

Belief-driven Interactive Object Search in Partially Observable Settings

by

Lucy Cai

S.B. Computer Science and Engineering, Massachusetts Institute of Technology, 2025

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER
SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2026

© 2026 Lucy Cai. All rights reserved.

The author hereby grants to MIT a nonexclusive, worldwide, irrevocable, royalty-free license to exercise any and all rights under copyright, including to reproduce, preserve, distribute and publicly display copies of the thesis, or release the thesis under an open-access license.

Authored by: Lucy Cai
Department of Electrical Engineering and Computer Science
July 24, 2026

Certified by: Leslie Pack Kaelbling
Professor of Computer Science and Engineering, Thesis Supervisor

Certified by: Aditya Agarwal
Ph.D. Student, Thesis Supervisor

Accepted by: Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Belief-driven Interactive Object Search in Partially Observable Settings

by

Lucy Cai

Submitted to the Department of Electrical Engineering and Computer Science
on July 24, 2026 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER
SCIENCE

ABSTRACT

Interactive object search— the task of finding and retrieving objects in unfamiliar, partially observed environments— is a foundational capability for general-purpose robots. Almost every useful task a robot might perform, from making a cup of tea to packing a suitcase, begins with locating the relevant objects. Classical POMDP-based approaches to this problem maintain a principled spatial belief state based on occupancy but lack the semantic priors necessary for efficient, human-like search. Recent foundation-model systems supply those priors but at the cost of the explicit belief, leaving no quantitative basis for decision-making. We present an end-to-end system that combines both: the robot maintains an explicit 3D voxel belief over the target’s location, informed by both occupancy and a vision-language model, and selects actions based on a heuristic grounded in the belief state. By using the vision-language model only to inform our semantic prior rather than to make decisions for the robot, our system preserves an explicit, interpretable belief state that any individual decision can be traced back to. We deploy the system on a Rainbow Robotics RBY1 humanoid and successfully demonstrate end-to-end interactive object search in an office-like environment.

Thesis supervisor: Leslie Pack Kaelbling

Title: Professor of Computer Science and Engineering

Thesis supervisor: Aditya Agarwal

Title: Ph.D. Student

Acknowledgments

I am deeply grateful to my mentor Aditya for his guidance and wisdom throughout this year, without which I would not have learned nearly a fraction of what I did. I am thankful to my supervisor Leslie for always posing fascinating research questions, including the one this thesis is based on. Thank you to Maggie, Sahit, and everyone else at LIS for our daily game sessions and for all the fun conversations during the late nights. I'm thankful to Ben for being a grounding force in my life— you inspire me to reach for the stars. Above all, I'm grateful to my parents, my sister Ellie, and the rest of my family, for the sacrifices they made for me to get here.

Contents

Acknowledgments	5
<i>List of Figures</i>	9
<i>List of Tables</i>	11
1 Introduction	13
1.1 Object Search as a Foundational Skill	13
1.2 A Motivating Example	13
1.3 Contributions	14
2 Related Work	15
2.1 Classical POMDP Methods	15
2.2 Foundation Model Systems	17
2.3 Our Approach	18
3 Methodology	19
3.1 Problem Formulation	19
3.2 System Overview	20
3.3 Semantic Map	20
3.3.1 Per-Frame Detection and Segmentation	20
3.3.2 Object Representation	20
3.3.3 Merging Detections Across Frames	21
3.4 Occupancy Map	21
3.4.1 Voxel States	22
3.5 Belief Map	22
3.5.1 Semantic Branch: Per-Frame VLM Scoring	23
3.5.2 Geometric Branch: Per-Voxel State Prior	24
3.5.3 Bayesian Fusion in Log-Odds	25
3.5.4 Combined Voxel Field	25
3.6 Action Selection	26
3.6.1 Picking an Object to Interact with	26
3.6.2 Selecting a Skill via the VLM	26
3.7 Skill Library	27

4	System Implementation	29
4.1	Robot Platform: RBY1	29
4.2	Capture Observation Loop	29
4.3	Classical Navigation	29
4.4	Engineered Skills	30
4.4.1	look_at	30
4.4.2	open_cabinet	31
4.4.3	open_drawer	31
4.4.4	pick_from_table	31
4.4.5	pick_from_shelf	31
4.4.6	pick_from_drawer	32
4.5	TiPToP Adaptation	32
4.6	Learned Skill Policies	33
5	Results	35
5.1	Preliminary Deployment Study	35
5.2	Next Steps	37
6	Conclusion	39
6.1	Future Work	39
6.2	Conclusion	39
	References	41

List of Figures

3.1	System overview.	20
3.2	Occupancy map for a representative scene, with the underlying scene point cloud overlaid for context. Left: OCC voxels (red). Right: OCCL voxels (yellow). FREE and UNEXP voxels are omitted for clarity.	22
3.3	Belief map for an apple-search query in a representative scene. (a) The input RGB scene. (b) Per-frame Gemini scores overlaid on the same scene. (c) Voxel visibility multipliers overlaid on the same scene. (d) The fused belief field $\beta_t(v)$ obtained by combining the two branches.	23
3.4	Examples of VLM action inference. Each panel shows an annotated image of the highest-belief object (green bounding box, class label, current belief value) alongside the action the VLM selected. From left to right: an open box seen from a distance triggers <code>look_at</code> ; a closed drawer triggers <code>open_drawer</code> ; a closed cabinet triggers <code>open_cabinet</code>	26
4.1	Path planning over the inflated occupancy grid. Orange voxels are inflated obstacles, gray is free space, and the green line is the smoothed A* path between the robot's start pose and the target.	30
4.2	The robot executing each of the six engineered skills. Top row, left to right: <code>look_at</code> on a cluttered table, <code>open_cabinet</code> , <code>open_drawer</code> . Bottom row, left to right: <code>pick_from_table</code> , <code>pick_from_shelf</code> , <code>pick_from_drawer</code>	31
5.1	The deployment scene from the robot's starting pose, with the two candidate target locations annotated in red (the cardboard box on the snack table) and blue (the office-supply drawers).	36
5.2	Belief-annotated observations from the apple search under the unpredictable placement condition. Contradicting evidence on each frame redirects belief mass to the next most plausible candidate.	37

List of Tables

3.1	Primitives in the skill library, grouped by category.	27
5.1	Number of dispatched skills (excluding the terminal retrieval action) per episode in the preliminary deployment study.	36

Chapter 1

Introduction

1.1 Object Search as a Foundational Skill

Almost every useful task a household or service robot might do begins with finding objects. Making tea requires gathering a kettle, a mug, and a tea bag. Bussing a table requires finding the plates and cups stored away in the kitchen. Packing a suitcase requires finding clothes, toiletries, and other necessities across multiple rooms of the house. Even tasks that look like manipulation problems— assembling furniture, for example— start with collecting the parts and tools the job requires.

In settings where we might want a robot to manipulate objects it doesn't already have knowledge of, object search is essential. A robot that cannot find the kettle cannot make the tea, no matter how skilled it is at pouring. Interactive object search— including cases where finding the target requires looking inside drawers or cabinets— is therefore a foundational capability for embodied agents.

1.2 A Motivating Example

Consider a robot tasked with restocking an office supply room of which it has no prior knowledge. A delivery of staplers, paper reams, sticky notes, and ink cartridges has arrived, and each item has to be placed wherever similar items are already stored. Some items are visible the moment the robot enters: paper reams on an open shelf, sticky notes on a counter. Others are inside closed cabinets or drawers that have to be opened to reveal their contents.

A person, in this situation, would use what they know about offices to complete this task— that ink cartridges are located near printers, for example— to inspect the most plausible regions first. They would not open every drawer in the room at random, but only do so when they deemed it likely that a target object might be inside. The amount of time it takes, and the number of actions taken, is what separates an efficient search from an exhaustive one.

This office supply room is one instance of a broader problem setting. The robot operates in a partially observed environment, in which finding the target may require both active perception as well as active manipulation with occluders. The robot's task is to find the target object as efficiently as possible. This is the problem setting which this thesis tackles.

