Abstract of ”Generalized Object Search”
by Kaiyu Zheng, Ph.D., Brown University, February, 2023

Future collaborative robots must be capable of finding objects. As such a fundamental skill, we expect object search to eventually become an off-the-shelf capability for any robot, similar to *e.g.*, object detection, SLAM, and motion planning. However, existing approaches either make unrealistic compromises (*e.g.*, reduce the problem from 3D to 2D), resort to ad-hoc, greedy search strategies, or attempt to learn end-to-end policies in simulation that are yet to generalize across real robots and environments. This thesis argues that through using Partially Observable Markov Decision Processes (POMDPs) to model object search while exploiting structures in the human world (*e.g.*, octrees, correlations) and in human-robot interaction (*e.g.*, spatial language), a practical and effective system for generalized object search can be achieved. In support of this argument, I develop methods and systems for (multi-)object search in 3D environments under uncertainty due to limited field of view, occlusion, noisy, unreliable detectors, spatial correlations between objects, and possibly ambiguous spatial language (*e.g.*, "The red car is *behind* Chase Bank"). Besides evaluation in simulators such as PyGame, AirSim, and AI2-THOR, I design and implement a robot-independent, environment-agnostic system for generalized object search in 3D and deploy it on the Boston Dynamics Spot robot, the Kinova MOVO robot, and the Universal Robots UR5e robotic arm, to perform object search in different environments. The system enables, for example, a Spot robot to find a toy cat hidden underneath a couch in a kitchen area in under one minute. This thesis also broadly surveys the object search literature, proposing taxonomies in object search problem settings, methods and systems.

Generalized Object Search

Kaiyu Zheng

A dissertation submitted in partial fulfillment of the
requirements of the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island
February 2023

This dissertation by Kaiyu Zheng is accepted in its present form
by the Department of Computer Science as satisfying the
dissertation requirement of the degree of Doctor of Philosophy.

Date _____ _____
Stefanie Tellex, Advisor

Recommended to the Graduate Council

Date _____ _____
George D. Konidaris, Reader

Date _____ _____
Michael L. Littman, Reader

Date _____ _____
Ellie Pavlick, Reader

Date _____ _____
Leslie P. Kaelbling, Reader

Approved by the Graduate Council

Date _____ _____
Thomas A. Lewis, Dean of the Graduate School

Abstract of "Generalized Object Search"
by Kaiyu Zheng, Ph.D., Brown University, February 2023.

Future collaborative robots must be capable of finding objects. As such a fundamental skill, we expect object search to eventually become an off-the-shelf capability for any robot, similar to *e.g.*, object detection, SLAM, and motion planning. However, existing approaches either make unrealistic compromises (*e.g.*, reduce the problem from 3D to 2D), resort to ad-hoc, greedy search strategies, or attempt to learn end-to-end policies in simulation that are yet to generalize across real robots and environments. This thesis argues that through using Partially Observable Markov Decision Processes (POMDPs) to model object search while exploiting structures in the human world (*e.g.*, octrees, correlations) and in human-robot interaction (*e.g.*, spatial language), a practical and effective system for generalized object search can be achieved. In support of this argument, I develop methods and systems for (multi-)object search in 3D environments under uncertainty due to limited field of view, occlusion, noisy, unreliable detectors, spatial correlations between objects, and possibly ambiguous spatial language (*e.g.*, "The red car is *behind* Chase Bank"). Besides evaluation in simulators such as PyGame, AirSim, and AI2-THOR, I design and implement a robot-independent, environment-agnostic system for generalized object search in 3D and deploy it on the Boston Dynamics Spot robot, the Kinova MOVO robot, and the Universal Robots UR5e robotic arm, to perform object search in different environments. The system enables, for example, a Spot robot to find a toy cat hidden underneath a couch in a kitchen area in under one minute. This thesis also broadly surveys the object search literature, proposing taxonomies in object search problem settings, methods and systems.

Curriculum Vitae

Kaiyu Zheng was raised in Guangzhou, China. He graduated from the Guangzhou No. 2 High School. Afterwards, he received a B.S. and M.S. with honors in computer science and a minor in mathematics from the University of Washington, Seattle. He then pursued a Ph.D. in computer science at Brown University and defended his thesis in December, 2022. His research interests lie in making robots reliably act in the human world and interact with humans under uncertainty. He received the IROS RoboCup Best Paper Award in 2021 and was selected as a Robotics: Science and Systems (RSS) Pioneer in 2022.

Service activities during Ph.D.:

Organizer, Brown Robotics Group Seminar Series, 2021
Head Teaching Assistant, Learning & Sequential Decision Making, 2021
Presenter, Staff Development Day, Brown University, 2021
Peer mentor, PhD Mentorship Program, Brown CS, 2020 - 2022
Moderator, PhD Alumni Panel, Brown CS, 2020
Peer mentor, International Graduate Student Orientation, 2019, 2021
Organizer, Hiking Group, 2019
Representative, Graduate Student Council, Brown CS, 2019

Acknowlegements

My journey in research could not have been more fortunate because of all the people who I have met and who have given me a hand along the way. First and foremost, I want to thank my amazing advisor, Stefanie Tellex, for her advice, mentorship, understanding, and unwavering support. I would always appreciate her direct and sharp advice, "do good research," "don't spread yourself too thin," which have been invaluable to me.

I would like to thank my committee, George Konidaris, Michael Littman, Ellie Pavlick and Leslie Kaelbling for your feedback and advice, and coming to my defense in person. In particular, I want to thank George for his insightful perspectives, Michael for his guidance on theoretical research, Ellie for how to lead discussions, and Leslie for her critique.

I want to thank a few key people who shaped my research journey. First, I thank Andrzej Pronobis, my undergraduate advisor, for taking me on, teaching me details, and guiding me in research. Then, I want to thank Yoonchang Sung, my mentor, from whom I learned the critical attitude towards research. Finally, I want to thank Bo Wang, one of my best friends, for introducing me to research in our freshman year. Otherwise, I wouldn't even know.

I want to thank all my collaborators, especially Rohan Chitnis, Yoonchang Sung, Deniz Bayazit, and Rebecca Matthew for the project times we shared. I enjoyed working as a mentor with many students, including Semanti Basu, Sreshtaa Rajesh, Monica Roy, Anirudha Paul, Shangqun Yu, Eliza Sun, and Vedant Gupta. Moreover, my peers in and out of the H2R and IRL labs and the department: Eric Rosen, Sam Lobel, Thao Nguyen, Ifrah Idrees, Max Merlin, Ben Abbatematteo, Matthew Corsaro, Seungchan Kim, Johnathan Chang, Nick DeMarinis, Akhil Bagaria, Albert Webson, Josh Roy, Nishanth Kumar, Tuluhan Akbulut, Nihal Nayak, Saket Tiwari, Shane Parr, Cam Allen, Skye Thompson, Haotian Fu, Anita de Mello Koch, Lilika Markatou, Denizalp Goktas, Lucy Qin, Aaron Traylor, Franco Solleza, Ghulam Murtaza, Alessio Mazzetto, Zheng-Xin Yong, Kenny Jones, Mikhail Okunev, Rao Fu, Ji Won Chung, Aditya Ganeshan, Benjamin Spiegel, Benedict Quartey, Calvin Luo, Marilyn George, Daniel Engel, Roma Patel, Charles Lovering, Jack Merullo, Tian Yun, Ziyi Yang, Jessica Forde, Talie Massachi, Brandon Woodard, Peilin Yu, David Tao, Lishuo Pan, Rahul Sajnani, Ruochen Zhang, Catherine Chen, Vivienne Chi, Wasiwasi Mgonzo, Amina Abdullahi, George Zerveas, Yingjie Xue, Yanqi Liu, Selena Ling, Ewina Pun, Kai Wang, Qian Zhang, Yanyan Ren, Fumeng Yang, Anna Wei – the list is inenumerable – have been my community and friends that enriched this journey with moments and discussions. Thank you. In addition, I was fortunate to interact with and learn from my respectful peers across different research fields: Andreea Bobu, Weiyu Liu, Shaoxiong Wang, Yuchen Xiao, Mohit Sridhar, Anirudha Vemula, Serena Booth, Yilun Zhou, Rachel Holladay, Xiaolin Fang, Tom Silver, Jiayuan Mao, Aidan Curtis, Caris Moses, Dhruv Shah, Ted Xiao, Shivam Vats, Angela Chen, Linfeng Zhao, Lena Downes, Kevin Doherty, and the list goes on and on.

I also want to thank many senior folks who have given me memorable advice and feedback, whom I also respect: Tomás Lozano-Pérez, Julie Shah, Dieter Fox, Vikash Kumar, R. Iris Bahar, Vikram Saraph, David Whitney, David Abel, Lawson Wong, David Paulius, Ankit Shah, Lucas Lehnert, Kavosh Asadi, Yu Xiang, Arunkumar Byravan, James Tompkin, Amy Greenwald, Paul Beame, Rajesh P. N. Rao, Ugur Cetintemel, just to name a few. I especially want to thank Matthew Gombolay for his insightful comments following my talk at Georgia Tech, which influenced my later presentations. The same goes to Nakul Gopalan, and others in the audience.

I want to thank the wonderful people I worked with at Viam, who helped me get the UR5e arm demo done at the closure of my PhD: Fahmina Ahmed, Matt Dannenberg, Bijan Haney, Nicholas Franczak, Gautham Varadarajan, Khari Jarrett, and Raymond Bjorkman.

I want to thank the amazing staff at Brown CS, including Suzanne Alden, Lauren Clarke, Kathy Kirman, John Tracey-Ursprung, Eugenia DeGouveia, and others, as well as Ronni Edmonds who manages Brown Auxiliary Housing. You made my life easier.

I want to thank my *friends* at Brown, in and out of the lab; from UW and around Seattle; from high school, middle school; from internships, elsewhere. You make up a big part of my life. I wish you all the best.

I also want to thank all my art teachers. I enjoyed art classes the most growing up.

Finally, I want to sincerely thank all my parents. All beyond paper. — Kaiyu Zheng

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

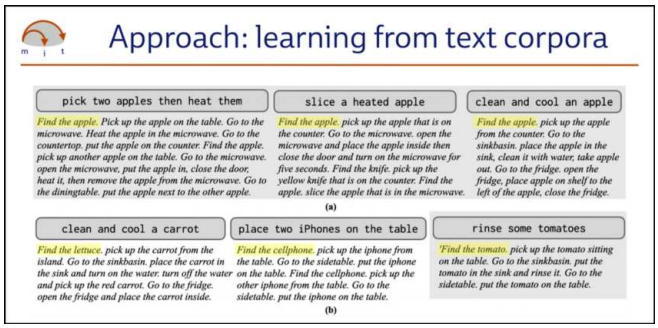
Introduction

1.1 Significance and Challenges of Object Search

SEARCHING for objects has been a fundamental skill for people since ages ago. From fruits to glasses, water sources to ore mines, the ability to search has enabled people's daily lives and provided much value to society. However, searching is unpleasant, to say the least, and can be difficult or prohibitive depending on the situation. Imagine an elderly person searching for their missing pair of glasses. Imagine a factory worker going back and searching for their tools. Imagine a wildfire that devastates a neighborhood, and a search and rescue team has to risk their lives in search of survivors. Now, imagine a robot that can help. Autonomy in object search is therefore intrinsically valuable.

Besides its intrinsic value, object search is also a prerequisite for basically any task involving the objects being searched for (*a.k.a.* the "target objects"). For intelligent, collaborative robots of the future, this means that a robot that can cook should better be able to search for the ingredients and the pan before cooking, and a robot that can clean should be able to find the cleaning tools. This point is well reflected in Figure 1.1a that shows six examples of a large language model (LLM) taking in a high-level task description (*e.g.* *"pick two apples then heat them"*) and outputting a sequence of sentences each corresponding to a lower-level action (P. Sharma et al., 2022). In all examples, the output begins with *"Find ..."* (*e.g.* *"Find the apple"*, *"Find the lettuce"*, etc.) as the first action. The SayCan system developed by Google (M. Ahn et al., 2022) with a similar capability exhibits the same behavior (Figure 1.1b). Object search is literally the prerequisite skill.

While significant, object search is also challenging. The challenges of object search are best illustrated through a practical scenario. Before diving into that, let us for a moment



(a) Examples from (P. Sharma et al., 2022) showing that an LLM breaks down given high-level descriptions into sequences of low-level actions. In all examples, the output begins with *"Find ..."*. Source: slide from Jacob Andreas's presentation at RLDM 2022. Courtesy of Jacob Andreas.

(b) Example trial of the SayCan system that outputs a sequence of low-level skills feasible for a robot given a high-level task (cleaning spilled drink). Again, the output begins with *"Find ..."*. Source: (M. Ahn et al., 2022).

Figure 1.1: The significance of object search is evident in part because it is a prerequisite for basically any task involving the target objects. Two supporting examples are provided above. In all cases, object search (*i.e.*. *"Find ..."*) comes first.

examine the phenomenon where *"Find ..."* is an atomic, low-level action. What happens in Figure 1.1a is in part due to the use of the ALFRED dataset (Shridhar et al., 2020) to train the LLM, where Amazon Mechanical Turkers provide step-by-step instructions (low-level actions) for a video demonstration of a high-level task. However, rather than breaking down the steps for object search into concrete commands such as *"Go to ..."* or *"Turn left"*, the Turkers summarize those steps as *"Find ..."*, which is vague for execution. Interestingly, SayCan (Figure 1.1b) exhibits the same phenomenon, using PaLM (Chowdhery et al., 2022), a 540-billion parameter LLM trained with a huge dataset of natural language in diverse contexts (*e.g.* books, webpages, etc.). Albeit speculative, one explanation is that carrying out *"Find ..."* would likely entail other sub-tasks such as navigation (*e.g.* maneuvering the robot's viewpoint), manipulation (*e.g.* opening a container), or another search task, recursively (*e.g.* to find an apple, one sub-task could be to first find the fridge), which is hard to elaborate when providing a single-step instruction. This duality of object search, being both an atomic skill in people's mind (and in LLM) as well as an abstract-level skill in reality, makes it a unique problem to study in robotics.

