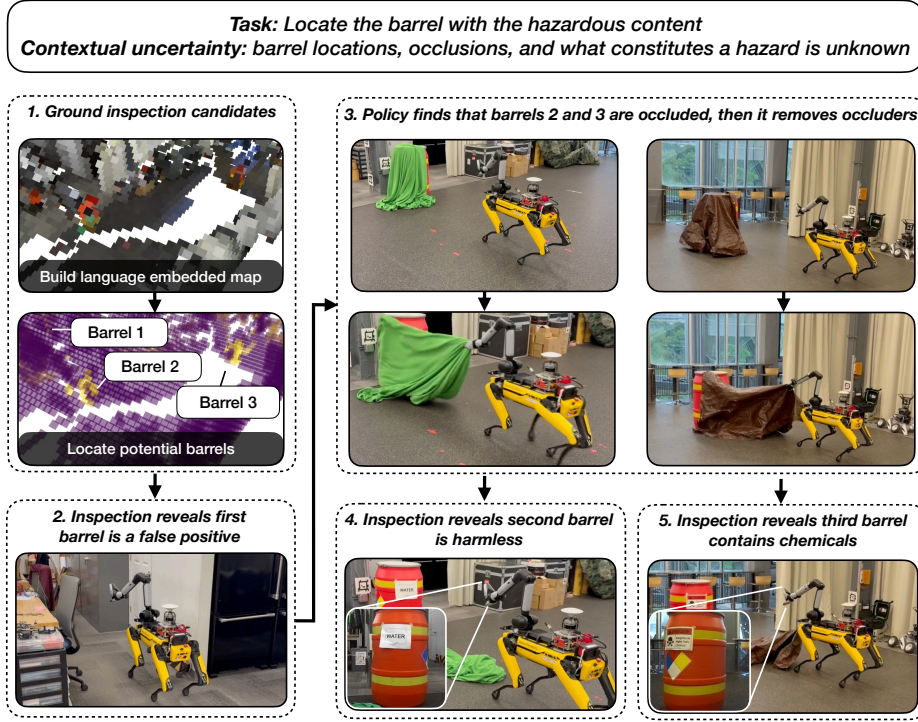


Fig. 1: **Example Task.** The robot must identify a hazardous barrel, but it does not know barrel locations or what constitutes a hazard *a priori*. It uses a language-embedded map constructed from RGB-D data to identify possible barrel locations. It rules out one as a false positive upon inspection. The other two candidates are barrels but are occluded by tarps. The robot removes the tarps, reads the signs hidden underneath, and finds the third barrel contains dangerous chemicals.

challenges—such as occlusions—that arise along the way. While a large body of work builds language-aware planners using large language models (LLMs) and vision-language models (VLMs), these planners largely assume explicit task instructions and high-quality prior information, limiting their applicability in contextually uncertain scenarios [23,36].

Classically, actively resolving uncertainty in task planning has been posed from a geometric perspective: the robot builds a metric map by selecting view-points that maintain localization accuracy or coverage [35,40]. Such formulations reason about where the robot has and has not looked, but they are largely agnostic to contextual associations—that a tarp may conceal a barrel or labels reveal hazard levels. Conversely, current language-driven planning work largely treats map retrieval as sufficient for task grounding. Open-vocabulary map representations let a robot query a pre-built map [5] or online mapper [38,24] with free-form language, but they assume a single query returns the correct answer and offer no mechanisms to verify detections. Dense scene-description methods caption every object and region encountered during mapping [11], but because captioning happens without knowledge of the task, the resulting descriptions are generic,

expensive to produce, and still cannot prompt the robot to *act*—e.g. move a tarp when it is occluding task-relevant information. A parallel line of work casts language-driven planning under uncertainty as belief-space or POMDP planning with foundation models [33,39,34,7,16,13]. These methods are largely designed for information retrieval or manipulation tasks in constrained workspaces, and they do not demonstrate the closed-loop integration of hypothesis refinement, targeted inspection, and physical intervention required for real-world contextually uncertain tasks. While they reason over uncertainty in object existence or location, they assume the task-relevant semantics are known *a priori*. In contextually uncertain tasks, relevant semantics are themselves latent variables that can only be resolved through closed-loop interaction with the environment.

We address these limitations via CLUE (Closed-Loop contextual Uncertainty rEsolution), a closed-loop framework for resolving contextual uncertainty that couples a language-embedded map with an LLM-enabled policy that iteratively replans as it gathers information. Given an underspecified instruction, CLUE uses an LLM to hypothesize contextually relevant information and potential actions. A language-embedded map is used to ground these hypotheses into concrete information-gathering goals; if a hypothesis is too precise to be grounded in the map, the LLM may broaden its hypothesis (e.g., “red backpack” to “backpack” to “bag”). CLUE then actively validates these hypotheses through closed-loop interaction with the environment via VLM inspection, navigation, and manipulation. CLUE uses observations acquired from interaction to update its hypothesis—a tractable approximation to belief for open-vocabulary environments that captures its understanding of the state (confirmed entities, what remains uncertain, and what must be verified) and goal conditions. As CLUE gathers new information, its hypotheses evolve to drive plan generation. To summarize, our contributions are:

1. A closed-loop framework that couples LLM-enabled contextual inference and language-embedded map grounding, enabling active information gathering for resolving contextual uncertainty.
2. A grounding mechanism that translates LLM-generated hypotheses into concrete planning objectives via a language-embedded map.
3. Feedback mechanisms via language-embedded map querying and targeted VLM inspection to resolve hypotheses and gather task-specific information.