1.3 Contributions

This thesis develops an end-to-end system for interactive object search that combines semantic and geometric knowledge into an explicit 3D belief state, and uses it to select the best action in a cost-aware manner. We frame the object search problem in terms of a discrete but dynamic set of hypothesis about where the objects could be, e.g., in chunks of space that are contained within observed objects, or within chunks of unobserved space. Concretely, the contributions are:

- **Belief-driven action selection.** A vision-language model provides semantic knowledge via scoring observed regions as possible locations for the target. These scores are fused with occupancy values into a spatial belief over a 3D voxel map. A cost-aware greedy heuristic deterministically decides the next action based on the highest probability belief mass.
- **A closed-loop system for end-to-end object search.** We demonstrate full end-to-end interactive object search on a mobile bimanual humanoid: the robot navigates, opens drawers and cabinets, and interacts with shelves, tables, and containers to locate and retrieve targets— all in real time, in a single closed loop from observation to retrieval.

Chapter 2

Related Work

Prior work in interactive object search can be broadly divided into two categories: classical POMDP-based methods, which focus on reasoning under uncertainty, and recent foundation-model-driven systems, which rely on VLMs for planning. This chapter surveys both, discusses the limitations of each, and then explains how our system combines the best of both without inheriting the limitations of either.

2.1 Classical POMDP Methods

Early work on object search is largely grounded in the framework of Partially Observable Markov Decision Processes (POMDPs), which provide a structured way to reason under uncertainty by maintaining a belief over possible world states. In this formulation, object search is treated as a sequential decision-making problem where the robot actively selects observations to reduce uncertainty about its environment.

Formally, a POMDP is defined as an 8-tuple

$$\mathcal{M} = \langle S, A, O, T, Z, R, b_0, \lambda \rangle,$$

where:

- S is the set of all possible states of the world, including the robot and its environment.
- A is the set of all actions the robot can take.
- O is the set of all observations the robot can receive.
- $T(s, a, s') = \Pr(s' \mid s, a)$ is the transition model, defining the probability of reaching state s' after taking action a in state s .
- $Z(s', a, o) = \Pr(o \mid s', a)$ is the observation model, defining the probability of observing o after taking action a and reaching state s' .
- $R(s, a, s')$ is the reward function, specifying the reward received after transitioning from s to s' via action a .

- b_0 is the robot’s initial belief over the state space.
- $\lambda \in (0, 1)$ is the discount factor.

The agent does not observe the true state directly. Instead, it maintains a belief b_t —a probability distribution over states— and updates it via Bayes’ rule after each action–observation pair.

Object search has been formulated as a POMDP in a number of ways, each addressing a different aspect of the problem. Wandzel et al. [1] introduce the Object-Oriented POMDP (OO-POMDP), which factors the joint belief over k objects across n cells into per-object marginals rather than maintaining a monolithic joint distribution. The factorization keeps belief updates tractable as the number of objects grows and supports encoding correlations between objects, such as forks tending to appear near plates. These correlations, however, are specified by hand or estimated from data collected in the same environment class, which limits transfer to scenes the system has not been designed for.

Zheng et al. [2] extend this line of work with multi-resolution planning over a hierarchy of grids, allowing the planner to reason coarsely over large regions and refine only when local evidence justifies a closer look. The hierarchy mitigates the scaling cost of 3D belief representations and enables online planning in larger environments than flat-grid formulations admit. The method assumes a known, mapped workspace and treats the search as a pure perception problem, with no manipulation actions to reveal occluded regions.

GENMOS [3] generalizes the formulation to arbitrary 3D scenes using an octree as the belief representation, sampling viewpoints in high-belief regions and running Monte Carlo Tree Search to select the next observation. The system is deployed end-to-end on real hardware and handles multi-object targets across rooms, making it the closest prior system-level demonstration to ours. Action selection is restricted to viewpoint changes, however; the robot does not open containers or otherwise manipulate the scene to expose hidden objects, so search effectively terminates at the boundary of any closed receptacle.

Chen and Kurniawati [4] target a harder regime in which the map itself is partially unknown, so the robot must reason jointly about exploring unmapped space and searching within mapped regions. They propose an online POMDP planner that handles this hybrid uncertainty without assuming a complete prior map, expanding the set of environments in which principled belief-space planning is feasible. The cost of this generality is computational: tree-search-based planning over both map and target uncertainty requires nontrivial per-step compute, and the priors over object location are not derived from semantic knowledge of the scene.

Limitations

These methods provide a calibrated spatial belief and a principled action-selection criterion, with some support for semantic priors, but in limited forms. The OO-POMDP formulation admits object correlations such as forks tending to appear near plates, but these correlations have to be specified by hand or estimated from data collected in the target environment class. Wong et al. [5] go further: their manipulation-based active search system combines object co-occurrence priors with geometric spatial constraints, reasoning jointly about which

container is semantically plausible and whether the target can spatially fit in the remaining unobserved volume, and uses manipulation actions to reveal occluded objects rather than relying on viewpoint changes alone. Their semantic priors, however, are drawn from a fixed object-type vocabulary with co-occurrence statistics learned from a training set of example containers, which does not scale to the open-vocabulary setting in which a robot is asked to find arbitrary objects in unfamiliar rooms. When semantic information is unavailable or unreliable for a particular target, the system falls back on the geometry-driven heuristic, which tends to favor regions like the largest unexplored volume or the nearest occluder. A robot searching for dish soap in an unfamiliar kitchen, for example, might prioritize peering inside the refrigerator simply because it presents the largest unresolved occluded volume in the scene, even though semantically it is an unlikely place to store dish soap.

The action space in much of this work is also restricted to viewpoint changes, with relatively few exceptions [5] that explicitly model manipulation of occluders. For the bulk of methods in this family, volumes hidden behind closed barriers are unreachable in principle.

2.2 Foundation Model Systems

Recent advances in foundation models have led to a paradigm shift in robotic perception and decision-making. The introduction of large vision-language models effectively reframed the field’s approach to object search: rather than designing observation models, priors, and planners from scratch, recent systems use foundation models for nearly every part of the task—interpreting scenes, scoring candidate locations, generating action plans, and performing commonsense reasoning across perception and manipulation. This shift has made it possible to handle open-vocabulary targets and unfamiliar environments at a scale that previous systems could not.

Many foundation-based approaches have focused on object search through active perception alone. SEMEXP [6] learns a goal-oriented semantic exploration policy over a top-down map, using semantic segmentation to bias exploration toward regions where the target category is likely to appear. VLFM [7] uses a VLM to score frontier points by their likelihood of containing the target, integrating language-derived priors directly into a frontier-based exploration loop. SEEK [8] estimates target probabilities across the spatial elements of a dynamic scene graph, allowing the robot to reason about the scene at the level of objects and rooms rather than raw geometry. These systems use semantic signals effectively but assume no active manipulation.

A second cluster of work introduces manipulation into the search loop, allowing the robot to open containers and move occluders rather than relying on viewpoint changes alone. RoboEXP [9] uses a tabletop manipulator to demonstrate that LLMs can serve as high-level planners by grounding goals into sequences of executable robot skills, including opening drawers and moving objects to expose hidden geometry. The same authors extend this idea in CuriousBot [10], a mobile manipulator that uses an Action-Conditioned Scene Graph (ACSG) to maintain its understanding of the scene as it interacts with articulated objects. However, they operate over a static scene graph built from initial observations with no way to capture additional images as the search progresses.

Closest to our setting is MOMA-LLM [11], which combines a dynamic scene graph with

an LLM planner. Its task and action space are essentially the same as ours: a mobile manipulator searching an unknown environment with both perception and manipulation actions available. The architectural choice it makes, however, is to delegate action selection entirely to the LLM— the planner is asked, at each step, which container to open or which region to inspect next, and the LLM’s output is executed directly. There is no explicit belief representation over target location, no quantification of how much information a candidate action would reveal, and no notion of action cost beyond what the LLM can be prompted to consider implicitly.

Limitations

Foundation-model methods supply the common sense that classical POMDP methods lack, but in doing so they discard two things. First, the explicit belief over space disappears, in favor of an implicit belief over objects. In abstracting the notion of space into a collection of individual objects, these systems lose the ability to reason over free space— space that might not be associated with any particular object. If an object falls under the table or gets wedged between the couch and the wall, it is no longer anchored to a specific node in the scene graph, and becomes unable to be found. Also, the lack of representation of space as a volume means that there is no quantitative measure of how much of the workspace remains unexplored, which means that systems like these have no basis for answering questions of the type “should I keep exploring this known region, or take a new observation?”