Returning to the discussion of challenges, Figure 1.2 shows a typical real-world scenario of object search. A Boston Dynamics Spot robot in our lab is tasked to find the book which is located in front of a monitor as quickly as possible. In this version of Spot, the most agile viewport, and the only colored one, is the camera on the mounted arm's gripper. The robot's objective is to perform search by executing a sequence of view pose changes to look around in the lab and eventually automatically declare the book to be found at some location. The robot's performance is evaluated based on success and efficiency. Several challenges emerge as we consider building a real-world system effective for this task:

- **Partial observability.** Although, as a third-person audience, we immediately see

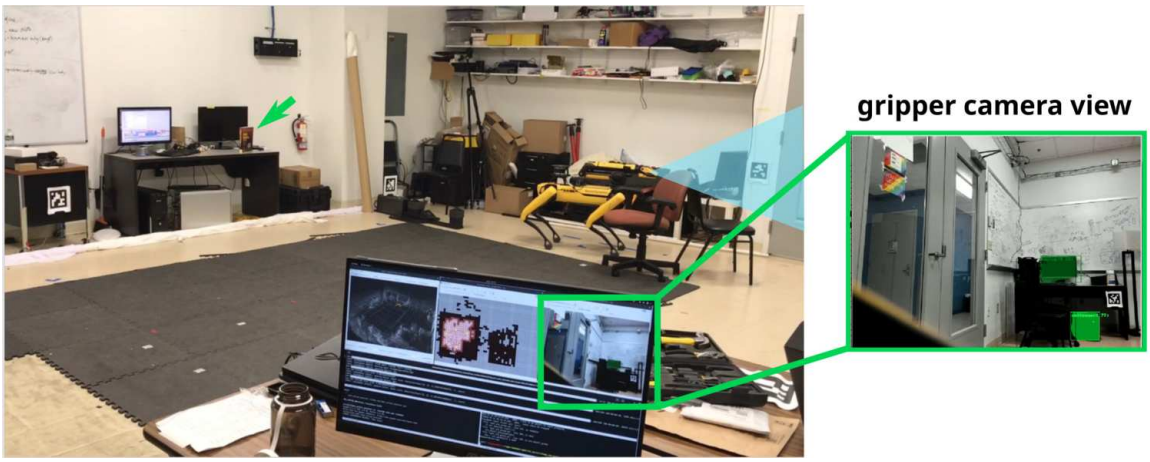


Figure 1.2: An example real-world object search scenario. The Spot robot (yellow legged robot) is tasked to find the book in front of a monitor (green arrow). The robot searches by navigating and controlling its 6-DOF gripper camera to look around within the room; the camera has a limited field of view subject to occlusion and the robot's perception pipeline is prone to uncertainty and errors.

where the book is, the robot has no knowledge of that and it has a limited field of view through its gripper camera subject to occlusion. The robot must reason about where to look under such partial observability.

- **Perceptual uncertainty.** Unlike the human eye, which can accurately distinguish most recognizable objects within the field of view when close enough, the robot's perception system is subject to noise, uncertainty, and errors in object detection and segmentation, even using state-of-the-art computer vision models. False negatives and false positives are inevitable for most real-world robotic systems since they often receive images not in line with the training distribution of those models. Such perceptual uncertainty is not limited to camera-based object detectors, and it is necessary to be considered by the robot for effective search using its on-board sensors.

- **Complex, unstructured environment.** The environment that the robot operates in is complex and unstructured, meaning that it does not just contain the robot itself and the target object. It also contains many other objects arranged in an unstructured fashion for the robot.

- **Human input (*e.g.* language hints).** Since the robot performs the search in a human environment or alongside a human teammate, it would be beneficial for searching more efficiently if the robot can take advantage of human input, for example, through natural language hints (*e.g.*, *"the book is in front of the monitor"*).

- **Generalization (across environments, across robots).** We do not just want object search to work for this robot in this room for this book. We want the ability to search

> to generalize across different environments and robots. Ultimately, the ability to search should become an off-the-shelf capability that a robot can *just have*.

Addressing the above challenges with an emphasis on building principled, practical robotic systems is at the core of this thesis (Chapters 3 to 7). Our effort aims to enable most robots today with a movable viewport to perform object search in an off-the-shelf manner; we call this problem *generalized object search*. This is a feasible objective today, thanks to recent progress in mapping, navigation, motion planning, and object detection. It is also widely applicable and a first step towards realizing the value of autonomy in object search.

Note that, however, object search can become even more difficult when additional challenges are factored in, such as when a human teammate can engage in a dialogue with the robot, when target objects are dynamic or adversarial, when physical environment interactions such as container opening, decluttering, or tool use are necessary and within the robot's arsenal of abilities, or when the robot performs search while exploring an unknown region. In Chapter 2, I provide a broad review on object search, including methods designated to address these challenges. There is much left to do here for future work.

Most prior work in object search focuses on the kind of basic but essential setting in Figure 1.2. As discussed next, despite the large body of literature, previous work fails to address the above challenges in a realistic, practical and generalizable way. A more thorough review and taxonomy of previous work in object search is provided in Chapter 2.

1.2 Limitations of Previous Work

Object search can be broken down into two subproblems: (1) which region(s) to search in, and (2) how to search within a region.

For the first subproblem, there has been a long line of work (Wixson & Ballard, 1994; Kollar & Roy, 2009; Aydemir et al., 2013; W. Liu et al., 2021), where the focus has been on scalability and common sense: Can the robot reduce the uncertainty down to small search region(s), such as a room, within a large search environment, such as an entire floor? Can the robot make use of semantic knowledge (*e.g.* sponges are usually in kitchens) between objects and places to select the search region(s)? The pioneering work along this line by Wixson & Ballard (1994) made the following interesting remark:

> Once such a subregion has been selected, however, a harder problem arises, namely the problem of how to select views that search the subregion. This problem is not really addressed here.

This remark nicely leads to the second subproblem, which is considered harder. Indeed, object search in a 3D region by planning views under a time budget is shown to be NP-Complete (Y. Ye & Tsotsos, 1997). Consequently, previous work uses ad-hoc, greedy search strategies despite the problem being inherently sequential (Aydemir et al., 2011;

Zeng et al., 2020). Other work makes unrealistic compromises, for example, by simplifying the problem from 3D to 2D (Kollar & Roy, 2009; Wandzel et al., 2019; Bejjani et al., 2021), or by manually specifying the set of viewpoints (Xiao et al., 2019).

Additionally, work such as (Saidi et al., 2007) and (Tsuru et al., 2021) focuses on building real-world systems for object search but for specific humanoid robots. On the other hand, data-driven approaches for semantic visual navigation that map raw RGB images directly to navigation actions are popular today yet most works have only conducted evaluation in simulation (Yang et al., 2019; Chaplot et al., 2020; Mayo et al., 2021; Deitke et al., 2022; Schmalstieg et al., 2022). Generalization of such models across different environments and robots in the real-world poses a serious challenge.[1]

The focus of this thesis is on the problem of how to search within a region. Ultimately, that determines the success of object search and is arguably harder, involving the low-level challenges fundamental to an effective object search system. Compared to the data-driven approaches which aim for *model generalization* (*i.e.*, training a model that generalizes to all test scenarios), our work differs fundamentally in that we pursue *methodological generalization* (*i.e.*, proposing a method that is robot- and environment-independent).[2] We demonstrate generalization of our method by developing a system and package for generalized object search and integrate it with different robots in different environments.[3]

1.3 Contributions

The central argument of this thesis is:

> Through using Partially Observable Markov Decision Processes (POMDPs) to model object search while exploiting structures in the human world and in human-robot interaction, a practical and effective system for generalized object search can be achieved.

In support of this argument, we briefly summarize the primary contributions of this thesis as follows; Refer to Figure 1.3 for an overview of research projects covered in this thesis.

- We exploit the structure of octrees for 3D multi-object search and develop a scalable and efficient multi-resolution planning algorithm with practicality for a real robot.

1. The issue of semantic visual navigation approaches being predominantly evaluated in simulation is recognized and progress is being made to evaluate those methods in the real world through sim-to-real transfer (Deitke et al., 2020; Gervet et al., 2022). However, current efforts are limited to a discrete, ego-centric action space (*e.g.*, *move forward*, *turn left*) and the real-robot evaluations involve only one robot platform compatible with the agent in simulation. Changing the action space or the type of robot platform breaks applicability and generalizability of the trained models, which already suffer in sim-to-real transform on a compatible robot platform. Our approach places no such constraints.
2. I first learned about this dichotomy of generalization articulated by Leslie P. Kaelbling.
3. In the long run (when tackling object search problems involving tool use, for example), the two ends will likely meet somewhere in the middle, I believe.

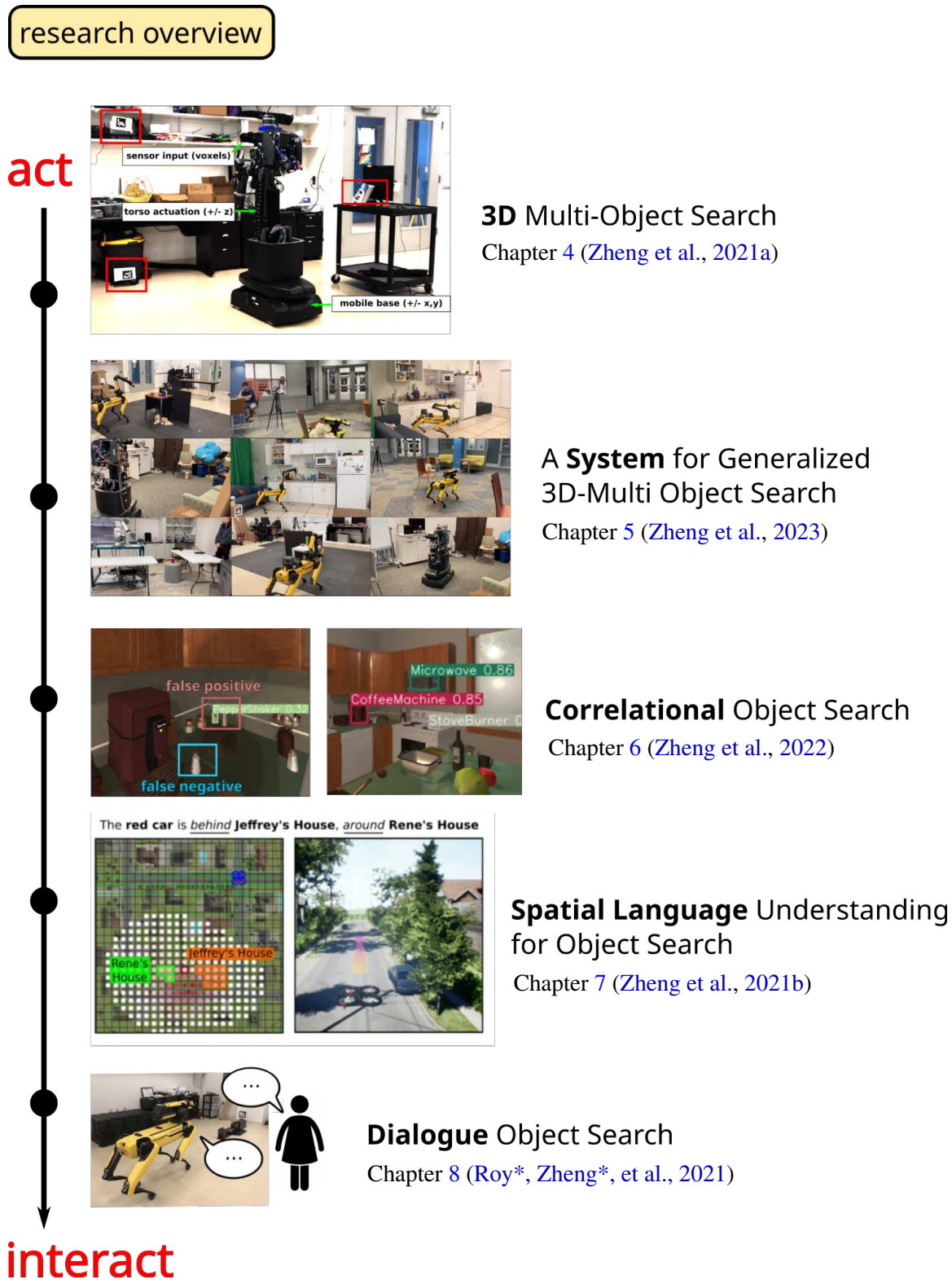


Figure 1.3: Overview of research projects covered in this thesis. The projects are arranged on an axis from robot *acting* in human environments to robot *interacting* with humans. See Section 1.3 for a high-level overview of each project.

- We then design and implement the first robot-independent and environment-agnostic system for generalized 3D multi-object search and deploy it on various robot platforms in different environments.
- We exploit the spatial correlation between easier-to-detect (*e.g.*, microwave) and hard-to-detect objects (*e.g.* pepper shaker) to find hard-to-detect objects faster.
- We integrate spatial language (*e.g.*, "the book is in front of the monitor") as an additional modality of stochastic observation for more efficient search.
- Moving along, we motivate and propose the dialogue object search task and draw insights from a pilot study between pairs of human participants.
- We formulate a general, overarching POMDP model for generalized object search as a "parent model" that ties together the above works on specific object search settings.
- We discuss how the above POMDP model be extended to address tasks involving additional challenges such as object dynamics and environment interaction.
- Last but not least, we provide a literature review on object search from a robotics perspective based on a survey of more than 125 related papers.

Here, we provide a high-level overview to elaborate the above contributions. For 3D multi-object search, we introduce **3D Multi-Object Search (3D-MOS)**, a general POMDP formulation of the problem with volumetric observation space, and we solve it with a novel multi-resolution planning algorithm that uses a new belief representation called **octree belief**. Our work demonstrates that such challenging POMDPs can be solved online efficiently and scalably with practicality for a real robot by extending existing general POMDP solvers with domain-specific structure and belief representation.

As a step further, we design and implement **GenMOS (Generalized Multi-Ob-ject Search)**, a robot-independent and environment-agnostic system of multi-object search in 3D regions. We evaluated the system by deploying it on several robotic platforms, including the Boston Dynamics Spot robot, the Kinova MOVO robot, and the Universal Robotics UR5e robotic arm. Our work makes 3D object search an off-the-shelf capability for different robots in different environments.

To find small, hard-to-detect objects, we propose **Correlational Object Search POMDP (COS-POMDP)**, which can be solved to produce search strategies that exploit spatial correlations between target objects and other objects in the environment that are easier-to-detect. Our experiments were conducted in AI2-THOR (Kolve et al., 2017), a realistic simulator featuring diverse-looking environments of four types: bedroom, bathroom, kitchen and living room, and we use YOLOv5 (Jocher et al., 2020) as the object detector. Results show that our method finds objects more successfully and efficiently compared to baselines, particularly for hard-to-detect objects such as srub brush and remote control.

To consider human input, we integrate spatial language into the object search POMDP model as a form of stochastic observation and derive a spatial language observation model. We call the resulting model **SLOOP (Spatial Language Object-Oriented POMDP)**.

Evaluation based on crowdsourced language data, collected over areas of five cities in OpenStreetMap, shows that our approach achieves faster search and higher success rate compared to baselines, with a wider margin as the spatial language becomes more complex. We demonstrate the proposed method in AirSim, a realistic simulator where a drone is tasked to find cars in a neighborhood environment. We further deploy our system on a Boston Dynamics Spot to accomplish tasks such as that in Figure 1.2.