We experimentally validate these contributions through fifteen tasks requiring CLUE to disambiguate objects based on contextual relevance, reason over functional affordances, and resolve occlusions observed online, all performed on a Boston Dynamics Spot robot in three real-world environments. We find that CLUE achieves a task success rate of within 7 percentage points of an oracle and over four times that of an LLM-enabled policy without closed-loop feedback. In supporting experiments, CLUE achieves over three times the success rate of a state-of-the-art scene understanding framework which builds a language-enriched map *a priori* then queries it for information—demonstrating that closed-loop environment interaction is necessary for resolving contextual uncertainty.

2 Related work

We now overview three key areas of related work.

LLM-Driven Planning. A primary methodological divide in works that leverage foundation models for planning is whether the model directly generates control sequences or plans over abstraction (i.e. behaviors) that are realized by downstream controllers. For single-robot systems, recent Vision-Language-Action (VLA) models favor direct end-to-end control, as demonstrated in [9,4,17], where the language agent maps open-world task descriptions and visual inputs directly to low-level motor commands, a trend mirrored in other single-agent [28,5] and multi-agent [15,6] planners. While direct LLM planning excels at implicit semantic reasoning, it often lacks formal safety guarantees and struggles with high-frequency reactive constraints. In contrast, works such as [21,29,37,26,8] position the LLM as a high-level translator that feeds objective weights, constraints, or 3D waypoints into robust classical solvers (e.g., MPC or RRT), ensuring safe, continuous execution while maintaining the LLM’s semantic adaptability.

Language-Driven Planning with Active Perception. Recent works have increasingly leveraged both LLMs and VLMs to handle active perception requirements during task planning in open-world environments. Several approaches model the problem in PDDL and utilize foundation models to resolve visual ambiguities or verify preconditions backward from a goal state during multi-step household tasks [23,19,36]. While these systems are able to ground language in visual observations, they generally require target objects to be in the robot’s immediate field of view. Other active perception methods focus on constructing and querying hierarchical scene graphs or other related semantic reconstructions of the environment [11,28,5]. These methods rely on a near-complete understanding of the environment prior to execution—assuming that map retrieval is sufficient to solve the task—and lack the ability to reason over partial observability.

Language-Driven Planning with Uncertainty. A line of research extends LLM planning methods to incorporate notions of uncertainty [30,18]. A common approach is to frame robotic planning within belief spaces, a concept rooted in early discrete symbolic planning [32]. These works often formulate the problem as a Partially Observable Markov Decision Process (POMDP). Recent frameworks utilize LLMs to generate POMDP transition and observation models, enabling robots to locate hidden objects [7,34,13,16,33]. Furthermore, multi-agent systems and LLM-based planners have been deployed to handle missing state variables during long-horizon manipulation tasks [22,39,3]. Although these methods can handle various forms of uncertainty, they lack the cognitive flexibility to dynamically interpret the task-relevance of an object (e.g., determining that a barrel is a “hazard”) or assess the affordances of an object as they update their semantic understanding of the scene (e.g., ascertaining that a cart can support a laptop). Unlike existing methods, our approach bridges this gap by reasoning over semantically embedded maps to locate conceptually relevant items, and subsequently interpreting objects and evaluating their physical appropriateness.

3 Problem Definition

A robot is placed in an unknown environment and must achieve an underspecified task ℓ in natural language. The robot is equipped with a set of high-level behaviors for navigation, inspection, and manipulation, an odometry system,

and an RGB-D camera. We assume (i) the robot’s pose is available from a reliable odometry source, (ii) the environment is static over the course of an episode, (iii) the information needed to satisfy ℓ is observable from RGB-D imagery once the appropriate location is inspected, possibly after manipulation to resolve occlusion, and (iv) the robot is given no predefined object catalog or scene graph.

POMDP formulation. We model this problem as a POMDP. The state space $\mathcal{S}$ comprises the robot’s pose and its surrounding environment, including semantics. The current state $s_i \in \mathcal{S}$ is unknown and must be estimated via observations $o_i \in \mathcal{O}$, which are constructed from RGB-D imagery upon executing a behavior. The action space $\mathcal{A}$ encompasses robot behaviors instantiated with referents in the environment. Referents are properties of the environment, so only the subset $\mathcal{A}(s_i) \subseteq \mathcal{A}$ is feasible in state s_i : e.g. `inspect(barrel)` can be executed only if a barrel exists and is reachable. Because s_i is not observed, the feasible set is itself uncertain: at each iteration the policy proposes actions it *believes* feasible, and a grounding step rejects those that cannot be realized against the robot’s current environment representation (§4.3). Each task induces a latent set of goal states $\mathcal{S}^\ell \subseteq \mathcal{S}$, i.e. what constitutes task success is not given explicitly but must be inferred from the instruction and observations gathered online. The robot must both reach a goal state and report task completion, which it does via a terminal `answer( $r$ )` action whose argument r conveys the task-required information:

$$R(s_i, a_i) = \begin{cases} +1, & \text{if } a_i = \text{answer}(r) \text{ and } s_i \in \mathcal{S}^\ell \\ -1, & \text{if } a_i = \text{answer}(r) \text{ and } s_i \notin \mathcal{S}^\ell \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The episode terminates if `answer` is executed, or the action budget N is exhausted.

Objective. Planning under uncertainty may be addressed by maintaining a belief state $b_i \in \mathcal{B}$, where $b_i(s_i) = \mathbb{P}(s_i \mid o_{1:i}, a_{1:i-1})$ is a probability distribution over the true state. The robot chooses actions based on its policy $\pi: \mathcal{B} \rightarrow \mathcal{A}$, and we seek

$$\pi^* = \arg \max_{\pi} \mathbb{E} \left[\sum_{i=0}^{N-1} \gamma^i R(s_i, a_i) \mid b_i, \pi \right] \quad (2)$$

where $\gamma \in (0, 1)$. Because at most one reward is non-zero per episode, Equation 2 amounts to maximizing the probability of terminating in a goal state while discouraging both superfluous information gathering and premature answers.