Second, the formal POMDP planning disappears, in favor of the informal reasoning of foundation models. Different prompts, model versions, and decoding temperatures produce different decisions, so the resulting policies are neither reproducible across runs nor interpretable after the fact. When the robot looks in the wrong place, there is no underlying belief or cost computation to point to as the source of the failure, only the model’s output for that particular prompt on that particular day.

2.3 Our Approach

Our system combines both geometric and semantic priors in a single explicit belief state. We maintain a 3D voxel belief that is informed not only by occupancy geometry but also by per-frame VLM scores assigning each observed object a likelihood of containing the target. Action selection is then a deterministic, cost-aware rule over this belief, with the VLM acting only as a perception module rather than a decision-maker. This preserves the explicit spatial belief that classical POMDP methods rely on without requiring full POMDP planning, and it incorporates the semantic priors that foundation-model methods rely on without delegating action selection to a VLM. Every decision can be traced to a value in the belief and a cost in the skill library, which makes the resulting policy both interpretable and reproducible.

Chapter 3

Methodology

3.1 Problem Formulation

This section defines the interactive object search problem that the rest of the thesis addresses.

The robot is placed in a room that it has no prior knowledge of. The room may contain furniture (e.g., tables, shelves, cabinets) as well as free objects. The environment is partially observable, with regions falling into three levels of observability: (1) directly visible from the robot’s starting pose, (2) occluded but observable through active perception (e.g., inside an open container), and (3) hidden behind physical barriers that the robot must manipulate to expose. There is no prior map of the scene; the robot must build whatever representation it needs from its own observations.

At each step the robot takes one of two kinds of actions: an *active observation* $a \in \mathcal{A}_{\text{obs}}$ (moving to a new viewpoint to bring a region into view) or an *active manipulation* $a \in \mathcal{A}_{\text{manip}}$ (physically interacting with the environment). Manipulations alter the environment state; observations do not.

Given a natural-language query q (e.g., “*find me an apple*”), the robot’s task is to locate *and retrieve* an instance of the queried category. Locating the target alone is not enough: the system must additionally pick the object up. The task terminates successfully when the robot has the target in its gripper. Formally, the robot must select a policy π that maps observation histories to actions in $\mathcal{A}_{\text{obs}} \cup \mathcal{A}_{\text{manip}}$, where each action a carries an execution cost $c(a)$. The objective is to minimize the expected total action cost until successful retrieval:

$$\min_{\pi} \mathbb{E} \left[\sum_{t=0}^T c(a_t) \right],$$

where T is the timestep at which the target is in the gripper.

We make three assumptions about the problem setting:

1. **Single room.** The search is constrained to a single room.
2. **Target presence.** At least one instance of the queried category exists somewhere in the workspace.
3. **Static environment.** Objects do not move except as a result of the robot’s own actions.

3.2 System Overview

The system consists of five components that form a closed loop: a *semantic map* that tracks discovered object instances (§3.3), an *occupancy map* that records observed and unobserved space (§3.4), a *belief map* over where the target might be (§3.5), *action selection* that decides what to do next (§3.6), and a *skill library* that executes the selected action (§3.7). At each step, the robot captures an RGB-D observation, updates the semantic, occupancy, and belief maps, selects an action greedily, and executes it. The loop terminates when the robot has the target object in its gripper.

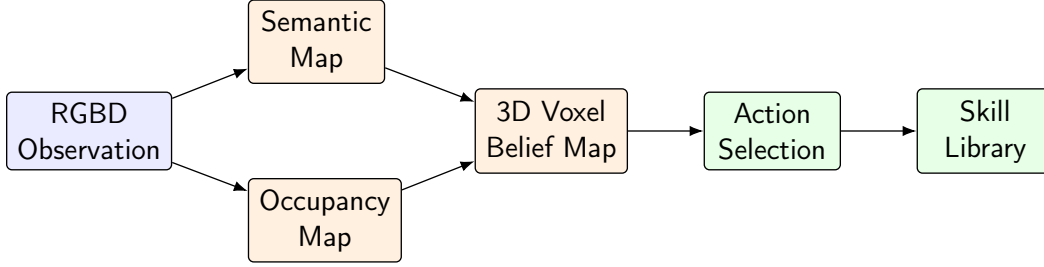


Figure 3.1: System overview.

3.3 Semantic Map

The semantic map is a 3D scene graph that the robot builds incrementally from RGB-D observations. It serves two roles. First, it supports target matching: given the query q , the system uses CLIP similarity between q and each object in the semantic map, as well as VLM verification, to check whether the target has already been observed. Second, the objects in the semantic map define the units over which the belief in Section 3.5 is scored.

3.3.1 Per-Frame Detection and Segmentation

Each new RGB-D frame is processed through an open-vocabulary object detector to produce 2D bounding boxes for all visible objects. The particular detector is not fixed; we use Grounding DINO [12] as the default backend and have also tested YOLO-World [13] and Gemini’s detection endpoint. Each box is then passed to SAM [14] as a prompt to obtain an instance segmentation mask. Combined with the depth image and camera pose, each (box, mask) pair yields a segmented 3D point cloud in world coordinates. We additionally extract a CLIP descriptor [15] for each detection by feeding its cropped image to a CLIP image encoder.

3.3.2 Object Representation

The map maintains a persistent set of merged instances

$$\mathcal{O}_t = \{o_1, o_2, \dots, o_{N_t}\}.$$

Each instance $o \in \mathcal{O}_t$ stores four pieces of information:

- a CLIP descriptor $f^{\text{CLIP}}(o)$, computed as the mean of the per-frame CLIP descriptors of all detections that have been merged into o ;
- a 3D bounding box $\mathcal{B}(o)$ enclosing the accumulated segmented point cloud of o ; and
- a global identifier that persists across frames and is mapped to from per-frame in-frame identifiers via a merging table.

3.3.3 Merging Detections Across Frames

When a new frame is processed, each new detection is either merged into an existing instance in $\mathcal{O}_t$ or used to create a new one. Let d denote a new detection and o an existing instance. The decision is based on a similarity score combining two signals:

- a *spatial similarity* $s_{\text{spatial}}(d, o) \in [0, 1]$, defined as the fraction of points in d 's segmented point cloud whose nearest neighbor in o 's point cloud is within $\varepsilon = 2.5$ cm. This measure is asymmetric, which is appropriate here: a small detection that lands inside a larger existing instance scores close to 1, whereas the standard 3D Intersection over Union of their bounding boxes would be small.
- a *visual similarity* $s_{\text{visual}}(d, o) \in [-1, 1]$, defined as the cosine similarity between the detection's CLIP descriptor and the instance's CLIP descriptor.

The two are aggregated as

$$s_{\text{agg}}(d, o) = (1 + \rho) s_{\text{spatial}}(d, o) + (1 - \rho) s_{\text{visual}}(d, o),$$

where $\rho \in [-1, 1]$ controls the balance between geometry and appearance; we use $\rho = 0$, so the two signals contribute equally. A detection is merged into the existing instance with the highest aggregated similarity if that similarity exceeds a threshold τ_{merge} , and otherwise a new instance is created. We use $\tau_{\text{merge}} = 1.2$, which corresponds, with $\rho = 0$, to $s_{\text{spatial}} + s_{\text{visual}} > 1.2$ —for example, a spatial overlap of 0.5 together with a visual similarity of 0.7. When a merge occurs, the instance's point cloud is updated by union with the detection's point cloud, the bounding box is recomputed from the union, and the CLIP descriptor is updated as a running average.

3.4 Occupancy Map

In parallel with the semantic map, the robot maintains an occupancy map that records, for each region of the workspace, whether it has been observed and whether it is occupied by observed point-cloud points. The occupancy map serves as the source of the geometric component of the belief: a closed container whose interior is OCCL gets a boost in belief, which an object whose interior has been ruled out as FREE gets no such boost. This is further formalized in Section 3.5.