To move farther along interaction with humans, we motivate and propose the task of **Dialogue Object Search** where a robot must simultaneously decide what to say and how to act while searching for an object in collaboration with a remote human assistant. We conduct a pilot study between human participants and analyze how humans approach the task in the place of the robot.

We tie together the problems and POMDP models proposed in the above works by defining a single, overarching POMDP model (the "parent" model). We briefly discuss in Chapter 9, as future work, how this parent model may be extended to additional challenges and its potential limitations.

Finally, due to the fundamental value and broad applicability of object search yet a lack of survey in this field, we conduct a literature review with the goal of organizing this field from a robotics perspective, clarify the relationships between different problem variants and approaches, and provide a taxonomy of object search.

1.4 Thesis Outline

The remainder of this thesis is organized as follows:

- In Chapter 2, we provide the background material of our research, including a literature review on object search and an introduction of POMDPs from a robotics perspective.
- In Chapter 3, we give the definition of the overarching POMDP model for object search, as hinted in the section above.
- In Chapter 4, we describe our work on 3D multi-object search and focus on addressing the algorithmic challenges of this problem that concludes with a real robot demonstration. This chapter covers the paper *Multi-Resolution POMDP Planning for Multi-Object Search in 3D* (Zheng et al., 2021a).
- In Chapter 5, we describe our work on further addressing the system-level challenges to make 3D multi-object search an off-the-shelf system and package for robots. This chapter covers the paper *A System for Generalized 3D Multi-Object Search* (Zheng et al., 2023).
- In Chapter 6, we describe our work on correlational object search for finding small, hard-to-detect objects such as a pepper shaker or a credit card. This chapter covers the paper *Towards Optimal Correlational Object Search* (Zheng et al., 2022).

- In Chapter 7, we describe our work on interpreting spatial language object search in city-scale environments. This chapter covers the paper *Spatial Language Understanding for Object Search in Partially Observed Cityscale Environments* (Zheng et al., 2021b). We include a follow-up integration of this work with the Boston Dynamic Spot robot to perform object search with spatial language understanding.
- In Chapter 8, we describe our work on introducing the dialogue object search task and a pilot study. This chapter covers *Dialogue Object Search* (Roy*, Zheng*, et al., 2021). Here, * denotes equal contribution.
- In Chapter 9, we conclude this thesis and remark on future work directions.
- Appendix A describes the `pomdp_py` library which supported the implementation of all POMDP models and algorithms proposed in this thesis. It covers the paper *pomdp_py: A Framework to Build and Solve POMDP Problems* (Zheng & Tellex, 2020). The library is available at: `https://github.com/h2r/pomdp-py`.

1.5 Remark on Terminology

For consistency throughout this theis, we use *"viewpoint"* to mean the same thing as *"view pose"*, which is a 6D camera pose. We use *"view position"* to mean the 3D position of the camera, and *"view orientation"* or *"view direction"* to mean the 3D rotation of the camera.

Additionally, we use *"generalized"* instead of *"generalizeable"* in the title of this thesis to emphasize the fact that our POMDP models (*e.g.*, 3D Multi-Object Search) are by definition not specific to any particular robot or environment, and our approach is based on general-purpose online POMDP planning algorithms (with problem-specific structure).

Finally, we refer the reader to Section 2.1.2 for a discussion of *object search* in relation to other problems, some with similar names, to clarify possible confusion.

CHAPTER 2

Background

2.1 Object Search in Robotics: A Review

DIRECTING searchers to find targets is a fundamental problem with broad applicability. Researchers with diverse backgrounds have studied variants of this problem with distinct emphases since the 1970s. If one zooms out, it does not take long before one finds the literature on this topic to be a seemingly hodgepodge. Given such diversity and long history, surveying past work is therefore valuable.

Although this thesis mainly concentrates on how to search within a region, we look broadly here in the background chapter to take on the challenge of surveying this field. The review below is the result of surveying more than 125 papers related to object search, including taxonomies of different aspects of object search.

This chapter is organized as follows:

- We begin by clarifying what the term "object search" entails in robotics, disambiguating it from related and similar problems.
- We provide taxonomies of basic aspects in object search research, including problem settings, methods and systems, and then apply them to organize papers from the field.
- Finally, we briefly summarize the history of research in object search so far.

Difference from Related Reviews

The closest literature reviews compared to ours are Chung et al. (2011) and Robin & Lacroix (2016). Chung et al. (2011) focuses on pursuit-evasion in mobile robotics. They

provide a taxonomy of pursuit-evasion and autonomous search problems that highlights the differences resulting from varying assumptions on the searchers, targets, and the environment, with a discussion of robotic systems. This survey is excellent, yet it has been more than a decade since it is written, and we pay direct attention on object search and not pursuit-evasion.

Robin & Lacroix (2016) reviews major algorithms for different problem settings related to target management, a term that summarizes both target detection and tracking. Search is considered a variant of target detection where the target(s) are actively pursued. However, this survey concerns providing a bird's-eye view of the spectrum of problems from target detection to tracking. In contrast, we focus on object search with the end goal of achieving practical and effective robotic systems for this task.

2.1.1 The Elements of Object Search in Robotics

What does "object search" entail in robotics? Imagine a person searching for their missing pair of glasses at home. It is most likely that the person would move around to search in different places rather than standing at a fixed location, and then be able to identify the glasses somehow, probably through vision. Analogously, when we think of "object search" in robotics, we envision a robot being able to do the same thing. This, in the most basic sense, should involve the following elements:

1. an object to be searched for,
2. an environment where the search happens,
3. a robot that can move itself in order to perceive different aspects of the physical environment that would not be perceivable if the robot does not move, and
4. the robot begins without knowing where the object is, and
5. the robot decides what to do and executes actions sequentially in search for the object.

The above basic set of conditions makes a problem in robotics an object search problem. Variants of object search, such as moving object search, multi-object search, mechanical search (object search by clearing clutter), etc., append additional conditions to this set.

Broadly, object search could refer to the parent problem of all variants. In a narrow sense, it refers to the problem setting in Figure 1.2 (Chapter 1), where no additional conditions are added. Indeed, the variants usually do not come across plainly as object search, but with a more characterizing name. For example, if the object actively moves, then the problem is called *moving object search* (Ding & Castañón, 2018). If the object's placement varies over time, the problem is called *dynamic object search* (Y. Zhang et al., 2019).

2.1.2 Differentiating Object Search from Related Problems

Below, we describe how object search differs from several related problems that might cause confusion. This is not an exhaustive list, but hopefully sufficient for clarification.

Object Detection, Object Tracking, and Object Pose Estimation

Object search differs from problems such as **object detection**, **object tracking**, and **object pose estimation** in that object search is fundamentally a *decision-making problem*, while the rest are *perception problems*. The goal of object detection (Zou et al., 2019) is to identify objects of certain classes given fixed raw sensor input, as an RGB image or a point cloud. Object tracking refers to estimating the trajectory of an object in the image plane or in the 3D space (Yilmaz et al., 2006; Weng et al., 2020). Object pose estimation (Rad & Lepetit, 2017; Xiang et al., 2017) is about estimating the pose (3D or 6D) of objects given sensor input. Both object detection and object pose estimation can serve as important building blocks of an object search system.

Variants of object detection or object pose estimation, such as **active object detection** where the sensor is controlled to maximize detection performance (Xu et al., 2021), do involve decision-making and are often approached with similar techniques as object search, such as next-best view (Doumanoglou et al., 2016), reinforcement learning (Luo et al., 2019), and POMDP planning (Atanasov et al., 2014). However, in contrast to object search, the environment typically revolves around the object to be detected, and the object typically starts out inside the field of view of the sensor. A similar comparison can be made between object search and the problem of **active object reconstruction** (Delmerico et al., 2018).

Object Retrieval and Object Localization

Object retrieval can be both a perception problem, choosing an object that matches a description from a set of objects (Nguyen et al., 2022), or a decision-making problem involving manipulation, where the robot is tasked to grasp and take out an object in clutter (J. Ahn et al., 2022). Object retrieval is a more specific problem than object search since in general, retrieval is not always necessary after the object is found. Often, object search is part of an object retrieval system, considering occlusion from clutter (Bejjani et al., 2021).

The term **object localization** commonly refers to the computer vision problem where the task is to produce a heatmap for an object's location given image (Xue et al., 2019; D. Zhang et al., 2021); **active object localization** (Caicedo & Lazebnik, 2015) has been used to describe object detection models that can actively zoom in on an image for better detection. In robotics, object localization is often a subproblem for object retrieval in clutter (M.-Y. Liu et al., 2012; Du et al., 2021). It has been used as a synonym to object search (Andreopoulos et al., 2010), though such usage is rare.

Target Capture, Target Pursuit and Target Detection

In **target capture** or **target pursuit** (Eaton & Zadeh, 1962), one or a group of agents coordinate to surround one or multiple moving targets on a graph, where capture typically means an agent occupies the same node as the target (Isaza et al., 2008), or the target is enclosed by the agents (R. Sharma et al., 2010). This problem has important practical im-

plications and has been studied by researchers in the game theory (Sticht et al., 1975), graph theory (Svensson & Vegh, 2011), multi-agent control (Bono et al., 2022) communities. If the target is adversarial, this becomes a classical problem called **pursuit-evasion** (Ho et al., 1965). Historically, this line of work has used **moving target search (MTS)** (Ishida & Korf, 1991) as the name of the target capture problem, causing some confusion with the object search problem in robotics.[1] Distinctively, however, object search in robotics differs in that the locations of target objects are not known and uncertainty in robot perception is inevitable, both aspects often not considered in that line of work.[2] Additionally, in robotics, the target and robot states are fundamentally metric and continuous instead of on top of a discrete graph.

In addition, **target detection** is yet another problem with different meanings in different communities. When the target is being actively detected by an agent, the problem is closer to target search or target capture (discussed above) (Robin & Lacroix, 2016; Dadgar et al., 2016). When the detection is passive, the problem is more closely related to sensor placement (O'rourke et al., 1987; Sadeghi et al., 2020), detection mechanism (Koopman, 1956), and signal processing (Tian et al., 2002). The sensors can be sonars (Mukherjee et al., 2011), radars (Bekkerman & Tabrikian, 2006), infrared (C. P. Chen et al., 2013), cameras (Sun et al., 2016), etc.

Semantic Visual Navigation

Several problems share a similar practical motivation as object search but were originated from different fields. Notably, in computer vision, the **ObjectGoal** (Anderson et al., 2018) and **ObjectNav** (Batra et al., 2020) tasks, which are instances of **semantic visual navigation** (Chang et al., 2020), or sometimes just **visual navigation** (Gupta et al., 2017; Wortsman et al., 2019), can be viewed as object search problems as well.[3] Broadly speaking, object search is a more general problem than semantic visual navigation as it does not require vision input or searching by navigation. Setting aside this technicality, researchers in the two communities tend to make different assumptions in problem settings, leading to the use of different methods. Many works in object search are planning-based, so a model of the environment is given, and the evaluation culminates at real robot demonstrations. On the other hand, end-to-end deep learning has been the predominant method for semantic visual navigation as the agent searches in an unknown environment and leverages simulators for large-scale training and benchmark testing (*e.g.*, AI2THOR by Kolve et al. (2017),

1. Searching for "target search" on the web leads to papers from this community, although the literal meaning of the phrase is nearly identical to object search.
2. Despite not considering uncertainty in perception, research in target capture strategy is valuable especially when the agents are, for example, human teammates.
3. Semantic visual navigation is the problem where an agent is placed in an unknown environment and tasked to navigate towards a given semantic target (such as "kitchen" or "chair"). The agent typically has access to behavioral datasets for training on the order of millions of frames and the challenge is typically in generalization. Practically, semantic visual navigation is motivated as a home robot application (Batra et al., 2020) and serves the same purpose as object search using cameras in robotics.

Habitat by Ramakrishnan et al. (2021), and Gibson by Xia et al. (2018), ThreeDWorld by (Gan et al., 2020)). Both communities can learn from each other, although interactions have currently been limited.[4] Nevertheless, it is hopeful that as both approach the common goal of a general robotic system for object search, more interaction will occur. This thesis takes a step in that direction.

Active Visual Search

The terms **visual search** (J. Vogel & Murphy, 2007) **visual object search** (Druon et al., 2020), **active visual search** (Sjöö et al., 2012), **active visual object search** (Aydemir et al., 2011; Zeng et al., 2020), and **active object search** (Elfring et al., 2013), are all synonymous to object search using a camera in robotics. Since cameras are predominantly used by robots for perceptual tasks such as face recognition and object detection in human environments, "visual" is often omitted (*e.g.*, in Y. Ye & Tsotsos (1997)). Since object search in robotics is inherently an active process, "active" is also often omitted. Note that in psychological testing (Erickson, 1964; Williams, 1967; Duncan & Humphreys, 1989) and computer vision (Tsotsos, 1992; Eckstein, 2011), "visual search" has historically referred to what we call object detection today. So, it makes sense that researchers coming from those backgrounds prefix it with "active" to mean object search by a robot. For the purpose of this thesis, we simply use object search (see Section 2.1.1 for elaboration).

2.1.3 A Taxonomy of Object Search Problem Settings

There are several dimensions along which we can classify object search problem settings:

1. number of objects: single or multiple;
2. number of robots: single or multiple;
3. property of the object(s): *e.g.* static, dynamic, or adversarial;
4. property of the robot(s): *e.g.* camera-only mobile robot (including drones), eye-in-hand robotic arm, or mobile manipulator;
5. property of the environment: *e.g.* unknown, cluttered.

Due to the large number of dimensions, it quickly gets out of hand if we try to enumerate all combinations. Instead, for clarity, we consider the following naming pattern:

[count]	[object property]	object search	[other property]
secondary	primary		tertiary

4. For example, one improvement of ObjectNav over ObjectGoal is the introduction of *intentionality* as a success criterion, which requires the agent to signal that the object is found. The same issue has been previously considered in the object search literature in robotics (J. K. Li et al., 2016; Wandzel et al., 2019).

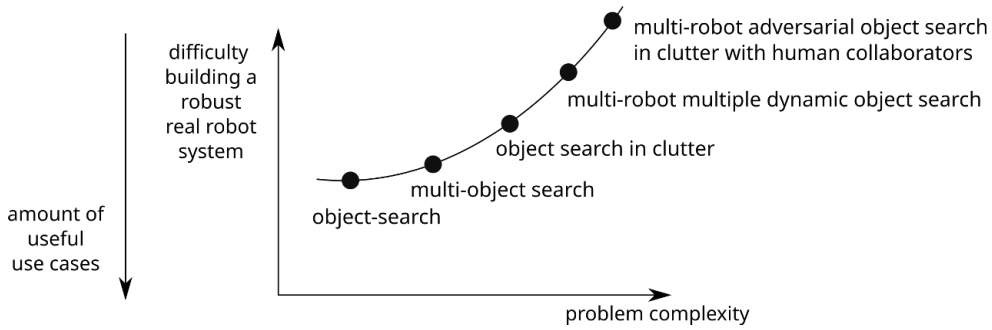


Figure 2.1: A sketch of the relationship between the complexity of object search variants and difficulty in building an actual system. A system that performs basic object search is arguably more broadly applicable than an intricate variant.