Contextual uncertainty. The underspecified task and unknown environment together induce *contextual uncertainty*, which manifests in three coupled dimensions. *Semantic uncertainty* concerns which entities are task-relevant and what constitutes task success. *Metric uncertainty* concerns where those entities are located, and hence which actions are feasible. *Functional uncertainty* concerns which physical interactions are required to make task progress, e.g. whether an occluding object must be moved before a referent can be inspected. Resolving contextual uncertainty amounts to maintaining a belief jointly over $\mathcal{S}^\ell$, s_i , and consequently $\mathcal{A}(s_i)$, updated as observations arrive.

Tractable approximation. Maintaining this belief exactly is intractable: the state space is continuous and open-vocabulary, and folding the unknown goal con-

Algorithm 1 Closed-Loop Contextual Uncertainty Resolution

```

1: Input: Task  $\ell$ , action budget  $N$ 
2:  $\mathcal{M}_0 \leftarrow \emptyset, i \leftarrow 1, o_0 \leftarrow \emptyset, h_0 \leftarrow \emptyset$ 
3: while  $i \leq N$  do
4:    $h_i, a_i \leftarrow \pi_{LM}(o_{i-1}, h_{i-1}, \ell)$ 
5:   if  $a_i$  is hypothesis_grounding then
6:      $(\mathcal{C}_i, \hat{a}_{1:C}) \leftarrow \mathcal{G}(h_i, \mathcal{M}_i)$ 
7:      $o_i \leftarrow \text{ToObservation}(\mathcal{C}_i, \hat{a}_{1:C})$ 
8:   end if
9:   if  $a_i$  is active_information_gathering then
10:     $o_i \leftarrow \text{execute}(a_i)$ 
11:   end if
12:   If  $a_i = \text{answer}(r)$ ; break
13:    $i \leftarrow i + 1$ 
14: end while

```

ditions $\mathcal{S}^\ell$ into the state only compounds its dimensionality. We therefore approximate the belief via a *hypothesis* state $h_i \in \mathcal{H}$, a symbolic account of the task, discovered entities, and outstanding uncertainties, whose structure is detailed in §4.2 and whose fields correspond directly to the three dimensions above. This yields a hypothesis update

$$\tau: (h_{i-1}, o_i, \ell) \mapsto h_i, \quad (3)$$

paired with a policy over hypotheses

$$\pi: h_i \mapsto a_i, \quad (4)$$

both of which CLUE realizes with a single query to a pre-trained LLM (§4.2). The entities carried in h_i are precisely those over which the policy proposes behaviors, and grounding against the environment representation determines which of those proposals are feasible. This does not solve Equation 2 exactly; rather, it serves as a tractable approximation that replans as each new observation reduces the robot’s uncertainty about the current state s_i and goal conditions $\mathcal{S}^\ell$.

4 Actively Resolving Contextual Uncertainty with CLUE

We address the above problem via CLUE, a closed-loop framework that pairs an LLM-enabled policy and a language-embedded mapping framework, as shown in Figure 2. The policy has access to two categories of actions: *hypothesis grounding* and *active information gathering*. In the first, the policy grounds hypotheses, which include task-relevant entities, into concrete action objectives. In the second, the policy actively validates hypotheses via inspection, navigation, and manipulation, or otherwise makes progress towards the task. The remainder of this section describes CLUE’s environment representations and observation space (§4.1), policy inference strategy (§4.2), hypothesis grounding method (§4.3) and active information gathering (§4.4).

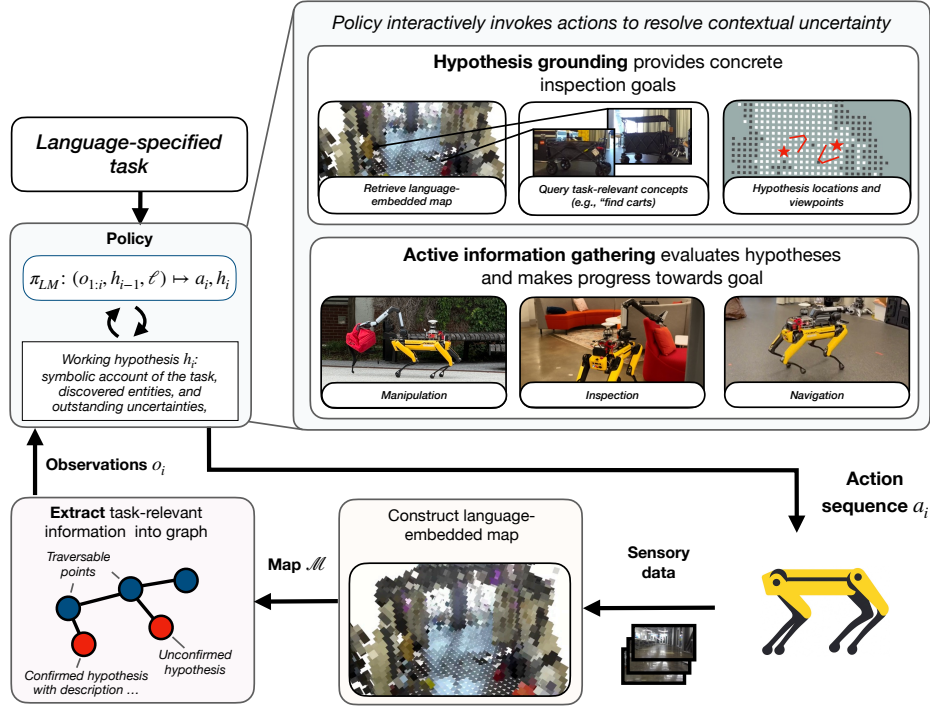


Fig. 2: **CLUE architecture.** CLUE uses an LLM as a policy to realize underspecified language tasks in a closed-loop manner via two action types—hypothesis grounding and active information gathering. The LLM then constructs actions comprising manipulation, inspection, and navigation to resolve contextual uncertainties and make task progress. Information from the physical world is provided to the LLM via a symbolic graph-based observation space.