3.4.1 Voxel States

We represent the workspace as a 3D voxel grid in which each voxel falls into one of four states:

- UNEXP (unexplored): the voxel has never fallen inside any camera frustum.
- FREE: a depth ray from some camera pose has passed through the voxel without terminating, confirming that it is empty.
- OCC (occupied): a depth measurement terminated at the voxel, indicating a surface.
- OCCL (occluded): the voxel has fallen inside a camera frustum but lies behind an OCC voxel along every ray that has reached it—e.g., the interior of a closed cabinet whose exterior has been observed.

The occupancy map is updated incrementally on each new RGB-D frame. For every depth pixel (u, v) with a finite, in-range measurement d , the corresponding 3D point is back-projected into world coordinates. The endpoint voxel is marked OCC; voxels along the ray from the camera origin to the endpoint are marked FREE; voxels beyond the endpoint inside the frustum are marked OCCL.

As the robot observes more of the scene, voxels transition between states under the precedence rule OCCUPIED $>$ FREE $>$ OCCLUDED $>$ UNEXPLORED. This reflects how OCCUPIED and FREE are the most definite occupancy states, while a voxel that was unexplored in one observation can become occluded in another, and a voxel that was occluded in one observation can become occupied or free in another.

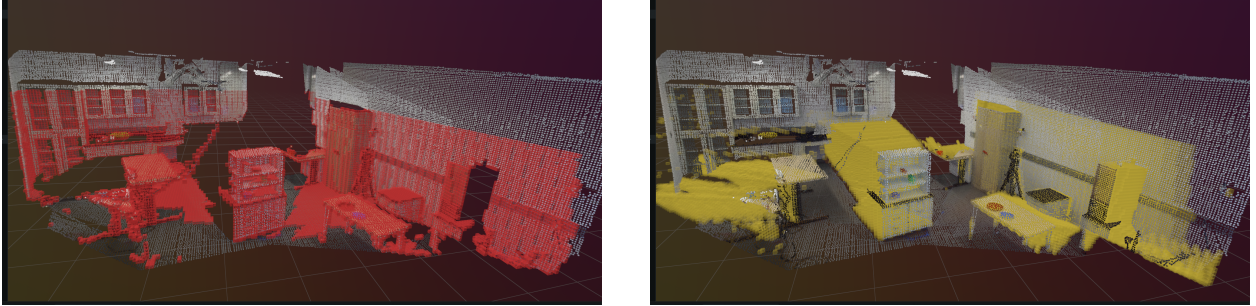


Figure 3.2: Occupancy map for a representative scene, with the underlying scene point cloud overlaid for context. Left: OCC voxels (red). Right: OCCL voxels (yellow). FREE and UNEXP voxels are omitted for clarity.

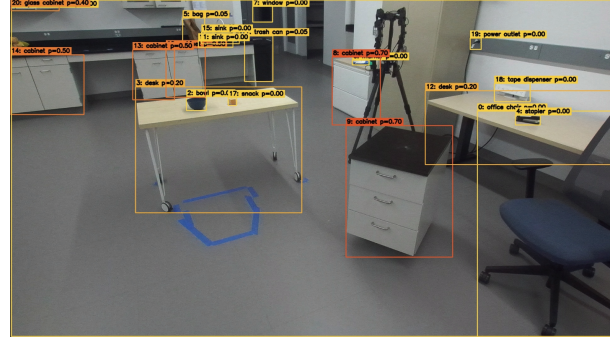
3.5 Belief Map

The belief map is an online, voxel-resolved estimate of where the target q might be. It combines a *semantic* signal from a vision-language model (which objects are plausible locations for the target) with a *geometric* signal from the occupancy map (which candidates still have unobserved interior left to search). The two are fused on the merged object instances of the

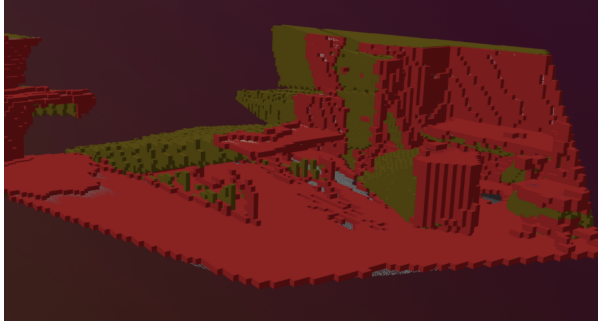
semantic map, yielding a per-object belief $\ell_t(o)$ and a voxel-level field $\beta_t(v)$ used downstream by the planner.



(a) Input scene



(b) Semantic branch



(c) Geometric branch



(d) Fused belief

Figure 3.3: Belief map for an apple-search query in a representative scene. (a) The input RGB scene. (b) Per-frame Gemini scores overlaid on the same scene. (c) Voxel visibility multipliers overlaid on the same scene. (d) The fused belief field $\beta_t(v)$ obtained by combining the two branches.

3.5.1 Semantic Branch: Per-Frame VLM Scoring

At each frame the system renders an annotated image $\tilde{I}_t$ in which each merged object’s 2D extent is labeled with a numeric in-frame identifier i and its class name (e.g., 7: `cabinet`, 16: `shelf`). Conditioned on $\tilde{I}_t$, the user query q , and a system prompt described below, the VLM (Gemini) emits an *independent* probability per identifier,

$$p_i^{(t)} = \text{VLM}(\tilde{I}_t, q, i) \in [0, 1],$$

interpreted as the probability that at least one instance of category q is currently inside, behind, or underneath the object labeled i . Identifiers are mapped back to global instances via the semantic map’s merging table, yielding a per-object probability $p_o^{(t)}$ for every observed $o \in \mathcal{O}_t$.

The scores are not normalized across identifiers. The model is asked to make independent judgments per object, so multiple plausible candidates in the same scene each retain large

mass— a fridge and a pantry can both score 0.7 for an apple query without being forced to compete.

Prompt design. The system prompt is built around two ideas. First, a *common-sense storage prior*: the VLM is asked to score every object as a plausible hiding or storage site for the target given typical human placement habits in the relevant environment (paper reams in supply cabinets, food in kitchens, books on shelves). The prompt explicitly states that the robot is an interactive agent that can open closed containers and reposition to inspect occluded regions, so closed containers receive non-trivial mass even when only their exteriors are visible— a crucial property for partially observed scenes.

Anchoring probabilities to a discrete scale. To stabilize calibration across frames and model versions, the prompt anchors probability values against a discrete scale:

Probability	Interpretation
0.9	Confident this object contains an instance
0.7	Likely contains an instance
0.5	Plausible but uncertain
0.2	Unlikely but not impossible
0.05	Very unlikely
0.0	Essentially impossible

The prompt explicitly forbids artificial concentration or spreading of mass: if many visible objects are plausible containers, all of them should score high; if none is plausible, all should score low. The discrete scale matters for two reasons. First, free-form numerical outputs from VLMs drift across prompts and model versions, and anchoring against a small set of canonical values reduces the drift. Second, the anchors are interpretable to a downstream user, so the belief can be inspected and debugged without running the full pipeline.

3.5.2 Geometric Branch: Per-Voxel State Prior

Independently of the VLM, the occupancy map supplies a per-voxel log-odds prior $\ell_{\text{geom}}(v)$ that depends only on the voxel’s current occupancy state. The prior is a fixed lookup table:

State	ℓ_{geom}	$\sigma(\ell_{\text{geom}})$
FREE	−6.91	0.001
OCCUPIED	−4.00	0.018
OCCLUDED	−0.85	0.30
UNEXPLORED	−2.20	0.10

These values reflect each state’s geometric plausibility as a host for the target. OCCLUDED voxels receive the highest prior since the robot has confirmed something is hidden behind a surface. UNEXPLORED voxels receive the next highest prior since the robot has neither positive nor negative evidence about the space. Finally, FREE and OCCUPIED voxels receive the lowest priors since the robot has confirmed that something else exists there (free space, another object) which is not the target object.