This naming rule consists of four parts. The three parts surrounding "object search" are used to characterize the specific problem setting, effectively appending conditions to the basic set that define object search laid out in Section 2.1.1. These parts are organized by the following ranking:[5]

- **Primary: object property.** An adjective that describes the object's property. If not provided, this adjective is assumed to be "static."
- **Secondary: count.** An adjective that describes the number of robots and the number of objects, in that order. If not provided, then both are assumed to be "single." For example, we would say, "multi-robot multi-object search" rather than "multi-object multi-robot search."[6]
- **Tertiary: other property.** A noun phrase that characterizes the robot, the environment, and (or) the task. For example, "open-world object search" means a robot explores in an open-world while performing search;"object search in clutter" implies that the environment is cluttered and the robot likely can manipulate clutter.

Figure 2.1 is a sketch of the relationships between different variants in terms of complexity, system-level difficulty and applicability. In Table 2.3, we apply this taxonomy to organize object search problem settings.

2.1.4 A Taxonomy of Object Search Methods

As object search is a sequential decision-making problem, common approaches to this class of problems can be applied. We can therefore categorize the object search methods based

5. The ordering is analogous to a radiation field centered at "object search", such that the radiation goes from attributes of objects to attributes of the robot, finally to the attributes of the environment.
6. In general, clarity should come first especially in paper narration. That means instead of saying "multi-robot multiple dynamic object search," it is better and more natural to say "multi-robot search for dynamic objects." The fact that consistent problem naming is difficult shows the multitude of object search variants.

on the type of decision-making algorithm used to generate the search strategy:

1. heuristics-based (*e.g.*, bio-inspired);
2. mathematical analysis;
3. greedy, next-best view;
4. graph search;
5. POMDP planning;
6. reinforcement learning.

We elaborate the above methods through examples.

Heuristics-based. Goldsmith & Robinett (1998) studies multi-robot search for a static object (a threat such as an explosive) and proposes an ad-hoc strategy that divides the robots into two social groups: alpha and beta. Robots in the alpha group are risk takers, while those in the beta group are conservative. Each group has a distinct rule-based policy for when and when not to move. This kind of method is typically viewed as less favorable compared to more principled approaches, but may work well in practice.

Mathematical analysis. Pollock (1970) formulated arguably the first model for moving object search where a target moves between two cells according to a Markovian model. The problem here is framed as finding an allocation of the amount of effort (*e.g.*, time) spent searching at each cell, which is solved it through dynamic programming. Stone (1976) generalized the problem and provides a thorough introduction on this topic. In this formulation, the searcher is given a probability distribution over the target's initial location, and a detection function that characterizes the probability density of detecting the target given a target location and an amount of effort. Later, Mangel (1981) considered the case where the target moves as a diffusion process. Instead of finding a search plan (*i.e.*, searcher's trajectory), the problem is to describe the probability density function $f(x, S, t)$ that search has been unsuccessful up to time t following a trajectory S, given that the target is at x at t (the so-called "descriptive problem"). Analytical methods were dominant in the early days when object search was studied in operations research, but results from this body of work are often general yet abstract. Direct transfer to real robotic systems has been rare.

Greedy, next-best view. Y. Ye & Tsotsos (1997) formulated 3D object search as choosing sensing actions over time under a cost budget; each action controls camera parameters such as position and zoom in some way and has an associated cost and probability of detecting the target. This is shown to be NP-Complete via a straightforward reduction to the Knapsack problem. Follow-up works hence often favor a greedy search strategy to build robotic systems for object search (Tsotsos et al., 1998; Shubina & Tsotsos, 2010; Andreopoulos et al., 2010; Rasouli, 2015). Andreopoulos et al. (2010) builds a vision-based system that enables a 26-DOF humanoid robot (the Honda ASIMO) to find an object (*e.g.*cup) in a 3D region. The system maintains a probability map over a $4\times4\times4$ grid, and uses a greedy, one-step look-ahead algorithm to search. This algorithm first hypothesizes a set of candidate sensor states, each made up of a body pose and a gaze pose, and then selects a sensor state that maximizes the probability of localizing the target position.

Greedy, next-best view algorithms are commonly used, often simple to implement and perform well in practice. Nevertheless, when the search actions carry different costs, or when multiple objects need to be searched for, sequential planning can consider important factors regarding action ordering that a greedy method does not.

Graph search. It is uncommon, but graph search algorithms such as A* have been used for object search, primarily when searching for occluded object in clutter (Dogar et al., 2014; Y.-C. Lin et al., 2015). Here, the search process is thought of as revealing occluded space by removing a sequence of visible obstacles until the target object becomes visible, where each removal action may carry a different cost.[7] The greedy approach is shown to be suboptimal (Dogar et al., 2014). Instead, imagine a graph where a node is a subset of all visible objects, and an edge indicates the removal of a single object. Then, the object search problem becomes a graph search problem of finding the shortest path from the current node[8] to the node corresponding to $\varnothing$ (all space revealed, target is therefore considered found). This is the approach taken by both Dogar et al. (2014) and Y.-C. Lin et al. (2015). It is elegant, yet requires the obstacles to be taken off from the environment at every step and does not consider uncertainty in perception.

POMDP planning. N. Roy et al. (2005) uses belief compression to trade off belief space dimensionality and information loss, allowing value iteration to be feasible in enabling a mobile robot to efficiently search for a person in a hallway-and-room environment. More recently, advances in online POMDP planning algorithms have sparked research interest in applying them for object search (*e.g.*, DESPOT (Somani et al., 2013) in J. K. Li et al. (2016) and POMCP (Silver & Veness, 2010) in Wandzel et al. (2019)); refer to Section 2.2.4 for a closer look at these algorithms.) For multi-robot search, POMDP planning was unfavored computationally by Hollinger et al. (2009) but continue to receive attention as both fields progress (Rizk et al., 2019). The appeal of POMDP planning-based approaches is in their generalizability, versatility, and reasoning over action sequences. Unpleasant theoretical results (*e.g.*, (Madani et al., 1999)) and practical computational challenges have been the main reasons of criticism to this kind of approach. Previous object search works have sacrificed realism for efficiency (*e.g.*, reducing the problem from 3D to 2D). This thesis uses POMDP's strengths in robot task modeling and mitigates its weaknesses through exploiting structures to achieve practical, effective object search systems.

Reinforcement learning. Y. Zhu et al. (2017) is a seminal work in semantic visual navigation[9] that trained a goal-conditioned actor-critic model in the AI2-THOR simulator and demonstrated sim-to-real generalization to a mobile robot. A large body of work followed and developed various end-to-end, model-free techniques (Wortsman et al., 2019; Mousavian et al., 2019; K. Chen et al., 2019; Chaplot et al., 2020; Liang et al., 2021); Sample complexity and real-world generalization are hurdles to overcome for this approach. Recently, D. Shah et al. (2022) achieved impressive outdoor visual navigation results on a

7. The costs differ since removing different obstacles may affect visibility (how much volume will be revealed) and accessibility (how accessible other obstacles will become) in different ways.
8. The current node corresponds to the current state (*i.e.*, current subset of visible objects).
9. See Section 2.1.2 (page 13) for a discussion on semantic visual navigation in relation to object search.

mobile robot through integrating pre-trained models like CLIP (Radford et al., 2021) with a navigation planning pipeline (no sim-to-real). Separately, in target pursuit, Shkurti et al. (2018) proposed a model-based reinforcement learning approach using POMDP planners.[10] Although our work uses learned object detectors, we formulate POMDP models analytically. In the long run, inspired by AlphaGo (Silver et al., 2017) and AlphaZero (Silver et al., 2018), we believe that combining online tree search-based POMDP planning with learned models is a promising vein for research in object search, especially when physical interaction in unstructured environments is involved.

2.1.5 A Taxonomy of Object Search Systems

Despite often motivated from in a robotics context, papers in object search without any real robotic system demonstration outnumbers those that do. Nevertheless, the state-of-the-art in object search is best reflected by what researchers can make real robots do, which do vary a lot. It is therefore useful to come up with a taxonomy that captures a spectrum of object search systems developed to evaluate object search algorithms.

It is straightforward that on one end of the spectrum, the system is idealistic; for example, the robot and the target are points on a plane. It is more difficult to determine what is the other end. Note that we are not classifying the object search system's performance,[11] but where it is deployed in. This includes considerations about what robotic capabilities are involved, and what environments the system is expected to operate in.

Therefore, we consider the following taxonomy to categorize object search systems:

S0	idealistic simulation
S1	realistic simulation
N0	single, navigation-only robot in a single environment
N1	single, navigation-only robot in different environments
N2	different navigation-only robots in different environments
M0	single, manipulation-only robot in a single environment
M1	single, manipulation-only robot in different environments
M2	different manipulation-only robot in different environments
R0	single, mobile manipulator robot in a single environment
R1	single, mobile manipulator robot in different environments
R2	different real mobile manipulator robot in different environments

10. Note that the technique of model-based reinforcement learning with POMDP planning is equivalent to that of POMDP planning with learned models (instead of analytical). However, typically in reinforcement learning, the agent starts with no knowledge of the world and learns from scratch.
11. We do not attempt a taxonomy of the system's performance because one could always hide a target object so that a robot, or even a person, would fail to find it. Therefore, a system that "always finds objects" is not a realistic goal. The highest expectation in performance is that the robot can find objects as efficiently or more efficiently than a person, which is something we are so far away from today to be worth classifying.

Table 2.1: A taxonomy of object search systems

This taxonomy is inspired by how levels of autonomous driving are concisely represented (from L0 to L5) (Briney, 2004). The name of each category (*e.g.*, S0, N1, etc.) consists of a letter that represents the type of system and a number that represents the level of generalization of that system. To elaborate, "S" = simulation, "N" = navigation-only, "M" = manipulation-only, and "R" = mobile-manipulator (both navigation and manipulation). The number starts from 0, goes up to 1 for "S", and up to 2 for the rest.

A **realistic simulation** must be 3D, and the objects, motion, and events in the simulator can easily be relatable to their counterparts in the real world. **"Navigation-only"** means no manipulation is involved *during* the search process; This includes common mobile robots but also, for example, an eye-in-hand robotic arm that only searches by looking through its gripper camera but does not manipulate its surroundings. **"Manipulation-only"** means the robot does *not* have a mobile base and it manipulates the environment *during* search, such as removing clutter or opening containers. **"Mobile manipulator"** means the robot has a mobile base *and* it manipulates the environment during search. Note that this does not require the robot to have a robotic arm – a mobile robot that knows how to use its body to clear up movable obstacles in search for an object also counts.

Note that we do not have a category for "different types of [...] robots in a single environment." This is because different robots typically are designed to be more suitable for operation in some environments than others. It is more natural and sufficient to have a robot be able to perform search in environments suitable for it. If a system can achieve this for different robots, it achieves level 2. We provide an imaginary example for each below:

S0	*the robot and the target are points on a plane or grids in a grid world.*
S1	*AI2-THOR, Habitat, Gibson, ThreeDWorld*
N0	*a turtlebot searches in an office.*
N1	*a turtlebot searches in an office and in a kitchen,*[12] *or two different offices*
N2	*a turtlebot searches in an office and a drone searches outdoors.*
M0	*a Panda arm searches over a cluttered tabletop.*
M1	*a Panda arm searches over a cluttered tabletop and over a cluttered shelf.*
M2	*a Panda arm searches over a cluttered tabletop, and a UR5e robot searches over a cluttered shelf.*
R0	*a MOVO searches in a room with a cluttered tabletop.*
R1	*a MOVO searches in a room with a cluttered tabletop and in a room with two cabinets.*
R2	*a MOVO searches in a room with a cluttered tabletop, and a Spot searches in a room with two cabinets.*

Table 2.2: Examples for the categories in the taxonomy; Application in Table 2.3

12. This emphasizes the capability to search in different environments, not necessarily simultaneously.

2.1.6 Application of the Taxonomies for Literature Survey

Below, we follow the taxonomies introduced in the previous sections to categorize papers from the literature, which concludes this review. We only consider papers that assume the target location to be initially unknown (excluding some papers on target capture). The result of our survey is in the table below. We divide up problem names applying the three-level taxonomy and we grouping references by methods used. The system classification is indicated by the superscript.

Notably, to the best of our knowledge, our work (Zheng et al., 2023) is the first system to achieve N2 among the class of navigation-only robots. For manipulation-only robots, the current state of the art is limited to at most M0. Xiao et al. (2019) is the first and only (so far) that achieves R0, in which a mobile manipulator robot, Fetch (Wise et al., 2016), searches for a target object in a tabletop scene by removing clutter and looking from different viewpoints. However, the setup is restricted to only two manually specified viewpoints, one on each side of the table, from where the entire scene is captured under the field of view. Works on multi-robot search and moving object search are yet to reach N1, or any manipulation-involved search task. No work has reached R1 or R2, which can be regarded as the holy grail in terms of object search systems.