4.1 Environment Representations

The primary challenge in designing an observation space is that the robot operates in a continuous, three dimensional world, while the LLM reasons over a symbolic (textual) space. CLUE addresses this by maintaining two levels of abstraction, each with distinct roles. The *language-embedded map* $\mathcal{M}_i = \{(x_j, y_j, z_j, f_j)\}$ is a dense voxel map whose feature embeddings f_j are accumulated online from RGB-D imagery and may be queried for semantic relevance (§4.3), and is augmented with updates $\Delta\mathcal{M}_i$ at each planning iteration. It serves as the robot’s perceptual memory, but it is inefficient to expose to the policy directly. We thus define an observation graph, a sparse, symbolic abstraction comprising discrete entities and traversable regions annotated with coordinates and natural-language attributes. The graph comprises two node types: object and region. Objects denote semantic entities, while regions are points in freespace traversable by the robot. Each node carries a coordinate and may be enriched with attributes in natural language, such as the output of a VLM query. These updates, along with associated metadata, constitute the observation o_i , which is provided to the LLM via in-context updates from a textual API which denotes graph operations

such as `add_node()` or `remove_edge()`. For example, the second barrel from the example in Figure 1 would yield the update:

```
add_node(name=barrel_2, coordinates=[5, -2], description=...)
add_node(name=near_barrel_2, coordinates=[0,0])
add_edge(barrel_2, near_barrel_2)
```

The description of these objects would change throughout the task. For example, `barrel_2` may start out with a description “candidate location from language-embedded map,” change to “occluded by tarp,” then finish with “sign indicates barrel contains water.”

4.2 Contextual Planning

We instantiate our policy defined in §3 with a pre-trained LLM that is configured with a system prompt that defines its role in CLUE.

LLM Configuration. The LLM’s system prompt specifies several components. It first describes the observation space and the action space (i.e. behaviors and instructions for instantiating them with environmental entities). It then provides a “planning principles” section which describes the sequential and closed-loop nature of the problem. Finally, it provides instructions for constructing a hypothesis state and the required output format.

Hypothesis state. At each planning iteration, the LLM emits a structured *hypothesis state* h_i : a running, symbolic account of the task. This hypothesis state tracks what the robot has confirmed (observed objects and regions), what it has hypothesized but not yet verified (inferred or candidate entities, together with hypotheses already ruled out), and the open uncertainties that remain task-relevant. Carrying this state in the output lets the policy reason across iterations: confirmed entities ground later decisions, ruled-out hypotheses prevent repeated effort, and outstanding uncertainties drive the next round of information gathering. A condensed schema is shown below.

```
{ "goal": "...", "confirmed": { "objects": [...], "regions": [...] }, "hypothesized": {
  "entities": [...], "ruled_out": [...] }, "uncertainties": [...] }
```

4.3 Hypothesis Grounding

The policy translates hypotheses into concrete planning objectives by invoking a grounding process $\mathcal{G}(h_i, \mathcal{M}_i) \rightarrow \{\mathcal{C}_i, \hat{a}_{1:C}\}$. As illustrated in Figure 3, this process takes as input a hypothesis and current language-embedded map $\mathcal{M}_i$, and it provides C candidate inspection locations $\mathcal{C}_i$ along with a suggested distance-optimal visitation sequence $\hat{a}_{1:C}$.

Map Querying. The policy triggers a grounding by identifying a task-relevant concept from a hypothesis and calling a query action. This concept is then queried in the map, which is constructed online from sensor data (Fig. 3.A-B), by first encoding the concept into a feature vector, $f_c = \phi(c)$, then computing the cosine similarity between the concept and every feature vector in the map,

$s_j = f_c^\top f_j / \|f_c\| \|f_j\|$, $\forall f_j \in \mathcal{M}_i$. Because this score is difficult to interpret as-is, we normalize it with a softmax against a set of background concepts (e.g., “object,” “ground,” “wall”), yielding a confidence $\kappa_j = e^{s_j} / (e^{s_j} + \sum_b e^{s_{b,j}})$, where $s_{b,j}$ is the similarity of voxel j to background concept b , and we retain all voxels whose confidence exceeds a threshold (Fig. 3.C). We then cluster retained voxels using the DBSCAN algorithm [10], yielding a set of clusters $\{(x_m, y_m, n_m, \kappa_m)\}_{m=0}^M$, each comprising a centroid (x_m, y_m) , a size in voxels n_m , and an average confidence κ_m (Fig. 3.D). The map’s feature embeddings may not be precise enough for the task at hand. For example, the task may require finding a “red backpack,” but only “bag” is identifiable by the embeddings. The policy may therefore broaden its query if initial results come up empty.

Candidate selection and viewpoint planning. CLUE only considers candidates with a confidence above threshold λ , yielding a set of candidates $\mathcal{C}_i = \{(x_c, y_c, n_c, \kappa_c)\}_{c=0}^C$. For each candidate, CLUE computes a viewpoint from which the robot can further inspect the candidate, if desired. We define a target viewing distance—i.e. how far the robot should stand from the object—and place a candidate viewpoint on the straight line between the robot’s current pose and the target, at that distance from the object. Finally, CLUE snaps this viewpoint to the nearest unoccupied point in freespace (Fig. 3.E). Upon a query, the candidate entities and their viewpoints are added to the observation as object and region nodes (the process $\text{ToObservation}(\mathcal{C}_i, \hat{a}_{1:C})$ in Algorithm 1). CLUE also provides a suggested minimum-distance inspection tour to the policy via observation metadata, where d_j is the distance incurred by the j -th action in the tour:

$$\hat{a}_{1:C} = \arg \min_{a_{1:C}} \sum_{j=1}^C d_j. \quad (5)$$

CLUE computes a distance-optimal visitation sequence by solving a Traveling Salesman Problem (TSP), where distance is Euclidean distance between points, and it provides this ordering to the policy (Fig. 3.F). The value of this hint is evaluated in §5.5. However, this sequence may be missing necessary preconditions—for example the robot must navigate near a candidate before inspection or remove occlusion—it thus serves as a hint from which the policy can construct an action.