3.5.3 Bayesian Fusion in Log-Odds

We treat each frame as an independent noisy observation of the target’s location and accumulate evidence additively in log-odds. Recall that q is the natural-language user query (e.g., “*find me the apple*”) and that $p_o^{(t)} \in [0, 1]$ is the per-frame VLM probability that object o contains an instance of q at time t . This probability is clipped to a bounded range and converted to a log-odds delta

$$\tilde{p}_o^{(t)} = \text{clip}(p_o^{(t)}, \epsilon, 1 - \epsilon), \quad \Delta\ell_o^{(t)} = \log \frac{\tilde{p}_o^{(t)}}{1 - \tilde{p}_o^{(t)}}, \quad \epsilon = 10^{-3},$$

where the clipping ensures bounded log-odds. The cumulative per-object log-odds is updated recursively as

$$\ell_{\text{sem}}^{(t)}(o) = \ell_{\text{sem}}^{(t-1)}(o) + \mathbf{1}[o \text{ observed at } t] \cdot \Delta\ell_o^{(t)},$$

where $\mathbf{1}[\cdot]$ is the indicator function. Objects unobserved at frame t retain their previous log-odds, so the belief persists through occlusions and is updated only when fresh evidence arrives. The initial value $\ell_{\text{sem}}^{(0)}(o) = 0$ corresponds to an uninformative prior $\sigma(0) = 0.5$, where σ is the logistic sigmoid. This is the canonical additive log-odds form of recursive Bayesian filtering [16]; equivalent in probability space to multiplying independent likelihoods and renormalizing, which is what we want when each frame’s observation is conditionally independent given the latent location of the target.

A useful property of this formulation is that consensus across frames compounds. Two views of the same cabinet, each scoring it with $p_o^{(t)} = 0.7$, drive its log-odds to $2\log(7/3) \approx 1.69$, or $\sigma \approx 0.85$; three views drive it to $\sigma \approx 0.92$; and so on. Conversely, a single high score from one view contradicted by a low score from another partially cancels: $\log(7/3) + \log(3/7) = 0$, so two opposing observations return the belief to its prior.

3.5.4 Combined Voxel Field

The semantic and geometric layers are combined at the voxel level. The per-object semantic log-odds $\ell_{\text{sem}}^{(t)}(o)$ are first painted onto the voxel grid: every voxel v inside an object’s bounding box $\mathcal{B}(o)$ inherits that object’s log-odds. Voxels claimed by more than one object are resolved by taking the per-voxel maximum, so a cup placed inside an open cabinet does not contribute its belief twice once it has been assigned to its own voxels:

$$\ell_{\text{sem}}^{(t)}(v) = \begin{cases} \max_{o \in \mathcal{O}_t : v \in \mathcal{B}(o)} \ell_{\text{sem}}^{(t)}(o) & \text{if } \exists o \in \mathcal{O}_t : v \in \mathcal{B}(o), \\ 0 & \text{otherwise.} \end{cases}$$

The combined per-voxel log-odds is then the sum of the geometric and semantic layers,

$$\ell_t(v) = \ell_{\text{geom}}(v) + \ell_{\text{sem}}^{(t)}(v),$$

yielding a complete log-odds field over the occupancy map. The two layers add because they encode independent sources of evidence. The field is recomputed at every frame and used downstream for action selection (§3.6).

3.6 Action Selection

At each step, the planner uses the combined voxel field $\ell_t(v)$ from §3.5.4 to pick the object the robot should investigate next, and then asks a VLM to choose which skill to apply to that object.

3.6.1 Picking an Object to Interact with

For each object $o \in \mathcal{O}_t$, the planner aggregates the combined log-odds over the voxels inside its bounding box,

$$L_t(o) = \sum_{v \in \mathcal{B}(o)} \ell_t(v).$$

The aggregate $L_t(o)$ summarizes how much total belief mass sits inside o 's extent. The target object is then selected as the argmax,

$$o^* = \arg \max_{o \in \mathcal{O}_t} L_t(o).$$

3.6.2 Selecting a Skill via the VLM

Once o^* is chosen, the planner picks the best stored RGB observation of o^* from its scene-graph entry, draws a bounding box around the object in the image, and labels the box with the object's class and current belief value. The annotated image, the user query q , and the object's class name are sent to a vision-language model in a single call. The VLM returns one action from the search-action vocabulary,

$$a^* \in \{\text{look_at}, \text{open_drawer}, \text{open_shelf}, \text{open_cabinet}, \text{pick_object}\},$$

with a one-sentence justification. `look_at` is used when the current view is too distant or oblique to commit to an interaction, `open_*` when the highlighted object is the relevant kind of container, and `pick_object` when the target itself is directly visible at this location. The VLM is given no global plan, no action history, and no explicit cost; its job is to look at the highlighted object and decide which of the five skills is appropriate.

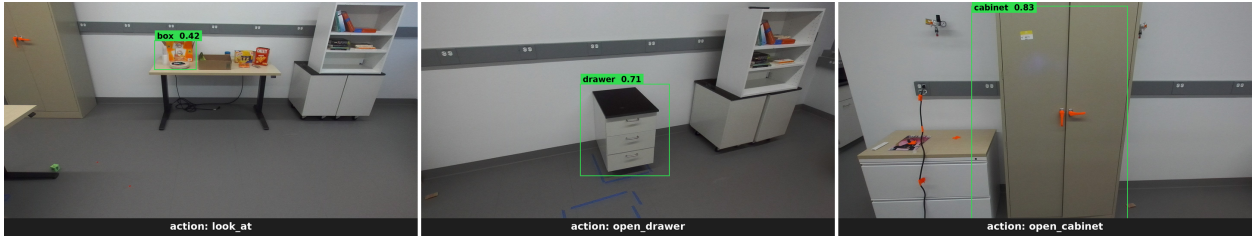


Figure 3.4: Examples of VLM action inference. Each panel shows an annotated image of the highest-belief object (green bounding box, class label, current belief value) alongside the action the VLM selected. From left to right: an open box seen from a distance triggers `look_at`; a closed drawer triggers `open_drawer`; a closed cabinet triggers `open_cabinet`.

The selected skill is then dispatched to the skill library described in §3.7. Once execution finishes, the loop returns to the perception pipeline and the cycle restarts.

3.7 Skill Library

The action selection rule above selects a single skill from a fixed library, grouped into three categories: perception, which brings an occluded space into view; interaction, which manipulates an object to reveal an occluded space; and retrieval, which is called on termination to retrieve the target object. The specific implementations of these skills are described in Chapter 4.

Category	Skill	Description
Perception	look_at	Moves the base and orients the robot's head to bring a cluster of voxels into view.
Interaction	open_cabinet	Opens both sides of a revolute cabinet.
	open_drawer	Opens a prismatic drawer.
Retrieval	pick_from_table	Picks an object resting on an open tabletop.
	pick_from_shelf	Picks an object resting inside an open shelf.
	pick_from_drawer	Picks an object resting inside an open drawer.

Table 3.1: Primitives in the skill library, grouped by category.

Chapter 4

System Implementation

We deployed the system on the Rainbow Robotics RBY1 humanoid, and rolled out the full pipeline in an office-like environment.

4.1 Robot Platform: RBY1

The RBY1 is a 24-DOF humanoid, with two wheels, six torso joints, seven joints per arms, two head joints, and two grippers. We additionally equipped it with a head-mounted ZED2 stereo RGB-D camera, as well as a ZED X One fisheye RGB-D camera on each wrist. A Livox solid-state lidar mounted at the center of rotation on the base provides high-frequency odometry.

The compute is split across two machines that communicate over the network. The jetson computer on the robot runs low-level controllers on the robot and sends RGB-D observations to a remote GPU workstation, which updates the maps and sends back low-level robot actions.

4.2 Capture Observation Loop

In addition to observations captured by the `look_at` skill, observations are also gathered by a background process that runs alongside the main control loop. The loop polls the base and head poses at 2 Hz and triggers an image capture when any of three thresholds have been exceeded: 0.15 m of base translation, 15° of base rotation, or 10° of head motion. This intermittent capture builds up the scene incrementally as the robot moves through the workspace.