THE TABLE STARTS FROM THE NEXT PAGE.

name	references
(Static) object search	**SINGLE-OBJECT** **heuristics-based:** (Nomatsu et al., 2015)[N0] (Izquierdo-Cordova et al., 2016)[S0], (Wixson & Ballard, 1994)[N0] **analytical:** (Y. Ye & Tsotsos, 1997)[N0] (Hernandez et al., 2021)[S1] **greedy:** (Y. Ye & Tsotsos, 1997)[N0] (Gelenbe & Cao, 1998)[S0] (Saidi et al., 2007)[N0] (Kollar & Roy, 2009)[N0] (Andreopoulos et al., 2010)[N0] (Shubina & Tsotsos, 2010)[N0] (Aydemir et al., 2011)[N1] (X. Chen & Lee, 2013)[N0] (Zeng et al., 2020)[S1,N0] **graph search:** (Gelenbe & Cao, 1998)[S0] (Song et al., 2020)[S0] (Y. Zhang et al., 2021)[S1,N1] (D. Shah et al., 2022)[N1] **POMDP planning:** (J. Vogel & Murphy, 2007)[S1] (Aydemir et al., 2013)[N1] (Lu et al., 2018)[N0] (C. Wang et al., 2018)[S1,N0] (Holzherr et al., 2021)[S0] (Zheng et al., 2022)[S1] **SINGLE-OBJECT (UNKNOWN ENVIRONMENT)** **heuristics-based:** (Y. Li et al., 2022)[S1,N0] **graph search:** (Joho et al., 2011)[N1] (Tsuru et al., 2021)[N0] **POMDP planning:** (Y. Wang et al., 2020)[S1] (Giuliari et al., 2021)[S1] **reinforcement learning:** (Y. Zhu et al., 2017)[S1,N0] (X. Ye et al., 2018)[S1,N0] (Mousavian et al., 2019)[S1] (K. Chen et al., 2019)[S1] (Chaplot et al., 2020)[S1] (Schmid et al., 2019)[S1] (Qiu et al., 2020)[S1] (Liang et al., 2021)[S1] (Schmalstieg et al., 2022)[S1,N0] (M. Zhu et al., 2022)[S1,N0] **SINGLE-OBJECT (IN CLUTTER)** **heuristics-based:** (Huang et al., 2022)[M0] **analytical:** (Wong et al., 2013)[S1] **greedy:** (Moldovan & De Raedt, 2014)[S1] **graph search:** (Dogar et al., 2014)[S0,M0] (Y.-C. Lin et al., 2015)[M0] (Nam et al., 2019)[S0,M0] (Huang et al., 2021)[S1,M0] **POMDP planning:** (J. K. Li et al., 2016)[S1] (Nie et al., 2016)[S0] (Xiao et al., 2019)[S1,R0] (Zhao & Chen, 2021)[S1] **reinforcement learning:** (Novkovic et al., 2019)[S1,M0] (Danielczuk et al., 2019)[M0] (Kurenkov et al., 2020)[S1] (Bejjani et al., 2021)[S0,M0] **MULTI-OBJECT** **POMDP planning:** (Wandzel et al., 2019)[S0,N0] (Zheng et al., 2021a)[S0,N0] (Zheng et al., 2021b)[S1,N0] (Zheng et al., 2023)[S1,N2]

Multi-robot (static) object search	SINGLE-OBJECT **heuristics-based:** (Goldsmith & Robinett, 1998)^{S0} (G. Zhang & Garg, 2008)^{S0} SINGLE-OBJECT (UNKNOWN ENVIRONMENT) **particle swarm optimization:** (Shirsat et al., 2020)^{S1} (Tang et al., 2021)^{S0} (Ebert et al., 2022)^{S0} MULTI-OBJECT **heuristics-based:** (Rybski et al., 2002)^{N0} **analytical:** (Czyzowicz et al., 2016)
Moving object search	SINGLE-OBJECT **analytical:** (Pollock, 1970) (Dobbie, 1974) (Stone, 1979) (Washburn, 1980) (Dogan & Zengin, 2006)^{S0} (Fidan et al., 2013) (Radmard & Croft, 2017)^{N0} **POMDP planning:** (N. Roy et al., 2005)
Multi-robot moving object search	SINGLE-OBJECT **heuristics-based**: (Hereford & Siebold, 2010)^{S0,N0} (Kulich et al., 2015)^{S0} **analytical:** (Zengin & Dogan, 2011)^{S0} **greedy:** (Sarmiento et al., 2004)^{S0} (Bourque, 2019)^{S0} (Kulich et al., 2015)^{S0} **graph search:** (Hollinger et al., 2009)^{S0,N0} SINGLE-OBJECT (UNKNOWN ENVIRONMENT) **graph search**: (Marjovi et al., 2009)^{S0,N0} (Kulich et al., 2015)^{S0} **particle swarm optimization:** (Tang et al., 2021)^{S0} MULTI-OBJECT **analytical:** (Dames, 2020)^{S0} MULTI-OBJECT (UNKNOWN ENVIRONMENT) **heuristics-based:** (Baxter et al., 2007)^{S0}
Ethical issues in object search[13]	(Sharkey & Sharkey, 2011) (Harbers et al., 2017) (Battistuzzi et al., 2021)

Table 2.3: Organization of papers from the object search literature using the proposed taxonomies

13. The topic "ethical issues in object search" is not technically an object search problem, but it is an important dimension to it. Researchers referenced have started to discuss ethical challenges related to robot's help in search and rescue; helps by robots are not equally good.

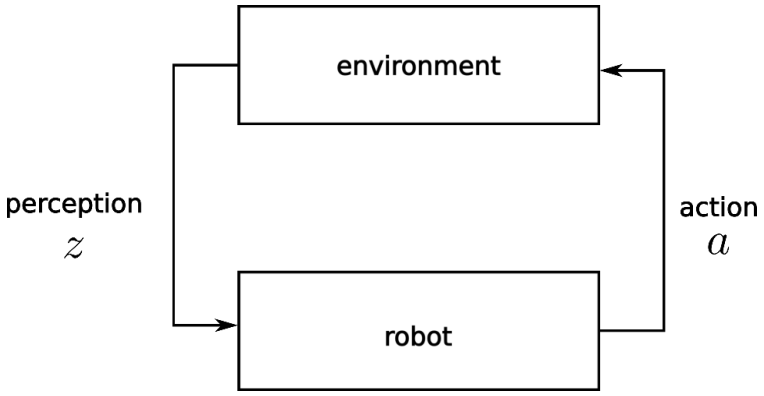


Figure 2.2: The perception-action loop. A robot is a system of sensors and actuators. The sensors can receive observations (perception), and the actuators can change the configuration of the robot, and perhaps, the external environment (action), which affects subsequent observations.

2.2 Partially Observable Markov Decision Process

POMDP was originally introduced to model control systems with incomplete state information in applied mathematics and operations research (Åström, 1965; Sondik, 1971; Smallwood & Sondik, 1973) and later introduced to robotics by Kaelbling et al. (1998).[14] A POMDP models a sequential decision making problem where the environment state is not fully observable by the agent, which is almost always the case if the agent is a robot in a human environment. As a result, it has gained popularity in robotics.

In this section, I first motivate the use of POMDP in robotics from first principles. Then, I provide a precise definition of a POMDP and discuss different methods to obtain policies to a POMDP with an emphasis on practicality. A more comprehensive literature review of POMDP in robotics can be found in Thrun et al. (2005); Kurniawati (2022); Lauri et al. (2022).

2.2.1 Motivation from a Robotics Perspective

A robot is a system of sensors and actuators[15] that operates in an environment. The robot can perform actions through its actuators and receive observations from its on-board sensors. The relationship between the robot and the environment can be illustrated by the perception-action loop (Figure 2.2), common in the sequential decision making literature (Littman, 1996; Kochenderfer, 2015).

14. See Littman (2009) for an excellent summary of the early literature on POMDPs.
15. "Actuator" here means more than mechanical ones, including *e.g.*, speakers, virtual messaging, etc.

Suppose the robot can choose an action from a set of possible actions, denoted as $\mathcal{A}$, and suppose it receives observations from a set of possible observations is denoted as $\mathcal{Z}$. The perspective in Figure 2.2 tells us that in the lifetime of a robot, it experiences a sequence of actions and observations $a_1, z_1, a_2, z_2, \cdots$ (and nothing else). Define the history at time t to be the sequence of *past* actions and observations, denoted as $h_t = (az)_{1:t-1}$.

The robot's purpose is to produce a sequence of actions that best achieves some task goal (over its lifetime). Suppose that the robot starts performing the task at time t.[16] There are two questions here: how to represent the goal and how to choose an action at each step. Assuming that the robot behaves rationally, a common idea to address both questions is to imagine that the robot observes a numeric *reward* each time it takes an action.[17] Note that this reward is conditioned on the history of the robot's experience rather than just the action taken. Then, the utility of an action sequence $a_t, a_{t+1}, \cdots$ can be defined as the sum of all rewards $r_t + r_{t+1} + \cdots$ (cumulative rewards). To limit this sum to be finite, we introduce a discount factor $\gamma \in [0, 1)$. So, instead of observing r_{t+k} ($k \geq 0$), the robot observes $\gamma^k r_{t+k}$. The utility is then $\gamma^0 r_t + \gamma^1 r_{t+1} + \cdots$, the discounted cumulative reward.[18]

This utility establishes the ground for rational choice over action sequences by the robot. As a consequence, following the utility theorem by Von Neumann & Morgenstern (1947), the robot shall behave as if it is maximizing the expectation of the utility, that is, maximizing the expectation of the discounted cumulative reward conditioned on history h_t:

$$\mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| h_t\right] \tag{2.1}$$

As mentioned, each reward should be conditioned on the *history* of the robot's experience. Intuitively, however, it is inconvenient to specify the goal based directly on the sequence of actions and observations the robot has experienced. This is because the robot's performance may depend on factors external to the robot, and accounting for all enumerations of history quickly gets out of hand.

Instead, define the notion of a *state*, denoted by $s \in \mathcal{S}$, which shall naturally capture the necessary and potentially external information to specify a goal through a *reward function* R such that $r_t = R(s_t, a_t)$. For example, if the robot's goal is to clean a room, then whether there still exists a dirty spot in the entire room could be a useful piece of information to include in the state, even though the robot does not observe that information directly. Clearly, the robot's action may cause the state to change.

Since the robot does not observe the state (partial observability), the robot can at most maintain a distribution over states given what it knows, that is, the history. This distribution is called the *belief state*, denoted $b_t(s) = \Pr(s|h_t) \equiv \Pr(s|b_t)$, where b_t acts as a sufficient

16. This is in fact the setting of online POMDP planning.
17. Whether rewards are sufficient to express all goals we would ever want a robot to achieve is a topic of ongoing research (Silver et al., 2021; Abel et al., 2021; Bowling et al., 2022). This discussion is entirely out of scope for this thesis.
18. Or, synonymously, the discounted return.

statistic for history h_t. The robot begins with an initial belief $b_1(s) = \Pr(s)$ based on prior knowledge, as the corresponding history h_1 is empty.

Now, define $V(b_t)$ as the expected utility of the optimal behavior at time t:

$$V(b_t) = \max_{a_t, a_{t+1}, \cdots \in \mathcal{A}} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| b_t\right] \tag{2.2}$$

Note that the expectation in Equation 2.2 is a variant of that in Equation 2.1 using the fact that b_t is a sufficient statistic for h_t. We call V the *value function* and the value at b_t means the maximum expected utility at b_t. From the definition of V in Equation 2.2, we can derive the following expression, which turns out to be the Bellman equation for POMDPs:[19]

$$V(b_t) = \max_{a_t \in \mathcal{A}} \left\{ \sum_{s \in \mathcal{S}} b_t(s) R(s, a_t) + \gamma \sum_{z \in \mathcal{Z}} \Pr(z|b_t, a_t) V(b_{t+1}^{a_t, z}) \right\} \tag{2.3}$$

where $b_{t+1}^{a_t,z}(s) = \Pr(s_{t+1}|b_t, a_t, z)$ is the result of recursive Bayesian state estimation (*i.e.*, belief update) based on b_t after taking action a_t and receiving observation z at time t. I provide a derivation of the Bellman equation in the appendix to this chapter (Section 2.2.5). Equation 2.3 incorporates into the robot decision making objective two major types of uncertainty, **partial observability**, through the belief state b_t, and **perceptual uncertainty**, through the probability distribution $\Pr(z|b_t, a_t)$.

It is, however, difficult to define $\Pr(z|b_t, a_t)$ directly. Instead, we can do the following expansion to derive the components that depend on states which are easier to specify:

$$\Pr(z|b_t, a_t) = \sum_{s \in \mathcal{S}} b_t(s) \sum_{s' \in \mathcal{S}} \Pr(s'|s, a_t) \Pr(z|s', a_t) \tag{2.4}$$

Define $O(s', a, z) = \Pr(z|s', a)$ and call that the *observation model*. Then define $T(s_t, a, s') = \Pr(s'|s, a)$ and call that the *transition model*, which exhibits the Markov property. It so happens that a POMDP is formally defined as a tuple $\langle S, A, \mathcal{Z}, T, O, R, \gamma \rangle$. This framework therefore encapsulates through first principles what a robot must necessarily consider to accomplish tasks during its lifetime.

Next, we provide a typical introduction of a POMDP for a quick review.

2.2.2 Formal Definition of POMDP

A POMDP models a sequential decision making problem where the environment state is not fully observable by the agent. It is formally defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{Z}, T, O, R, \gamma \rangle$, where $\mathcal{S}, \mathcal{A}, \mathcal{Z}$ denote the state, action and observation spaces, and the functions $T(s, a, s') = \Pr(s'|s, a)$, $O(s', a, z) = \Pr(z|s', a)$, and $R(s, a) \in \mathbb{R}$ denote the transition, observation, and reward models. The agent takes an action $a \in \mathcal{A}$ that causes the

19. In general, $\mathcal{S}$, $\mathcal{A}$ and Ω can be continuous.

environment state to transition from $s \in \mathcal{S}$ to $s' \in \mathcal{S}$. The environment in turn returns the agent an observation $z \in \mathcal{Z}$ and reward $r \in \mathbb{R}$. A *history* $h_t = (az)_{1:t-1}$ captures all past actions and observations. The agent maintains a distribution over states given current history $b_t(s) = \Pr(s|h_t)$. The agent updates its belief after taking an action and receiving an observation by recursive Bayesian state estimation:

$$b_{t+1}(s') = \eta \Pr(z|s', a) \sum_{s \in \mathcal{S}} \Pr(s'|s, a) b_t(s) \tag{2.5}$$

where $\eta = \sum_s \sum_{s'} \Pr(z|s', a) \Pr(s'|s, a) b_t(s)$ is the normalizing constant. The solution to a POMDP is a *policy* π that maps a belief state or the corresponding history to an action. The *value* of a POMDP at a belief under policy π is the expected discounted cumulative reward following that policy:

$$V_\pi(b_t) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, \pi(b_{t+k})) \middle| b_t\right] \tag{2.6}$$

where $\gamma \in [0, 1)$ is the discount factor. The optimal value at belief b_t is $V(b_t) = \max_\pi V_\pi(b_t)$. Equation 2.6 can also be written equivalently as $V_\pi(h_t) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, \pi(h_{t+k}))|h_t]$

2.2.3 Object-Oriented POMDP

An Object-Oriented POMDP (OO-POMDP) (Wandzel et al., 2019) (generalization of OO-MDP (Diuk et al., 2008)) is a POMDP that considers the state and observation spaces to be factored by a set of n objects, $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$, $\mathcal{Z} = \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_n$, where each object belongs to a class with a set of attributes. A simplifying assumption is made for the 2D multi-object search domain (Wandzel et al., 2019) that objects are independent so that the belief space scales linearly rather than exponentially in the number of objects: $b_t(s) = \prod_i b_t^i(s_i)$. We make this assumption for the same computational reason when we formulate multi-object search in 3D (Chapter 4).

2.2.4 Obtaining Policies to POMDPs

General-purpose algorithms for POMDP planning can be grouped into offline algorithms and online algorithms. Data-driven approaches (*e.g.*, end-to-end deep reinforcement learning) can be applied to learn POMDP policies for domains where POMDP models are hard to define. Here, we provide an overview for the ideas behind major POMDP planning algorithms.[20] We spend more effort on tree search-based online algorithms as we use them to produce object search policies, due to scalability to large domains, asymptotic optimality and ease of implementation. For a more detailed review, please refer to Lauri et al. (2022).