4.4 Active Information Gathering

Having either queried the language-embedded map for candidates or acquired observations through direct environment interaction, the policy constructs and executes an action sequence to make task progress. To realize the candidates as actions, the LLM infers the necessary preconditions, such as navigating to the associated viewpoint or clearing occlusion in order to properly inspect a candidate, as in Figure 1. The policy replans as new information is discovered, and may terminate early once the task is resolved.

The policy constructs actions by instantiating behaviors with concrete goals in the environment. We consider six behaviors: `goto()`, `inspect()`, `grasp()`, `place()`, `query()`, and `answer()`. `goto()` drives the robot to a location; `inspect()` calls a VLM with an LLM-generated query (e.g., “does the barrel contain hazardous content”) and retrieves the response; `grasp()` and `place()` perform manipulation; and `answer()` terminates the task with a response to the user. The

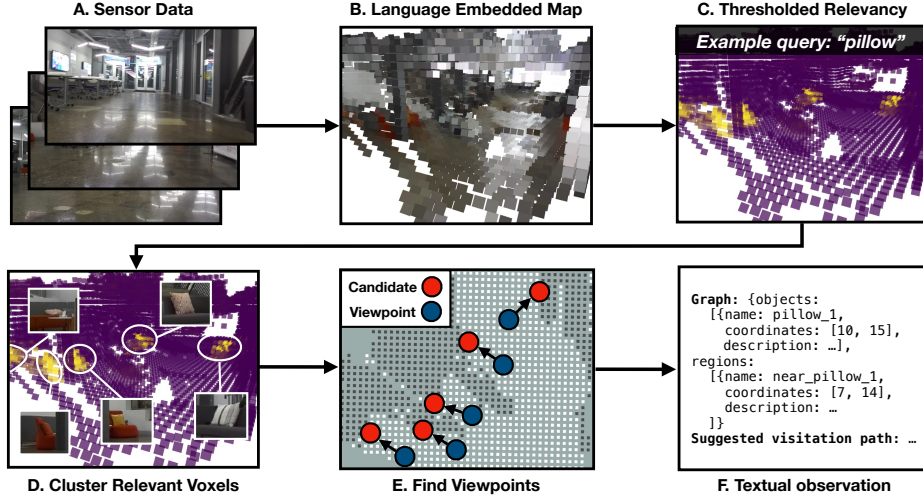


Fig. 3: Hypothesis grounding. Sensory data is used to construct a language-embedded map. Given a concept from the policy’s hypothesis, candidate entities are identified by finding high-relevancy voxels and clustering. Viewpoints from which the candidates may be observed are identified via the map’s occupancy grid. These candidates and viewpoints form a graph which is provided to the policy via a textual observation along with a minimum distance visitation order.

policy instantiates actions using entities currently in the observation graph. For example, the valid actions from the episode illustrated in Figure 1 include `goto(near_barrel_1)` and `inspect(barrel_1)`.

5 Experimental Evaluation

We design experiments to evaluate the contributions claimed in §1. Concretely, we aim to answer the following questions: **Q1.** Can CLUE realize tasks that require resolving contextual uncertainty? **Q2.** Is closed-loop feedback required, or does open-loop planning suffice? **Q3.** Is CLUE’s grounding process necessary, or does simply building and then querying a semantic map suffice? We describe our implementation details in §5.1, our experimental setup—including environments, tasks, and baselines—in §5.2. We report results in §5.3, and we provide supporting experiments in §5.4 and §5.5.

5.1 Implementation details

We implement our mapping framework by extending RayFronts [2] with the RadSeg feature encoder [1]. We use SciPy’s implementation of DBSCAN [31] and implement a TSP solver using Gurobi [12]. We implement our policy and its VLM using GPT-5.1 with “low” reasoning [25]. We run all experiments on the Boston Dynamics Spot, which is equipped with a forward-facing ZED 2i camera for RGB and depth sensing, an NVIDIA Jetson AGX Thor for compute, and provides odometry via internal state estimation. Mobility relies on Nav2 for trajectory

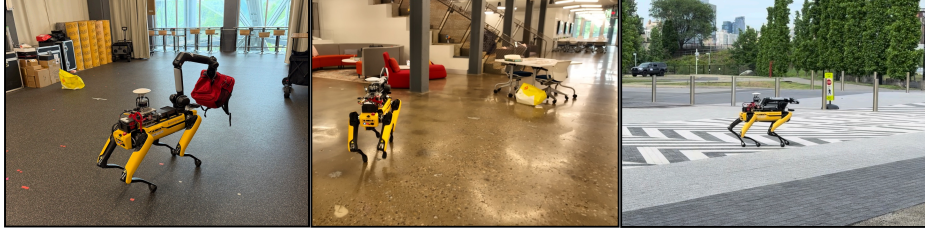


Fig. 4: Experimental snapshots showing the evaluation environments: two indoor settings and one outdoor setting.

planning and the Spot’s internal controller for velocity tracking [20]. Similarly, we use the Spot’s internal SDK for object grasping and placement. We query GPT-5.1 via a network connection; all other processes run onboard.

5.2 Experimental setup

This section describes the experimental environments considered in our work, tasks, and baselines.

Environments. We conduct experiments in two indoor environments and the outside of an office park, which are illustrated in Figure 4. These environments provide an experimental area of roughly 100m^2 , 250m^2 , and $1,000\text{m}^2$.