4.3 Classical Navigation

The `navigate` skill drives the robot base from its current pose to a target 2D pose using collision-free path planning over the occupancy map of §3.4. The planner reduces the 3D occupancy grid to two 2D grids by height band: a base-height grid (15–30 cm above the floor) that captures obstacles in the wheel envelope, and a torso-height grid (30–200 cm)

that captures obstacles in the torso swept volume. Each 2D grid is a binary OCC/FREE mask obtained by collapsing the corresponding height slice of the 3D grid.

Each 2D grid is then *inflated* by the corresponding collision radius of the robot— 20 cm for the base, 35 cm for the torso— to ensure the planned path keeps the robot’s footprint clear of obstacles. The two inflated grids are merged by union; A* is run on the merged grid to find the shortest cell-by-cell path from start to goal; the resulting path is smoothed by line-of-sight checks against the inflated grid to remove unnecessary waypoints. The smoothed waypoints are dispatched to the base controller, which rotates in place to face each waypoint and drives forward until within a fixed tolerance of the goal.

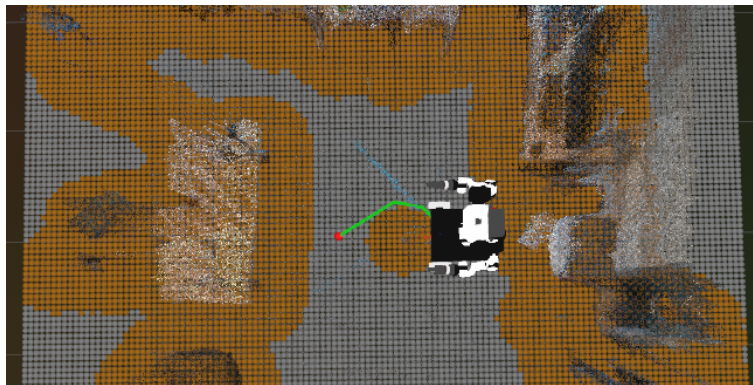


Figure 4.1: Path planning over the inflated occupancy grid. Orange voxels are inflated obstacles, gray is free space, and the green line is the smoothed A* path between the robot’s start pose and the target.

4.4 Engineered Skills

The remaining skills in the library are implemented as parameterized engineered controllers. Each takes a point cloud of an object and produces a fixed motion sequence. Photos of the robot executing each skill are collected in Figure 4.2.

Base Placement

Each skill requires the robot to stand in a specific spatial relationship to its target before execution. Prior to running the each skill, we therefore run a base-placement procedure that sends the base to an SE(2) pose from which the skill is executable. Most skills compute a single deterministic base pose with respect to the manipulated object. `pick_from_table` samples 1000 candidate base poses within two radius bands, and filters against a collision constraint between the RBY1’s collision spheres and the obstacle field.

4.4.1 look_at

`look_at` jointly plans a base SE(2) pose and head pan/tilt angles that place the target inside the head camera’s field of view with no occluders along the line of sight. Visibility is checked

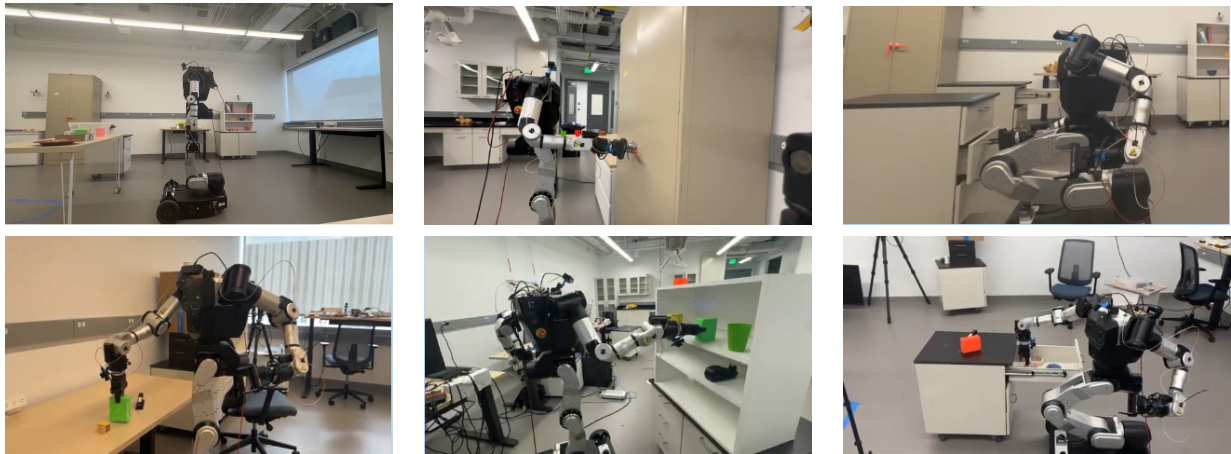


Figure 4.2: The robot executing each of the six engineered skills. Top row, left to right: `look_at` on a cluttered table, `open_cabinet`, `open_drawer`. Bottom row, left to right: `pick_from_table`, `pick_from_shelf`, `pick_from_drawer`.

by projecting the target into the synthetic image plane at the candidate pose and walking a DDA ray through the occupancy grid, treating only OCCUPIED voxels as opaque.

4.4.2 `open_cabinet`

`open_cabinet` opens both doors of a two-door cabinet in sequence, right then left, backing the base off 0.4m between them. For each door, Gemini is queried for the handle pixel in the head image, a plane is fit to the door front to obtain the surface normal, the base is navigated to the standoff pose, and a pre-recorded arm trajectory is replayed to swing the door through its hinge arc.

4.4.3 `open_drawer`

`open_drawer` queries Gemini for the drawer handle pixel in the head image and back-projects it via depth into the base frame. The arm then executes a four-waypoint fingertip trajectory: preapproach 23 cm behind the handle, grasp at the handle, pull 40 cm back to fully open, and retreat a further 5 cm to clear the gripper.

4.4.4 `pick_from_table`

`pick_from_table` moves the arm to the TiPToP pick-home configuration and dispatches the pick to TiPToP (§4.5).

4.4.5 `pick_from_shelf`

`pick_from_shelf` moves the arm to the TiPToP shelf-home configuration and dispatches the pick to TiPToP (§4.5).

4.4.6 pick_from_drawer

`pick_from_drawer` replays the a pre-recorded arm trajectory to position the gripper above the drawer interior, then dispatches the pick to TiPToP (§4.5).

4.5 TiPToP Adaptation

TiPToP [17] is an open-vocabulary task-and-motion-planning system for grasping novel objects on cluttered tabletops. Given an RGB-D observation and a natural-language instruction (e.g., *"pick up the apple"*), it segments the scene with an open-vocabulary detector, generates candidate grasps on the target object, and runs a sampling-based motion planner to produce a collision-free arm trajectory that executes the grasp. The output is a sequence of joint waypoints that the robot replays to perform the pick.

TiPToP was originally targeted at tabletop pick-and-place on a robot arm. We use it to plan only over the right arm and torso joint 5 (the topmost torso joint), treating the rest of the body as a static obstacle. Adapting this system to the search-and-retrieve setting on the RBY1 humanoid required three substantive changes.

Pick-only and place-only goals via predicates. The original TiPToP expressed the task as a single pick-and-place. Our action selector treats pick and place as separate skills, so we extended the goal-state vocabulary so that TiPToP can be invoked for either alone. Goal states are built from grounded atoms: `Holding(obj)` expresses a pure-pick goal (the robot ends the episode holding the object), and `On(obj, surface)` expresses a placement. `HandEmpty` is added to the goal only when no `Holding` atom is present. At invocation time, TiPToP checks the gripper width and, if the gripper is already closed, infers from the goal which object is being held and seeds the initial TAMP state with that holding atom; this lets a place-only call run from a state in which the robot is already mid-grasp from a previous skill execution.

Dynamic start pose. The original TiPToP implementation assumed a fixed home configuration at the start of every plan. In our setting the robot must reach into shelves and drawers at varying heights, which requires the torso to crouch to different levels before the planner runs. We therefore added a sensing step that runs before the planner is warmed up: the client reads the live robot configuration and locks all non-active joints (torso joints 0–4, left arm, wheels, head) to their sensed values for the duration of planning.