20. We remarked on learning-based approaches for POMDP in our taxonomy of object search methods (Section 2.1.4, page 15); Refer to Arulkumaran et al. (2017) and Mousavi et al. (2016) for more discussions.

Offline Planning with Exact and Point-based Methods

Exact methods for solving POMDP include linear programming (Smallwood & Sondik, 1973) and value iteration (Cassandra et al., 1994). The basis of exact methods is that the value function of a POMDP is piecewise linear and convex. To elaborate, the value function under policy π as in Equation 2.6 can be written as a dot product:

$$V_\pi(b_t) = b_t \cdot \alpha_\pi \tag{2.7}$$

where $b_t \cdot \alpha_\pi = \sum_{s \in \mathcal{S}} b_t(s)\alpha(s)$ and α_π is called an α-vector (Smallwood & Sondik, 1973). Each element $\alpha_\pi(s)$ equals to the expected discounted cumulative reward for state trajectories $sa_1o_1a_2o_2\cdots$ starting at s, with actions from π and observations according to O:[21]

$$\alpha_\pi(s) = R(s, a_i) + \gamma \sum_{s' \in \mathcal{S}} T(s, a_i, s') \sum_{z_i \in \mathcal{Z}} O(s', a_i, z_i)\alpha_\pi(s') \tag{2.8}$$

Geometrically, V_π is a hyperplane over the belief space, and the optimal value function can be viewed as the upper envelope of the hyperplanes corresponding to all α-vectors, which is piecewise linear and convex (refer to Kaelbling et al. (1998) for an illustration).

Exact methods compute all α-vectors that form the optimal value function. They often produce policies that exhibit interesting behavior but are often too slow to be practical for large domains (Ross et al., 2008). In fact, solving POMDPs exactly is likely unwise for robotics problems since it is PSPACE-complete (Papadimitriou & Tsitsiklis, 1987), and undecidable whether a desirable solution exists (Madani et al., 1999).

Point-based methods (Pineau et al., 2003; Spaan & Vlassis, 2005; Kurniawati et al., 2008) take a different approach. Instead of computing the optimal value function over the entire belief space, a set of belief states (*i.e.*, points) are selected, and one α-vector is maintained per belief state. The set of α-vectors then approximates the optimal value function. Point-based methods allow computing policies offline with tunable approximation and are of sustained interest in robotics (Lauri et al., 2022), yet their effectiveness is limited to small domains due to the intractability in belief update (Equation 2.5) for larger domains.

Online Planning with Sparse Tree Search-based Methods

In contrast to offline planning which aims to compute or approximate the optimal policy given any belief state, online planning interleaves planning and execution, and cares about outputting the action to be executed by the robot given its current belief state b_t.

The idea behind sparse tree search-based algorithms for online POMDP planning is simple and appealing (Figure 2.3). Online POMDP planning can be thought of as computing the Q-value $Q(b_t, a)$ for any $a \in \mathcal{A}$, *i.e.*, the expected return after taking action a

21. The idea is that the policy π can be thought of as encoding a set of trees, each rooting at some belief b and a path from the root $ba_1o_1a_2o_2\cdots$ represents a belief trajectory with actions from π and observations from $\mathcal{Z}$. Given a state s sampled from b, a trajectory $sa_1o_1a_2o_2\cdots$ can be defined the same way.

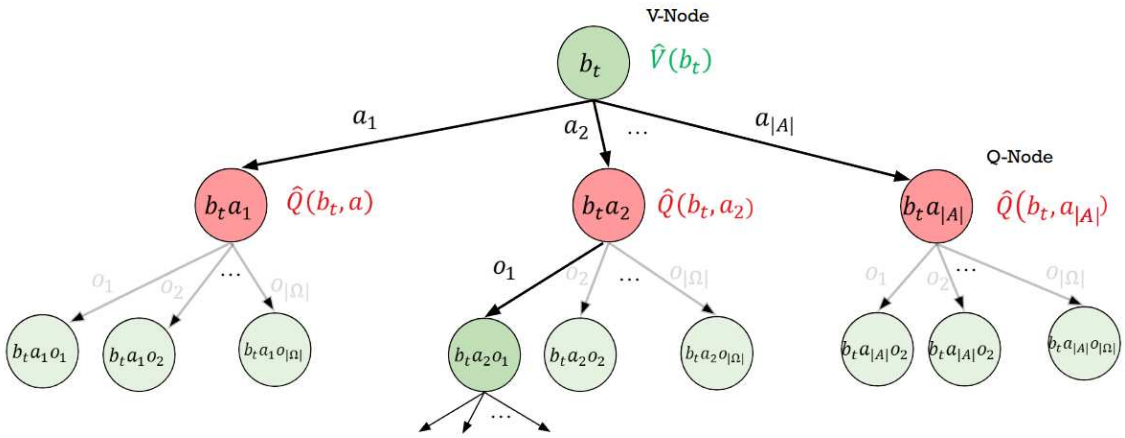


Figure 2.3: Computing the exact value over the full belief tree (black + gray) is infeasible for practical domains. Instead, approximating the value based on a subtree of the full belief tree (black) is an intuitive and promising approach. This is the idea behind sparse tree search-based online POMDP planning algorithms.

at belief state b_t over all possible future belief trajectories. These trajectories form a *belief tree*, illustrated in Figure 2.3. This tree can get wide and deep for large domains with long horizons. Computing Q-values exactly becomes intractable as a result. One idea to work around this is to approximate the full tree with a subtree, which represents an estimation of the Q-values. This is a general and intuitive idea, and it is exactly what sparse tree search algorithms do: A set of belief-based samples $\{s \sim b_t\}$ individually travels down the subtree to "experience the domain" and expands the subtree in the process. The difference between specific algorithms boils down to two key issues regarding building the subtree: (1) how a sample obtains its "experience" (*i.e.*, an action-observation sequence) and (2) how the rewards observed by a sample are incorporated into the estimation of value. Currently, the two state-of-the-art sparse tree search-based algorithms are POUCT (Silver & Veness, 2010) and DESPOT (Somani et al., 2013; N. Ye et al., 2017). We briefly summarize how each algorithm deals with the two issues above, which reflects their differences.

POUCT (Partially-Observable UCT) is based on Monte Carlo Tree Search (MCTS), which is a general sample-based sequential decision making algorithm with a track record of breaking state-of-the-art results in game playing, most notably the difficult game of Go (Browne et al., 2012; Silver et al., 2017). POUCT is an extension of UCT (Upper Confidence bounds for Trees) (Kocsis & Szepesvári, 2006) to partially observable domains, which is a version of MCTS that uses the UCB1 algorithm (Auer et al., 2002) for action selection. Effectively, POUCT addresses (1) with UCB1 and generator sampling, that is, $a \leftarrow \text{argmax}_a\, Q(b,a) + c\sqrt{\frac{N(b)}{N(ba)}}$, and $(s', z, r) \sim \mathcal{G}(s, a)$; The latter ensures sparse observation branching. It addresses (2) with the update rule $\hat{Q}(b,a) \leftarrow \hat{Q}(b,a) + \frac{R - \hat{Q}(b,a)}{N(ba)}$

that so that $\hat{Q}$ is the average of the observed returns so far.[22] Steps for POUCT are shown in Algorithm 1 (Section 2.2.6). Note that POMCP is a specific version of POUCT with particle-based belief representation. We use POUCT for its generality and our work does not rely on particle beliefs.

DESPOT (Determinized Sparse Partially Observable Trees) is both known as a planning algorithm as well as the name for the belief subtree built by this algorithm. DESPOT addresses issue (1) with action selection $a \leftarrow \text{argmax}_a U(b, a)$ based on an upper bound estimate $U(b, a)$ of the Q-value $Q(b, a)$, and observation selection based on a pre-determined subset of observations; This subset is deterministic under a set of K pre-specified *scenarios* (*i.e.*, random seeds), which leads to sparse observation branching ahead of time, instead of during the tree search as done by POUCT. DESPOT addresses (2) by maintaining an upper and lower bound on $Q(b, a)$; Refer to Somani et al. (2013) for details on the lower bound.

POUCT is asymptotically optimal as the number of samples approaches infinity, while DESPOT outputs a near-optimal policy if K is large enough. Subsequent works have extended both algorithms to handle continuous domains (Sunberg & Kochenderfer, 2018a; Garg et al., 2019). Empirically, one is not shown to be strictly better than the other (Sunberg & Kochenderfer, 2018a), and both are affected by the rollout policy (*a.k.a.*, default policy) used. We build upon POUCT for planning due to its optimality and simplicity of implementation, although our POMDP formulations are general.

22. Here, R is the observed return and $N(b)$ denotes the visitation count to the node for belief b.

2.2.5 Appendix: Derivation of the POMDP Bellman Equation

Below, I provide a derivation of the POMDP Bellman Equation in Equation 2.3. Start by defining $U(b_t)$ to be the utility of some arbitrary action sequence given b_t,

$$U(b_t) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \Big| b_t\right] \tag{2.9}$$

Take out the first term of the summation,

$$= \mathbb{E}\left[r_t + \sum_{k=1}^{\infty} \gamma^k r_{t+k} \Big| b_t\right] \tag{2.10}$$

By linearity of the expectation,

$$= \mathbb{E}\left[r_t | b_t\right] + \mathbb{E}\left[\sum_{k=1}^{\infty} \gamma^k r_{t+k} \Big| b_t\right] \tag{2.11}$$

Pull out γ and rewriting the summation index,

$$= \mathbb{E}\left[r_t | b_t\right] + \gamma\mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \Big| b_t\right] \tag{2.12}$$

Recall the Tower property of conditional expectation: $\mathbb{E}[X|Y] = \mathbb{E}[\mathbb{E}[X|Y,Z]|Y]$. In this case, "$X$" is the summation, "$Y$" is b_t, and "Z" is b_{t+1}. The utility starting at time $t+1$ (summation) is conditionally independent of b_t given b_{t+1}, since the history corresponding to b_{t+1} subsumes b_t. Therefore, we drop b_t in the inner conditional expectation:

$$= \mathbb{E}\left[r_t | b_t\right] + \gamma\mathbb{E}\left[\mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \Big| b_{t+1}\right] \Big| b_t\right] \tag{2.13}$$

Using the definition of U,

$$= \mathbb{E}\left[r_t | b_t\right] + \gamma\mathbb{E}\left[U(b_{t+1}) \big| b_t\right] \tag{2.14}$$

Recall the definition of $V(b_t)$ in Equation 2.2. We have:

$$V(b_t) = \max_{a_t, a_{t+1}, \cdots \in A} U(b_t) \tag{2.15}$$

$$= \max_{a_t, a_{t+1}, \cdots \in A} \left\{ \mathbb{E}\left[r_t | b_t\right] + \gamma \mathbb{E}\left[U(b_{t+1}) \middle| b_t\right] \right\} \tag{2.16}$$

$$= \max_{a_t} \left\{ \mathbb{E}\left[r_t | b_t\right] + \max_{a_{t+1}, a_{t+2} \cdots \in A} \gamma \mathbb{E}\left[U(b_{t+1}) \middle| b_t\right] \right\} \tag{2.17}$$

$$= \max_{a_t} \left\{ \sum_{s \in S} b_t(s) R(s, a_t) + \max_{a_{t+1}, a_{t+2} \cdots \in A} \gamma \sum_{z \in \mathcal{Z}} \Pr(z | b_t, a_t) U(b_{t+1}^{a_t, z}) \right\} \tag{2.18}$$

$$= \max_{a_t} \left\{ \sum_{s \in S} b_t(s) R(s, a_t) + \gamma \sum_{z \in \mathcal{Z}} \Pr(z | b_t, a_t) \max_{a_{t+1}, a_{t+2} \cdots \in A} U(b_{t+1}^{a_t, z}) \right\} \tag{2.19}$$

$$= \max_{a_t} \left\{ \sum_{s \in S} b_t(s) R(s, a_t) + \gamma \sum_{z \in \mathcal{Z}} \Pr(z | b_t, a_t) V(b_{t+1}^{a_t, z}) \right\} \tag{2.20}$$

2.2.6 Appendix: The POUCT Algorithm

In Algorithm 1, we provide the pseudocode for the POUCT (Partially Observable UCT) (Silver & Veness, 2010) algorithm. This is the same algorithm as POMCP as presented in Silver & Veness (2010) without particle belief representation. We slightly modified the notation to match the that of Algorithm 3 (Chapter 4, page 49).

Additional notations:

$\Pr(s\|h)$	belief state corresponding to history h
$\mathcal{G}$	black-box generator
d	maximum tree depth (planning depth)
T	belief tree
V	value estimate
N	visitation count
R	observed discounted return
γ	discount factor
$\pi_{rollout}$	rollout policy
hao	a history from h following a and o

Algorithm 1: Partially Observable UCT $(\mathcal{G}, h, d) \rightarrow a$

procedure Search(h)
  // Entry function of POUCT
  repeat
    $s \sim \Pr(s|h)$; // $\Pr(s|h)$ is the belief state
    Simulate($s, h, 0$);
  until TIMEOUT();
  return $\operatorname{argmax}_a V(ha)$; // $V(ha)$ is the Q-value of action a

procedure Simulate *($s, h, depth$)*
  if $depth > d$ **then**
    return *0*
  end
  if $h \notin T$ **then**
    foreach $a \in \mathcal{A}$ **do**
      $T(ha) \leftarrow (N_{init}(ha), V_{init}(ha))$;
    end
    return Rollout *($s, h, depth$)*
  end
  $a \leftarrow \operatorname{argmax}_a V(ha) + c\sqrt{\frac{\log N(h)}{N(ha)}}$;
  $(s', o, r) \sim \mathcal{G}(s, a)$;
  $R \leftarrow r + \gamma \cdot$ Simulate$(s', hao, depth + 1)$; // R is the discounted return
  $N(h) \leftarrow N(h) + 1$;
  $N(ha) \leftarrow N(ha) + 1$;
  $V(ha) \leftarrow V(ha) + \frac{R - V(ha)}{N(ha)}$;
  return *R*

procedure Rollout *($s, h, depth$)*
  if $depth > H$ **then**
    return *0*
  end
  $a \leftarrow \pi_{rollout}(h, \cdot)$;
  $(s', o, r) \sim \mathcal{G}(s, a)$;
  return $r + \gamma \cdot$ Rollout$(s', hao, depth + 1)$;

CHAPTER 3

Overarching Methodology

AT its core, this thesis argues for modeling object search as a POMDP while exploiting structures for practicality. The idea is for the model to take advantage of structures in human environments and human-robot interaction, while being independent of any specific robot or environment. Then, a system or package that implements a solution strategy to this POMDP can be general across, and thus integrable with, different robots and environments. What does this POMDP look like, and how do we approach "solving" this POMDP? The goal of this chapter is to address these questions.