Tasks. We consider three types of tasks: object disambiguation, functional inference, and occlusion reasoning. Object disambiguation tasks require CLUE to identify a specific description of an object (e.g., a Jansport bag). The feature embeddings of the language-embedded map do not provide a sufficient level of detail to locate the object, and the map may have false positives. CLUE must therefore use the map to gather candidate inspection locations and then verify candidates online. CLUE must then situate this object within the context of the task (e.g. mobile manipulation). Functional inference tasks suggest desired object attributes, such as “I need something to transport equipment for field experiments. Ideally with a perch for my laptop.” CLUE must infer entities that match the functional attributes, e.g. a cart for transport and some supporting structure for a laptop. Finally, occlusion reasoning requires CLUE to recognize when an object of interest may be occluded and resolve that occlusion, as illustrated in Figure 1. These tasks require both manipulation—for occlusion reasoning and acquiring objects of interest—and mobility. We consider a total of fifteen tasks—nine in the laboratory environment, three in a floor of the office building, and three outdoors—and we split these tasks into six object disambiguation, six functional inference, and three occlusion reasoning. We evaluate each task once.

Baselines. We consider two baselines. First, we evaluate against an oracle which is given step-by-step instructions on how to solve the task, but otherwise has the same action space as our method. This baseline estimates an upper bound on task performance. We also compare against NLMaps [5]. This policy has access to the same language-embedded map and action space as our method, but it assumes queries from the map are noiseless. The policy must therefore generate

Method	Success			Actions	Time (s)	LLM Calls	LLM Tok. (k)	VLM Calls	VLM Tok. (k)
	Dis.	Func.	Occl.						
Oracle	83.3%	100.0%	100.0%	4.3±1.0	175.4±62.0	4.4±1.2	2.3±8.0	1.1±0.5	0.9±0.4
CLUE	83.3%	100.0%	66.7%	8.1±4.9	222.9±120.6	6.3±3.6	5.7±5.2	2.9±2.1	2.1±1.7
NLMaps [5]	16.7%	33.3%	0.0%	4.3±1.2	115.2±58.4	3.4±0.8	1.2±4.6	0.9±0.6	0.5±0.3

Table 1: Baseline comparison mean and std. across fifteen real-world experiments, with success broken down by task type: object disambiguation (Dis.), functional inference (Func.), and occlusion reasoning (Occl.).

an open-loop plan, and this baseline measures the importance of closed-loop feedback for contextual information gathering.

5.3 Results

Table 1 reports results across seven metrics: task success rate, the number of actions required to finish a task, the time required to terminate a task, the number of LLM calls, the number of LLM tokens (LLM Tok.), the number of VLM calls, and the number of VLM tokens (VLM Tok.) used. The oracle achieved a 93.3% success rate. Failure occurred when a candidate was mislocalized in the map due to depth reconstruction error, and the policy could not recover despite additional exploration attempts. CLUE achieved an 86.7% success rate averaged over tasks, addressing **Q1**. Resource usage—as measured by the remaining six metrics—is significantly higher. CLUE takes 1.3x as long with nearly 2x as many actions, 1.4x as many LLM calls with 2.5x as many LLM tokens, 2.6x as many VLM calls, and 2.3x as many VLM tokens. NLMaps only achieves a success rate of 20%, addressing **Q2**. The resource usage is comparable or lower than the oracle’s, with the same number of actions on average, 0.7x as long, 0.8x as many LLM calls with nearly half the number of tokens, and similar trends for VLM usage.

CLUE uses more resources (actions, time, LLM/VLM calls and tokens) because it had to resolve false positives and distractors in the map. In contrast, the oracle received ground truth instructions so could directly address task-relevant content. CLUE had two primary failure modes: contextual inference and behavior execution failure. Contextual inference failure came when the policy incorrectly interpreted the task. For example, in response to the task “I need something to carry my robot outside,” it suggested a packing box rather than a cart. In another experiment, the robot failed to grasp a task-relevant object, leading to an execution failure. NLMaps’ primary failure mode was false positives in the initial map query. It assumed this result was correct, so it had no mechanism to recover from false positives (see Figure 5 for an example). Because NLMaps would simply stop after reaching a false positive, rather than trying to explore additional locations, its resource usage is comparatively low.

5.4 Does Information Retrieval on a Map Suffice?

We next evaluate the necessity of active perception as opposed to information retrieval from a pre-built map. We compare CLUE to DAAAM [11], a state-of-the-art scene understanding framework which differs from our approach in two key

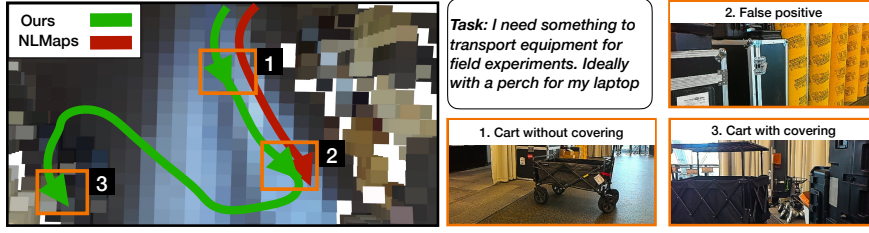


Fig. 5: **Qualitative Results.** Our method sequentially inspects candidates derived from the map, reasoning about functional use until it arrives at a suitable object. NLMaps only visits one candidate, which is a false positive.