Container sensing and meshing. The original TiPToP modeled the workspace as a flat tabletop and treated every object above the table surface as a movable object, mesh-completing each one via its convex hull. This breaks for containers: a target inside an open drawer or box would be hidden inside the convex hull of the container itself, making it inaccessible to the planner. We therefore added a branch that treats containers as a special case, voxelizing them and meshing the resulting voxel grid directly rather than collapsing them to a convex hull. An object is classified as a container if its detected class label is one

of `drawer`, `cabinet`, or `shelf`, or if its bounding box strictly surrounds another detected object.

4.6 Learned Skill Policies

We plan to train policies for the `open_drawer` and `open_cabinet` skills using Action Chunking with Transformers (ACT) [18]. ACT is a behavior-cloning architecture that predicts a chunk of future actions at each step given the current RGB observation and joint values.

We plan to collect ~ 200 teleoperated demonstrations of each skill. At deployment time, the action selector chooses a skill name as before, and the corresponding learned policy is rolled out instead of the engineered controller. The input/output interface to the rest of the system is unchanged.

Chapter 5

Results

This chapter presents a preliminary deployment study of the system on a Rainbow Robotics RBY1 humanoid in a real office-like environment, followed by a planned evaluation protocol and a discussion of next steps.

5.1 Preliminary Deployment Study

We deploy the system end-to-end in a fixed office-like scene and run it on four search episodes spanning two queries and two placement conditions. The aim of this preliminary study is to verify that the system runs end-to-end on real hardware and that the belief update produces the intended behavior — updating in response to fresh evidence, redirecting the robot toward more plausible candidates as the search progresses, and reaching retrieval in a variety of placements of the target.

Task

The scene contains a chest of two closed drawers with office supplies (markers, scissors, measuring tape) visibly stacked on top, an open shelf with textbooks, and a table cluttered with snack items including a cardboard box with bananas visible inside. The scene is staged so that each candidate location carries clear visual context cues from the robot’s starting viewpoint: the supply objects on top of the drawers establish them as office-supply storage, and the snack items establish the table as food storage.

We use two natural-language queries, “*find the apple*” and “*find the stapler*”, each run under two placement conditions. In the *predictable* condition, the target sits in the location its semantic context predicts: the apple in the cardboard box on the snack table, the stapler in one of the office-supply drawers. In the *unpredictable* condition the placements are swapped: the apple is in a drawer and the stapler is in the cardboard box on the snack table. The robot’s starting pose is held constant across all four episodes.

Results

Table 5.1 reports the number of dispatched skills per episode. All four episodes reached retrieval. In the predictable condition, the system reached the target in a single action



Figure 5.1: The deployment scene from the robot’s starting pose, with the two candidate target locations annotated in red (the cardboard box on the snack table) and blue (the office-supply drawers).

for both queries: the highest-belief object on the initial frame matched the actual target location, and a single `look_at` or `open_drawer` action was sufficient to confirm the target. In the unpredictable condition, the system reached the target in 3 actions for the apple and 4 for the stapler. The additional actions correspond to the system ruling out higher-scoring candidates one by one as fresh observations contradicted the initial belief, eventually shifting belief mass onto the actual target location.

Query	Placement	Actions to retrieval
Apple	Predictable	1
Apple	Unpredictable	3
Stapler	Predictable	1
Stapler	Unpredictable	4

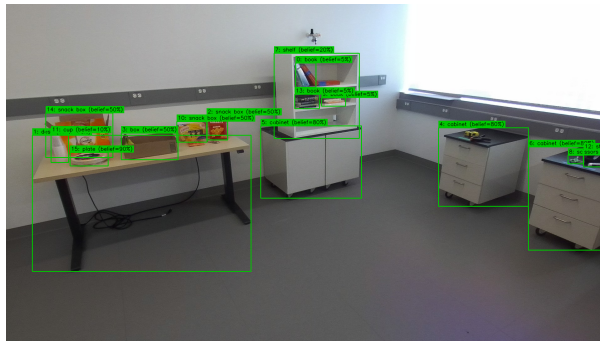
Table 5.1: Number of dispatched skills (excluding the terminal retrieval action) per episode in the preliminary deployment study.

An example walkthrough

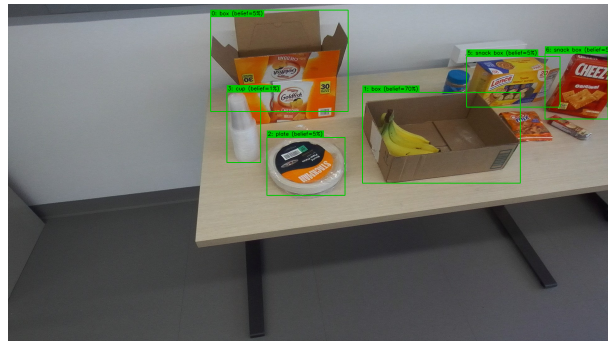
Figure 5.2 shows the belief-annotated observations from the apple search under the unpredictable placement condition, in which the apple is hidden in one of the office-supply drawers. This episode is informative because the initial semantic prior strongly favored the wrong candidates, and the resulting trajectory exercises the belief-update behavior the system was designed to produce.

On the initial frame (Fig. 5.2a), the VLM scored the plate on the snack table at 0.9 and the two office-supply drawers at 0.8. The highest-belief object was the plate, so the action selector chose it and dispatched a `look_at` skill to bring it into closer view. In the second frame (Fig. 5.2b), the closer viewpoint showed the plate had no apple on it; the VLM accordingly scored the plate at 0.05 on this frame, and the additive log-odds update drove the plate’s cumulative belief sharply downward. The highest-belief object then became one of the two drawers, which the system opened via `open_drawer`. The drawer was revealed to

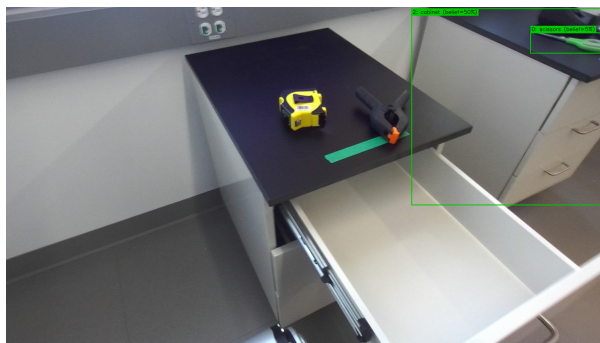
be empty (Fig. 5.2c), so the belief then shifted to the second drawer. Opening the second drawer revealed the apple (Fig. 5.2d), and the episode terminated.



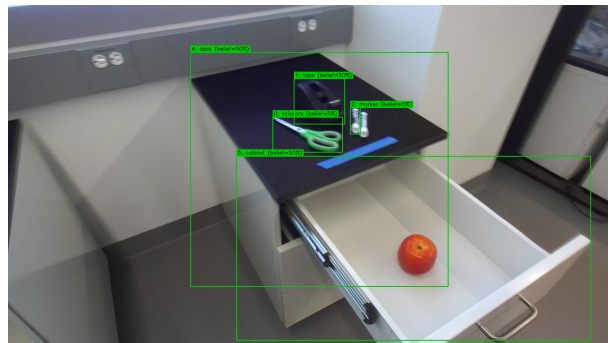
(a) Frame 00 – initial observation. Plate scored 0.9; drawers scored 0.8.



(b) Frame 01 – after `look_at` on the plate. Plate now scored 0.05.



(c) Frame 02 – after `open_drawer` on the first drawer. Apple not inside.



(d) Frame 03 – after `open_drawer` on the second drawer. Apple found; episode terminates.

Figure 5.2: Belief-annotated observations from the apple search under the unpredictable placement condition. Contradicting evidence on each frame redirects belief mass to the next most plausible candidate.

5.2 Next Steps

Planned task scenarios

The deployment scene used above is small and tests two placements per query. A full evaluation will scale this up along three axes. First, a larger scene with more candidate locations—additional cabinets, drawers, and shelves—so that the search has more candidate regions to choose between. Second, more queries spanning different object categories, to reduce the dependence of the results on any single target. Third, multiple trials per (query, placement) cell with the target’s in-container pose perturbed between trials, to support statistical claims about success rate and efficiency.