3.1 A Generic POMDP Model for Object Search

First, what does this POMDP looks like? Here, let us consider the basic problem setting as motivated in Figure 1.2 (Chapter 1, pp. 2-4), where a robot with a movable camera searches for a single static target object. I describe a sufficient and generic formulation of this POMDP as a starting point.[1] This POMDP model can be considered the "parent" of the variants in later chapters, which tackle specific problem settings (*e.g.*, multi-object search, searching in 3D, correlations, spatial language, etc.). This draws parallels with the "parent problem" of object search variants in Background (Section 2.1.1, page 11). I briefly discuss how this model can be extended to address the additional challenges in Chapter 9.

The formulation of this POMDP is as follows:

- **State space $\mathcal{S}$.** A state $s \in \mathcal{S}$, $s = (s_{\text{robot}}, s_{\text{target}})$ is factored into the robot state s_{robot} and the target state s_{target}. The robot state contains the robot pose (position and

1. Refer to Section 2.2 for an introduction of POMDPs.

orientation of the camera). The target state contains the location of the target object.

- **Action space $\mathcal{A}$.** Generically, we consider just two action types LOOK and FIND; The purpose of LOOK is to perceive some part of the search region, and the purpose of FIND is to declare object(s) to be found at some location. When camera is used, LOOK corresponds to changing the position and orientation of the camera, either relative to the current pose or to some goal pose, and then project a field of view to receive observations. Note that these actions are abstract and can be broken down into finer-grained, parameterized action types as needed, as done, for example, in the 3D Multi-Object Search (3D-MOS) model (Section 4.3.3, page 42).

- **Observation space $\mathcal{O}$.** An observation $z = (z_{\text{robot}}, z_{\text{target}})$ is factored into the observation of the robot itself z_{robot} and the observation of the target object z_{target}. For object search, z_{target} is typically the detected location of the object (though in 3D-MOS we consider it to be a set of voxels in the field of view; see Section 4.3.2, page 41) and z_{robot} should be an estimation of the robot state, for example, an estimated robot pose.

- **Transition model $T(s, a, s')$.** When the robot takes a LOOK action, the robot should change its pose to the desired destination with some domain-specific noise. When the robot takes FIND, the target should be marked as found if the condition for success is satisfied, such as when the target object is within the field of view of the robot, which should be determined based on the state s. The target is assumed to be static.

- **Observation model $O(s', a, z)$.** The robot observation is an estimate of the robot state. So z_{robot} is independent of the target observation given the robot state. Therefore, the observation model $O(s', a, z)$ can be factored as $O(s', a, z) = \Pr(z|s', a) = \Pr(z_{\text{robot}}|s'_{\text{robot}}, a) \Pr(z_{\text{target}}|s', a)$. Here, $\Pr(z_{\text{robot}}|s'_{\text{robot}}, a)$ can be regarded as a model of the robot's localization module, and $\Pr(z_{\text{target}}|s', a)$ models the object detection mechanism (*e.g.* through a field of view) and the uncertainty of object detection.

- **Reward function $R(s, a)$.** The reward of a LOOK action should depend on the robot state and the viewpoint changing action, which should reflect the cost needed to complete the LOOK action, such as time or travel distance. Taking FIND signals a commitment by the robot its belief of the object location. If correct, then the robot successfully completes the task, receiving a high reward R_{max}. However, the correctness of declarations can be expensive to verify, for example, by a human teammate. Therefore, taking FIND is also significantly more costly than LOOK actions, receiving a high penalty R_{min}.

3.1.1 Remark

This model might seem simple, but it is not obvious. Since POMDP is such a general model for robot behavior, it may be tempting to pack too much or too little information into the model. The key is to determine the right level of abstraction this POMDP model should live at in a practical robot system.

Our insight is that the perception and action of an object search system should happen at a level higher than the basic building blocks, such as localization, object detection, navigation or low-level control. It should also be at a lower level of abstraction than a high-level task such as *"pick two apples and then heat them"* (example from Figure 1.1a). The model above echos that vision. This somewhat "middle-level" of abstraction of this object search model also coincides with the duality of object search's role in people's mind, as discussed in Chapter 1 (page 2).

3.1.2 Solution Method

This thesis takes the explicit, online planning approach to obtaining policies for POMDPs. Concretely, this means to explicitly maintain the belief state and model the POMDP's components, such as programmatically define observation models based on analytical functions, and then apply general-purpose online POMDP planning algorithms based on Monte Carlo Tree Search to obtain an approximately optimal policy. To improve performance and practicality, I develop and employ techniques such as multi-resolution planning, hierarchical planning, view position graph sampling, and using heuristic rollout policies depending on the specific problem setting, discussed in the following Chapters (Chapters 4-7).

THIS IS THE END OF THIS CHAPTER.

CHAPTER 4

3D Multi-Object Search

4.1 Motivation - Why 3D Object Search?

ROBOTS operating in human spaces must find objects such as glasses, books, or cleaning supplies that could be on the floor, shelves, or tables. This search space is naturally 3D. When multiple objects must be searched for, such as a cup and a mobile phone, an intuitive strategy is to first hypothesize likely search regions for each target object based on semantic knowledge or past experience (Kollar & Roy, 2009; Aydemir et al., 2013), then search carefully within those regions. Since the latter directly determines the success of the search, it is *essential* for the robot to produce an efficient search policy within a designated 3D search region under limited field of view (FOV), where target objects could be partially or completely occluded. In this chapter and next, we consider the problem setting where a robot must search for multiple objects in a 3D search region by actively moving its camera, with as few steps as possible (Figure 8.1).

This chapter begins by articulating both algorithmic and system-level challenges for 3D multi-object search, followed by remark on previous work and a summary of our contributions. From then on, the chapter focuses on the theoretical side of this problem and tackling algorithm-level challenges. The next chapter (Chapter 5) tackles system-level challenges by presenting a system for generalized 3D multi-object search deployed on different robots.

4.1.1 Algorithm-Level Challenges

Searching for objects in a large search region requires acting over long horizons under various sources of uncertainty in a partially observable environment. For this reason, previous

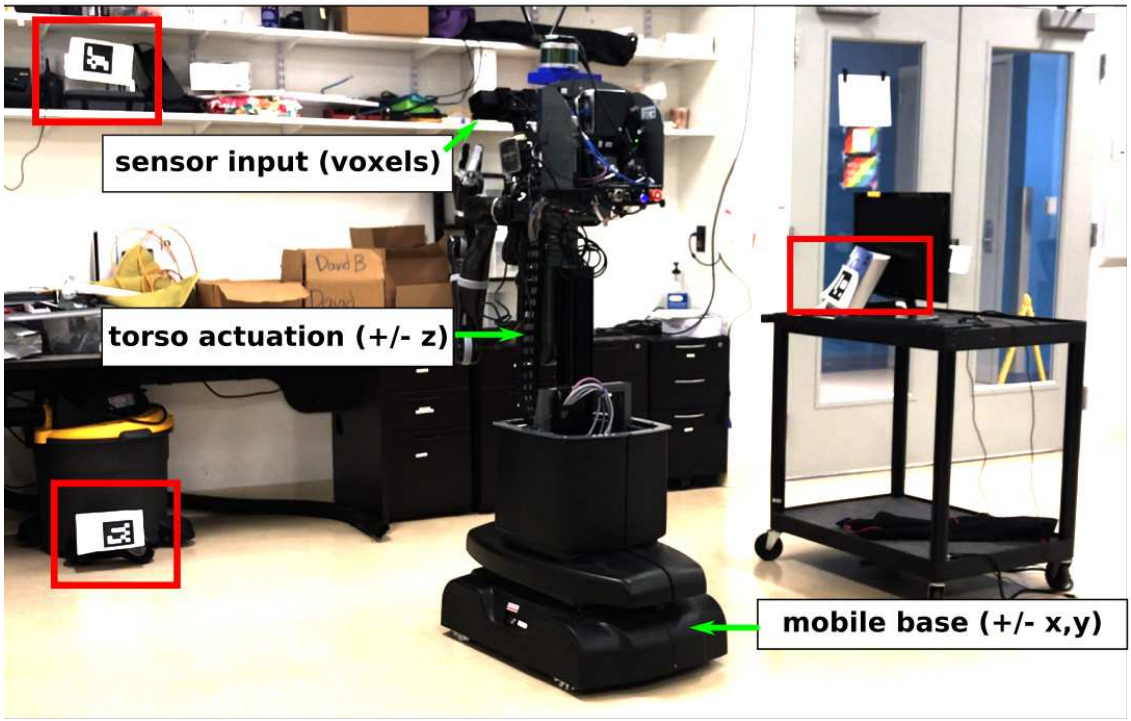


Figure 4.1: An example of the 3D-MOS problem where a torso-actuated mobile robot is tasked to search for three objects placed at different heights in a lab environment. The objects are represented by paper AR tags marked by red boxes. Note that the robot must actively move itself due to limited field of view, and the objects can be occluded by the attached obstacles if viewed from the side.

works have used Partially Observable Markov Decision Process (POMDP) as a principled decision-theoretic framework for object search (Xiao et al., 2019; Atanasov et al., 2014; Danielczuk et al., 2019). However, to ensure the POMDP is manageable to solve, previous works reduce the search space or robot mobility to 2D (Aydemir et al., 2013; Wandzel et al., 2019; J. K. Li et al., 2016), although objects exist in rich 3D environments. The key challenges lie in the intractability of maintaining exact belief due to large state space (Silver & Veness, 2010), and the high branching factor for planning due to large observation space (Sunberg & Kochenderfer, 2018b; Garg et al., 2019).

4.1.2 System-Level Challenges

As such a valuable and fundamental skill for robots, we expect that eventually object search becomes an *off-the-shelf* ability any robot can acquire to search for objects in the environment that it operates in, similar to other capabilities such as object detection, SLAM, and motion planning. However, unlike the other aforementioned robotic capabilities, to the best

of our knowledge, there is no general-purpose object search package available for robotics researchers and practitioners. Sophisticated mobile robot platforms, such as the Kinova MOVO (*Kinova MOVO*, 2017) and the Boston Dynamics Spot (*Boston Dynamics Spot*, 2019), do not come equipped with an object search system, despite their otherwise impressive capabilities. This thesis takes the first step towards a robot-independent, environment-agnostic system and package for generalized object search.

4.1.3 Remark on Previous Work

Searching for a single, static object in 3D by planning sensing parameters (*e.g.*position, orientation, and zoom of a camera) under a time budget is NP-complete (Y. Ye & Tsotsos, 1997).[1] Previous work primarily address the computational complexity of object search by hypothesizing likely regions based on object co-occurrence (Kollar & Roy, 2009; Wixson & Ballard, 1994), semantic knowledge (Aydemir et al., 2013) or language (Wandzel et al., 2019), reducing the state space from 3D to 2D (Wandzel et al., 2019; C. Wang et al., 2018; Sarmiento et al., 2003; Nie et al., 2016), or constrain the sensor to be stationary (Danielczuk et al., 2019; Dogar et al., 2014). The work in this chapter focuses on multi-object search within a 3D region where the robot actively moves the mounted camera, for example, through pan or tilt, or by moving itself.

Several works explicitly reason over the arrangement of occluded objects based on given geometry models of clutter (Xiao et al., 2019; Nie et al., 2016; Wong et al., 2013). Our approach considers occlusion as part of the observation that contains no information about target locations and we do not require geometry models.

Many works formulate object search as a POMDP. Notably, Aydemir et al. (2013) first infer a room to search in then perform search by calculating candidate viewpoints in a 2D plane. J. K. Li et al. (2016) plan sensor movements online, yet assume objects are placed at the same surface level in a container with partial occlusion. Xiao et al. (2019) address object fetching on a cluttered tabletop where the robot's FOV fully covers the scene, and that occluding obstacles are removed permanently during search. Wandzel et al. (2019) formulate the multi-object search (MOS) task on a 2D map using the proposed Object-Oriented POMDP (OO-POMDP). We extend that work to 3D and tackle additional challenges by proposing a new observation model and belief representation, and a multi-resolution planning algorithm. In addition, our POMDP formulation allows fully occluded objects and can be in principle applied on different robots such as mobile robots or drones.

Beginning with CARMEN (Montemerlo et al., 2003), open source libraries for SLAM have greatly lowered the barrier to entry into robotics (Grisetti et al., 2007; Hess et al., 2016). Similarly, for motion planning, libraries such as OMPL (Şucan et al., 2012) and MoveIt! (Chitta, 2016) have broadened access to motion planning to a variety of different robotic platforms. Our work aims to do the same thing for object search.

1. See Section 2.1.4 page 16 for elaboration.

Wixson & Ballard (1994) remarked that selecting views for object search in a local region is a harder problem than the selection of which region to search in. Most works that demonstrate real-world robotic search are constrained within a 2D search region or reduce some aspect of the problem (*e.g.*, the observation or action space) (Aydemir et al., 2013; Wandzel et al., 2019; J. K. Li et al., 2016; Zeng et al., 2020; Bejjani et al., 2021; Holzherr et al., 2021; Giuliari et al., 2021; Schmalstieg et al., 2022).

Deep learning methods that typically map raw observations to actions (Yang et al., 2019; Chaplot et al., 2020; Mayo et al., 2021; Deitke et al., 2022; Schmalstieg et al., 2022) can enable 3D object search, yet it is hard to train such a model on a new robot and ensure generalization to a new real-world environment; ongoing work (*e.g.*, by Deitke et al. (2020); Schmalstieg et al. (2022); Gervet et al. (2022)) is addressing this challenge through sim-to-real transfer. In contrast, our approach only requires basic perception capabilities such as object detection and localization to enable object search; point cloud observations can be optionally considered by our system to be occlusion-aware.

4.2 Contributions

The contributions of our work are as follows:

- We introduce **3D Multi-Object Search (3D-MOS)**, a general POMDP formulation for the multi-object search task with 3D state and action spaces, and a realistic observation space in the form of labeled voxels within the viewing frustum from a mounted camera. Following the Object-Oriented POMDP (OO-POMDP) framework proposed by Wandzel et al. (2019), the state, observation spaces are factored by independent objects, allowing the belief space to scale linearly instead of exponentially in the number of objects.
- We address the algorithmic challenges of computational complexity in solving 3D-MOS by developing several techniques that converge to an online multi-resolution planning algorithm:
    - *First,* we propose a per-voxel observation model which drastically reduces the size of the observation space necessary for planning.
    - *Next,* we present a novel belief representation, called **octree belief**, that captures beliefs at different resolutions and allows efficient and exact belief updates.
    - *Then,* we exploit the octree structure and derive abstractions of the ground problem at different resolution levels leveraging abstraction theory for MDPs (L. Li et al., 2006; Bai et al., 2016).
    - *Finally,* a Monte-Carlo Tree Search (MCTS) based online planning algorithm, called Partially-Observable Upper Confidence bounds for Trees (POUCT) (Silver & Veness, 2010), is employed to solve these abstract instances in parallel, and the action with highest value in its MCTS tree is selected for execution.