Method	Success			LLM Calls	LLM Tok. (k)	VLM Calls	VLM Tok. (k)
	Dis.	Func.	Occl.				
CLUE	66.7%	100.0%	66.7%	5.2±3.1	4.2±4.4	1.9±1.3	1.4±1.2
DAAAM [11]	33.3%	33.3%	0.0%	2.1±0.6	3.4±0.7	5.4±0.5	17.1±2.4

Table 2: Map retrieval mean and std. across nine real-world experiments, with success broken down by task type: object disambiguation (Dis.), functional inference (Func.), and occlusion reasoning (Occl.).

aspects. First, DAAAM forgoes a dense language-embedded map and instead directly constructs a scene graph—explicitly segmenting objects and regions. Second, DAAAM uses a VLM to caption objects and regions observed during map construction. Because DAAAM does not receive a prior task, these captions are descriptive but generic. We compare CLUE to DAAAM across the nine tasks performed in the laboratory environment. We first use the sensory data collected during the experiments from §5.3 to build a map of the environment. We then provide DAAAM’s agentic information retrieval tool with the task specification.

Table 2 reports the mean and standard deviation over five metrics—task success, LLM calls, LLM tokens used, VLM calls, and VLM tokens used. We find that DAAAM only achieves a 22.2% task success rate averaged across tasks, addressing Q3. The primary failure modes were contextual imprecision where the map did not have enough information to disambiguate objects, false negative object detections which prevented task-relevant object retrieval, and the inability to resolve occlusions in the scene. Because CLUE uses an LLM to repeatedly generate new plans, it requires over twice as many LLM calls, with higher variance resulting from variable task length. Interestingly, CLUE requires only 1.2x as many tokens despite a higher number of queries; this is because DAAAM’s agent requires the entire scene graph, which is significantly larger than the symbolic observation space described in §4.1. Finally, because DAAAM is using a VLM to caption all observed objects rather than only ones that are contextually-relevant, it requires over 2.8x as many VLM calls with 12x as many tokens.

5.5 Can LLMs Provide Informative Paths?

We next assess the value of providing the LLM with a plan generated by a TSP (§4.3) by comparing CLUE with an ablated version that does not receive plan

hints (CLUE w/o hint), an LLM that has been prompted to produce a distance-efficient path (Path LLM), and the result from TSP. Each method must plan a path over waypoints randomly sampled from a 10×10 grid, which corresponds to candidates from CLUE’s hypothesis grounding process. We evaluate the ability of each method to produce distance-optimal paths from two to ten waypoints. For “CLUE” and “CLUE w/o hint,” we append the generated points to the observation history from a recorded experiment to mimic deployment. Figure 6 reports the mean and standard deviation of the distance ratio to the TSP solution averaged over 25 trials from two to ten waypoints.

The Path LLM is able to rival the performance of the TSP, which is a result suggested by [14]. Interestingly, without a TSP hint our method’s LLM produces significantly longer paths—over 1.5 times as long for six or more points. This degradation is likely due to the additional system instructions—as outlined in §4.2—which focus the LLM’s attention on making contextual associations in the environment rather than producing distance-efficient paths. Future work is required to develop generation structures that enable both reasoning and efficient path assignments.

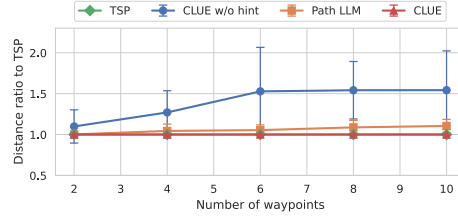


Fig. 6: LLM ability to plan distance-efficient paths (mean and std.)

6 Limitations and Future Work

We note several limitations and areas for future work. The information provided in the language-embedded map may be used for semantic frontier exploration, such as in [24], and this is a promising avenue for future work. The language-embedded map is also memory intensive, with maps for the outdoor environment reaching 100 GB. Existing work suggests that memory requirements may be reduced via submap caching, which stores inactive portions of the map to disk [24].

Finally, we only consider a cloud-based model (i.e. GPT-5.1). Future work may evaluate the performance of CLUE using smaller open-source models or consider distillation methods in order to produce on-device policies [27].

7 Conclusion

This work addresses the problem of resolving contextual uncertainty for underspecified tasks in natural language, where the required subgoals as well as the relevant semantic, functional, and metric information must be inferred. We propose CLUE, a closed-loop framework that uses an LLM-enabled policy to infer hypotheses, an approximation of belief for open-vocabulary environments, given a task description and observations. CLUE grounds these hypotheses into concrete actions using a language-embedded map, and verifies hypotheses through active interaction with the environment. These interactions produce observations, which the policy uses to update its hypotheses and iteratively replan. Across real-world

experiments in indoor and outdoor environments that require object disambiguation, functional affordance reasoning, and occlusion handling, CLUE rivals the performance of an oracle with ground truth instructions (within 7 percentage points task success), outperforms an LLM-enabled planner without closed-loop information gathering by a 4x margin and a mapping then querying approach by a 3x margin, while being significantly more VLM token efficient.

Acknowledgments. We gratefully acknowledge support from DCIST CRA W911NF-17-2-0181, NSF Grant CCF-2112665, and the NSF GRFP.