We will report three metrics per episode. The first is *success rate*: the fraction of episodes in which the policy retrieves the target within a fixed time budget. Higher is better. The remaining two metrics are computed only over successful episodes, and for both, lower is better. *Time to retrieval* is the wall-clock time, in seconds, from the start of an episode to

the moment the target object is held in the gripper. *Number of actions to retrieval* is the count of skill executions (`look_at`, `open_cabinet`, `open_drawer`, `pick_from_*`) the policy issued until the target was in the gripper. The two efficiency metrics capture different facets of policy quality: action count reflects decision-making, while wall-clock time reflects total execution efficiency. A policy may score well on one and poorly on the other if, for example, it consistently picks correct but slow skills, so reporting both lets us distinguish these failure modes when they diverge.

Planned baselines

We will compare against two baselines, each replacing a single component of our pipeline with a representative alternative drawn from prior work. Both baselines will retain the robot platform, the perception stack, and the full skill library, so the comparison is strictly over how the belief and action selection components are realized, not over what the robot can physically do.

GenMOS baseline. This baseline replaces our two-layer belief with a geometry-only formulation following GenMOS [3]. We will implement an octree-based occupancy belief representation in place of our voxel grid, matching GenMOS’s hierarchical spatial representation, with the per-voxel log-odds prior derived from occupancy state (§3.5.2) attached to the leaves and aggregated up the tree. Action selection follows the same pattern as our system.

MoMa-LLM / CuriousBot baseline. This baseline removes the explicit belief representation and delegates action selection to an LLM, mirroring the architectural choice shared by MoMa-LLM [11] and CuriousBot [10]. Both methods build a scene graph from RGB-D observations and serialize it into a structured text prompt, from which an LLM picks the next high-level action over a fixed vocabulary. We will follow this design: the baseline builds the same semantic scene graph (§3.3) as our system, but instead of scoring objects with a per-frame VLM and accumulating log-odds, it serializes the current scene graph to text and asks an LLM to choose the next skill and target directly.

Chapter 6

Conclusion

6.1 Future Work

Cost-aware action selection. Assigning a cost to each robot action and incorporating this into the action selection would enable the robot to start with cheaper actions and naturally escalate into more expensive ones. To do so, we could define a heuristic for expected information gain of an action based on the belief state, and divide it by the cost of the action.

Multi-step lookahead. The current selector commits to the top-1 most-believed object at each step and does not reason about how the next several actions compose. Wrapping the per-step rule in a Monte Carlo Tree Search over short action sequences would enable the planner to make decisions that are temporally coherent, such as fully exploring one part of the room before moving to the opposite side instead of going back-and-forth.

Richer skill library. The interaction skills currently handle articulated containers (cabinets, drawers) and surface picks, but not movable occluders. Adding skills like `push_aside` or `pick_up_cloth` would let the system handle a richer class of scenes, such as ones where the target is behind a stack of boxes or underneath a piece of clothing.

Multi-room scope. The current problem formulation assumes a single room, and our implementation uses the lidar to estimate the bounds of the room to define the occupancy grid. Extending to multi-room workspaces requires us to relax this assumption and maintain an occupancy grid that could in theory extend infinitely in all directions.

6.2 Conclusion

The most consequential design choice in the system is that the belief is explicit, voxel-resolved, and built from two independent sources of evidence: a per-object semantic score from a vision-language model and a per-voxel geometric prior from the occupancy map. Both signals are bounded, calibrated probabilities; the rule for combining them is a simple sum in log-odds space; and the result is a 3D field that can be rendered, inspected, and debugged frame by frame.

Action selection on top of this substrate currently uses a VLM, and is therefore subject to the reproducibility caveats of any VLM-driven decision rule. But because the belief underneath is explicit, it is straightforward to swap the selector for something more accountable— a cost-aware curiosity rule, a tree-search policy, or a learned planner trained on logged episodes— without changing anything upstream. We see this separation between an explicit, inspectable belief and a swappable policy as the design’s durable property: regardless of how language and foundation models continue to evolve, having an interpretable intermediate representation between perception and decision makes the resulting system easier to reason about, easier to fix, and easier to extend.

Object search is a foundational skill that almost every useful robot task begins with. A robot that can fold laundry but cannot find the laundry, or one that can pour tea but cannot locate the kettle, has a limited domain of usefulness— bounded by whatever a human is willing to lay out for it in advance. The point of this thesis is that the gap between those constrained demos and a robot that can be turned loose in an unfamiliar room is, in large part, an object search problem. The system presented here is one step toward closing that gap. As the underlying components— vision-language models, occupancy mapping, learned manipulation skills— continue to improve, the framework around them should let those improvements compound rather than require redesign. The longer-term hope is that interactive object search becomes a default capability rather than a research problem, so that future robots can be asked to pack a suitcase, restock a supply room, or set a table without anyone having to first arrange the objects they need within reach.

References

- [1] A. Wandzel et al. “Multi-Object Search using Object-Oriented POMDPs.” In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 7194–7200.
- [2] K. Zheng et al. “Multi-Resolution POMDP Planning for Multi-Object Search in 3D.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021. URL: <https://arxiv.org/abs/2005.02878>.
- [3] K. Zheng, A. Paul, and S. Tellex. “Multi-Object Search in 3D: A System Approach.” In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2023. URL: <https://arxiv.org/abs/2303.03178>.
- [4] Y. Chen and H. Kurniawati. “POMDP Planning for Object Search in Partially Unknown Environment.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2023. URL: <https://arxiv.org/abs/2311.06430>.
- [5] L. L. S. Wong, L. P. Kaelbling, and T. Lozano-Pérez. “Manipulation-Based Active Search for Occluded Objects.” In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2013.
- [6] D. S. Chaplot et al. “Object Goal Navigation using Goal-Oriented Semantic Exploration.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020. URL: <https://arxiv.org/abs/2007.00643>.
- [7] N. Yokoyama, D. Batra, and S. Ha. “Vision-Language Frontier Maps for Zero-Shot Semantic Navigation.” In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2024. URL: <https://arxiv.org/abs/2312.03275>.
- [8] M. F. Gupta et al. “SEEK: Semantic Reasoning for Object Goal Navigation in Real World Inspection Tasks.” In: *Robotics: Science and Systems (RSS)*. 2024. URL: <https://arxiv.org/abs/2405.09822>.
- [9] H. Jiang, B. Huang, et al. “RoboEXP: Action-Conditioned Scene Graph via Interactive Exploration for Robotic Manipulation.” In: *Conference on Robot Learning (CoRL)*. 2024. URL: <https://arxiv.org/abs/2402.15487>.
- [10] Y. Wang, L. Fermoselle, et al. “CuriousBot: Interactive Mobile Exploration via Actionable 3D Relational Object Graph.” In: *arXiv preprint arXiv:2501.13338* (2025). URL: <https://arxiv.org/abs/2501.13338>.
- [11] D. Honerkamp et al. “Language-Grounded Dynamic Scene Graphs for Interactive Object Search with Mobile Manipulation.” In: *IEEE Robotics and Automation Letters (RA-L)* (2024). URL: <https://arxiv.org/abs/2403.08605>.

- [12] S. Liu et al. “Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection.” In: *European Conference on Computer Vision (ECCV)*. 2024. URL: <https://arxiv.org/abs/2303.05499>.
- [13] T. Cheng et al. “YOLO-World: Real-Time Open-Vocabulary Object Detection.” In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024. URL: <https://arxiv.org/abs/2401.17270>.
- [14] A. Kirillov et al. “Segment Anything.” In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023. URL: <https://arxiv.org/abs/2304.02643>.
- [15] A. Radford et al. “Learning Transferable Visual Models from Natural Language Supervision.” In: *International Conference on Machine Learning (ICML)*. 2021. URL: <https://arxiv.org/abs/2103.00020>.
- [16] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.
- [17] W. Shen, N. Kumar, et al. “TipTop: ...” In: *arXiv preprint arXiv:2603.09971* (2025). URL: <https://arxiv.org/pdf/2603.09971v1>.
- [18] T. Z. Zhao et al. “Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware.” In: *Robotics: Science and Systems (RSS)*. 2023. URL: <https://arxiv.org/abs/2304.13705>.