References

1. Alama, O., Jariwala, D., Bhattacharya, A., Kim, S., Wang, W., Scherer, S.: Radseg: Unleashing parameter and compute efficient zero-shot open-vocabulary segmentation using agglomerative models. arXiv preprint arXiv:2511.19704 (2025)
2. Alama, O., et al.: Rayfronts: Open-set semantic ray frontiers for online scene understanding and exploration. IROS (2025)
3. Arnob, R.I., Merlin, M., Paudel, A., Hedegaard, B., Konidaris, G., Stein, G.J.: Effective task planning with missing objects using learning-informed object search. arXiv preprint arXiv:2602.11468 (2026)
4. Black, K., et al.: $\pi_{0.5}$: a vision-language-action model with open-world generalization. In: CoRL (2025)
5. Chen, B., et al.: Open-vocabulary queryable scene representations for real world planning. In: ICRA 2023
6. Chen, Y., Arkin, J., Zhang, Y., Roy, N., Fan, C.: Scalable multi-robot collaboration with large language models: Centralized or decentralized systems? In: ICRA (2024)
7. Curtis, A., et al.: Llm-guided probabilistic program induction for pomdp model estimation. In: CoRL 2025
8. Dai, Z., Asgharivaskasi, A., Duong, T., Lin, S., Tzes, M.E., Pappas, G., Atanasov, N.: Optimal scene graph planning with large language model guidance. In: ICRA 2024
9. Deepmind: Rt-2: Vision-language-action models transfer web knowledge to robotic control. In: 2023 Conference on Robot Learning
10. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (1996)
11. Gorlo, N., Schmid, L., Carlone, L.: Describe anything anywhere at any moment (2025). URL <https://arxiv.org/abs/2512.00565>
12. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2024)
13. He, I.X., Tellex, S., Liu, J.X.: Legs-pomdp: Language and gesture-guided object search in partially observable environments. In: HRI (2026)
14. Huang, Z., Shi, G., Sukhatme, G.S.: Can large language models solve robot routing? In: International Symposium on Robotics Research (ISRR) (2024)
15. Kannan, S.S., Venkatesh, V.L.N., Min, B.C.: Smart-llm: Smart multi-agent robot task planning using large language models. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2024)
16. Kim, Y., Arora, R., Martín-Martín, R., Stone, P., Abbatematteo, B., Sung, Y.: Large-language-model-guided state estimation for partially observable task and motion planning. In: ICRA (2026)
17. Li, X., et al.: Llara: Supercharging robot learning data for vision-language policy. In: International Conference on Learning Representations (2025)
18. Liang, K., Zhang, Z., Fisac, J.F.: Introspective planning: Aligning robots’ uncertainty with inherent task ambiguity. In: NeurIPs 2024

19. Liu, X., Toroghi, A., Liang, J., Courtis, D., Li, R., Pesaranghader, A., Kim, J., Sadhu, T., Jeon, H., Sanner, S.: Natural language PDDL (NL-PDDL) for open-world goal-oriented commonsense regression planning in embodied AI. In: ICLR (2026). Poster presentation
20. Macenski, S., Martin, F., White, R., Ginés Clavero, J.: The marathon 2: A navigation system. In: International Conference on Intelligent Robots and Systems (IROS) (2020)
21. Mandi, Z., Jain, S., Song, S.: Roco: Dialectic multi-robot collaboration with large language models. In: ICRA (2024)
22. Nayak, S., et al.: MAP-THOR: Benchmarking long-horizon multi-agent planning frameworks in partially observable environments. In: Multi-modal Foundation Model meets Embodied AI Workshop @ ICML2024 (2024)
23. Oloo, A., Altaweel, Z., Hayamizu, Y., Liu, P., Ding, Y., Amiri, S., Yang, H., Kaminski, A., Esselink, C., Paxton, C., Zhang, X., Zhang, S.: Robot planning and situation handling with active perception. In: IROS (2026)
24. Ong, D., Tao, Y., Murali, V., Spasojevic, I., Kumar, V., Chaudhari, P.: Atlas navigator: Active task-driven language-embedded gaussian splatting. arXiv preprint arXiv:2502.20386 (2025)
25. OpenAI: Openai gpt-5 system card (2025)
26. Ravichandran, Z., Cladera, F., Prabhu, A., Hughes, J., Murali, V., Taylor, C., Pappas, G.J., Kumar, V.: Heterogeneous robot collaboration in unstructured environments with grounded generative intelligence. IROS 2026
27. Ravichandran, Z., Hounie, I., Cladera, F., Ribeiro, A., Pappas, G.J., Kumar, V.: Distilling on-device language models for robot planning with minimal human intervention. Conference on Robot Learning (CoRL) (2025). URL <https://arxiv.org/abs/2506.17486>
28. Ravichandran, Z., Murali, V., Tzes, M., Pappas, G.J., Kumar, V.: Spine: Online semantic planning for missions with incomplete natural language specifications in unstructured environments. In: ICRA (2025)
29. Ren, A.Z., Clark, J., Dixit, A., Itkina, M., Majumdar, A., Sadigh, D.: Explore until confident: Efficient exploration for embodied question answering. In: RSS (2024)
30. Ren, A.Z., et al.: Robots that ask for help: Uncertainty alignment for large language model planners. In: CoRL (2023)
31. SciPy 1.0 Contributors: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* **17**, 261–272 (2020)
32. Srivastava, S., Immerman, N., Zilberstein, S.: Using abstraction for generalized planning. In: ICAPS Workshop on AI Planning and Learning (2007)
33. Sun, L., et al.: Interactive planning using large language models for partially observable robotics tasks (2023)
34. Tang, W., et al.: Tru-POMDP: Task planning under uncertainty via tree of hypotheses and open-ended POMDPs. In: NeurIPS (2025)
35. Tao, Y., Ong, D., Murali, V., Spasojevic, I., Chaudhari, P., Kumar, V.: Rt-guide: Real-time gaussian splatting for information-driven exploration. IEEE RA-L (2025)
36. Wang, J., Ly, K.T., Cloete, J., Tsarakis, N., Havoutis, I.: Vision-language-policy model for dynamic robot task planning. arXiv preprint arXiv:2512.19178 (2025)
37. Wu, Y., et al.: Hierarchical llms in-the-loop optimization for real-time multi-robot target tracking under unknown hazards. In: IEEE MRS (2025)
38. Yokoyama, N., Ha, S., Batra, D., Wang, J., Bucher, B.: Vlfm: Vision-language frontier maps for zero-shot semantic navigation. In: 2024 ICRA (2024)
39. Zhao, L., et al.: Seeing is believing: Belief-space planning with foundation models as uncertainty estimators. arXiv 2504.03245 (2025)
40. Zhou, B., Zhang, Y., Chen, X., Shen, S.: Fuel: Fast uav exploration using incremental frontier structure and hierarchical planning. IEEE RA-L (2021)