We evaluate the proposed algorithm in a simulated, discretized 3D domain where a robot with a 6 DOF camera searches for objects of different shapes and sizes randomly generated and placed in a grid environment. In addition, we demonstrate our approach as a proof-of-concept system on a torso-actuated mobile robot in our lab.

- To address system-level challenges, we present **GenMOS (Generalized Multi-Object Search)**, a general-purpose object search system that is robot-independent and environment-agnostic. GenMOS takes as input point cloud observations of the local region (when available), 3D object detection bounding boxes (if detection occurs), and localization of robot camera pose, and outputs a viewpoint to move to as the result of sequential online planning. The point cloud observations are used in three ways: (1) simulate occlusion; (2) inform occupancy and initialize octree belief; and (3) sample a belief-based graph of view positions that avoids obstacles.
    - I implemented this system as a software package based on gRPC (*gPRC Documentation*, n.d.); Besides evaluating it in simulation, I deployed it on the Boston Dynamics Spot robot, the Kinova MOVO robot, and the Universal Robotics UR5e robotic arm, performing object search in different environments.

4.3 Formulation of the 3D-MOS POMDP

The robot is tasked to search for n static target objects (e.g. cup and book) of known type but unknown location in a search space that also contains static non-target obstacles. We assume the robot has access to detectors for the objects that it is searching for. The search region is a 3D grid map environment denoted by G. Let $g \in G \subseteq R^3$ be a 3D grid cell in the environment. We use G^l to denote a grid at *resolution level* $l \in \mathbb{N}$, and $g^l \in G^l$ to denote a grid cell at this level. When l is omitted, it is assumed that g is at the ground resolution level. We introduce the 3D-MOS domain as an OO-POMDP as follows. See Figure 4.2 for illustrations. This model is a multi-object extension and a 3D specialization of the overarching object search POMDP described in Chapter 3.

4.3.1 State space $\mathcal{S}$

An environment state $s = \{s_1, \cdots, s_n, s_r\}$ is factored in an object-oriented way, where $s_r \in \mathcal{S}_r$ is the state of the robot, and $s_i \in \mathcal{S}_i$ is the state of target object i. A robot state is defined as $s_r = (p, \mathcal{F}) \in \mathcal{S}_r$ where p is the 6D camera pose and $\mathcal{F}$ is the set of found objects. The robot state is assumed to be observable to the robot. In this work, we consider the object state to be specified by one attribute, the 3D object pose at its center of mass, corresponding to a cell in grid G. We denote a state $s_i^l \in \mathcal{S}_i^l$ to be an object state at resolution level l, where $\mathcal{S}_i^l = G^l$.

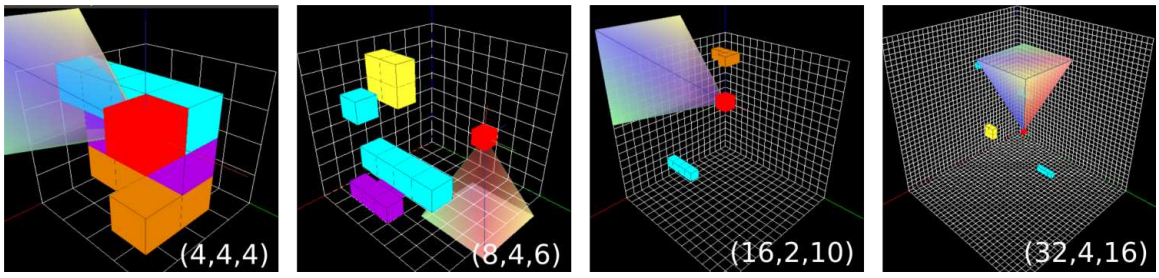


Figure 4.2: Example illustrations of the 3D-MOS POMDP model. The robot (represented as a red cube) can project a viewing frustum to observe the search space, in which objects are represented by sets of cubes. In these examples, the tuple (m, n, d) at lower-right of each image means that the search space in total has $m \times m \times m$ grid cells, with n randomly placed objects, and the robot can project a 45-degree frustum with a far plane at distance d grid cells to the robot. The percentage of search space covered by each viewing frustum, parameterized by field-of-view depth d, decreases as the world size increases.

4.3.2 Observation space $\mathcal{O}$

The robot receives an observation through a viewing frustum projected from a mounted camera. The viewing frustum forms the FOV of the robot, denoted by V, which consists of $|V|$ voxels. Note that the resolution of a voxel should be no lower than that of a 3D grid cell g. We assume both resolutions to be the same in this chapter for notational convenience, hence $V \subseteq G$, but in general a voxel with higher resolution can be easily mapped to a corresponding grid cell.

For each voxel $v \in V$, a *detection function* $d(v)$ labels the voxel to be either an object $i \in \{1, \cdots, n\}$, FREE, or UNKNOWN (Figure 4.3). FREE denotes that the voxel is a free space or an obstacle. We include the label UNKNOWN to take into account occlusion incurred by target objects or static obstacles. In this case, the corresponding voxel in V does not give any information about the environment. An observation $o = \{(v, d(v)) | v \in V\}$ is defined as a set of voxel-label tuples. This can be thought of as the result of voxelization and object segmentation given a raw point cloud.

We can factor o by objects in the following way. First, given the robot state s_r at which o is received, the voxels in V have known locations. Under this condition, V can be reduced to exclude voxels labeled UNKNOWN while still maintaining the same information. Then, V can be decomposed by objects into $V_1, \cdots, V_n$ where for any $v \in V_i$, $d(v) \in \{i, \text{FREE}\}$ which retain the same information as V for a given robot state.[2] Hence, the observation $o = \bigcup_{i=1}^{n} o_i$ where $o_i = \{(v, d(v)) | v \in V_i\}$.

2. The FOV V is fixed for a given camera pose in the robot state, therefore excluding UNKNOWN voxels does not lose information.

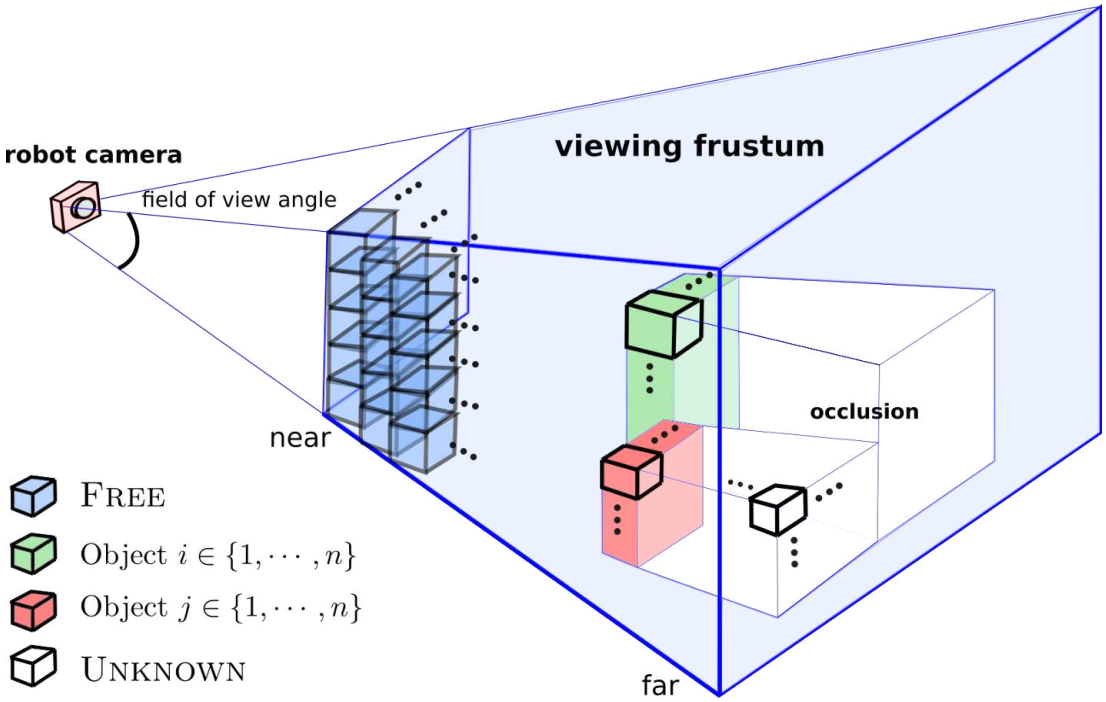


Figure 4.3: Illustration of the viewing frustum and volumetric observation. The viewing frustum V consists of $|V|$ voxels, where each $v \in V$ can be labeled as $i \in \{1, \cdots, n\}$, FREE or UNKNOWN.

4.3.3 Action space $\mathcal{A}$.

Searching for objects generally requires three basic capabilities: *moving*, *looking*, and *declaring* an object to be found at some location. Formally, the action space consists of these three types of primitive actions: MOVE(s_r, g) action moves the robot from pose in s_r to destination $g \in G$ stochastically. LOOK(θ) changes the camera pose to look in the direction specified by $\theta \in \mathbb{R}^3$, and projects a viewing frustum V. FIND(i, g) declares object i to be found at location g. The implementation of these actions may vary depending on the type of search space or robot. Note that this formulation allows macro actions, such as "look after move" to be composed for planning.

4.3.4 Transition function T.

Target objects and obstacles are static objects, thus $\Pr(s'_i|s, a) = \mathbf{1}(s'_i = s_i)$. For the robot, the actions MOVE(s_r, g) and LOOK(θ) change the camera location and direction to g and θ following a domain-specific stochastic dynamics function. The action FIND(i, g) adds i to the set of *found objects* in the robot state only if g is within the FOV determined by s_r.

4.3.5 Reward function R.

The correctness of declarations can only be determined by, for example, a human who has knowledge about the target object or additional interactions with the object; therefore, we consider declarations to be expensive. The robot receives $R_{\text{max}} \gg 0$ if an object is correctly identified by a FIND action, otherwise the FIND action incurs a $R_{\text{min}} \ll 0$ penalty. MOVE and LOOK receive a negative step cost $R_{\text{step}} < 0$ dependent on the robot state and the action itself. This is a sparse reward function.

4.3.6 Observation Model O

We have previously described how a volumetric observation o can be factored by objects into $o_1, \cdots, o_n$. Here, we describe a method to model $\Pr(o_i|s', a)$, the probabilistic distribution over an observation o_i for object i.

Modeling a distribution over a 3D volume is a challenging problem (Park et al., 2019). To develop an efficient model, we make the simplifying assumption that object i is contained within a single voxel located at the grid cell $g = s'_i$. We address the case of searching for objects of unknown sizes with our planning algorithm (Section 4.5) that plans at multiple resolutions in parallel.

Under this assumption, $d(v) = \text{FREE}$ deterministically for $v \neq s'_i$, and the uncertainty of o_i is reduced to the uncertainty of $d(s'_i)$. As a result, $\Pr(o_i|s', a)$ can be simplified to $\Pr(d(s_i)|s', a)$. When $s'_i \notin V_i$, either $d(s'_i) = \text{UNKNOWN}$ (occlusion) or $s'_i \notin V$ (not in FOV). In this case, there is no information regarding the value of $d(s'_i)$ in the observation o_i, therefore $\Pr(d(s'_i)|s', a)$ is a uniform distribution. When $s'_i \in V_i$, that is, the non-occluded region within the FOV covers s'_i, the case of $d(s'_i) = i$ indicates correct detection while $d(s'_i) = \text{FREE}$ indicates sensing error. We let $\Pr(d(s'_i) = i|s', a) = \alpha$ and $\Pr(d(s'_i) = \text{FREE}|s', a) = \beta$. It should be noted that α and β do not necessarily sum to one because the belief update equation (Equation 2.5) does not require the observation model to be normalized, as explained in Section 2.2.2. Thus, hyperparameters α and β independently control the reliability of the observation model.

4.4 Octree Belief Representation

Particle belief representation (Silver & Veness, 2010; Somani et al., 2013) suffers from particle depletion under large observation spaces. Moreover, if the resolution of G is dense, it may be possible that most of 3D grid cells do not contribute to the behavior of the robot.

We represent a belief state $b^i_t(s_i)$ for object i as an octree, referred to as an *octree belief*. It can be constructed incrementally as observations are received and it tracks the belief of object state at different resolution levels. Furthermore, it allows efficient belief sampling and belief update using the per-voxel observation model (Sec. 4.3.6).

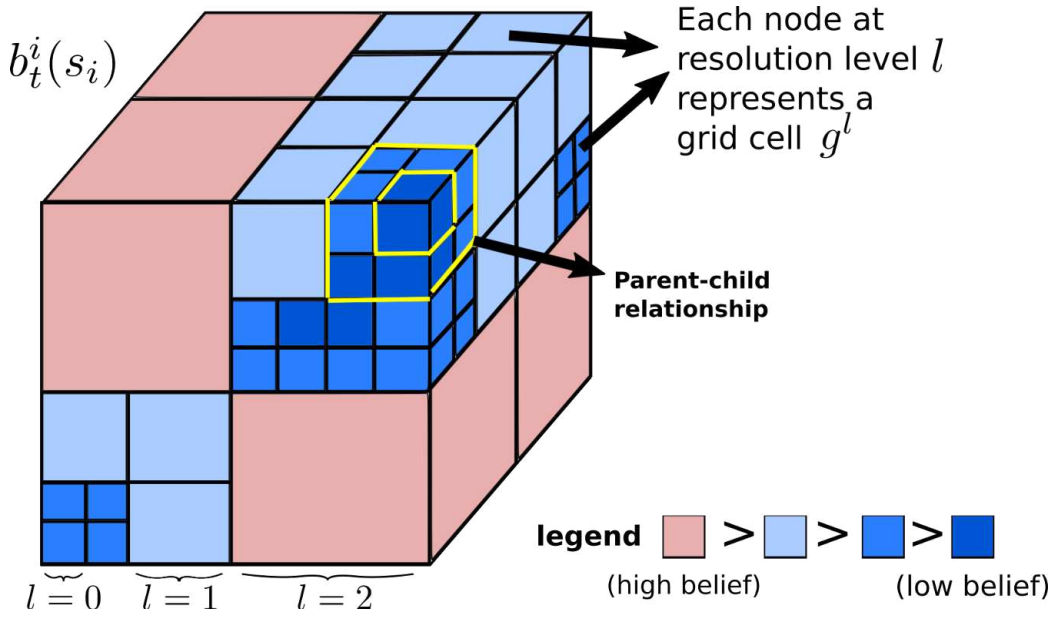


Figure 4.4: Illustration the octree belief representation $b_t^i(s_i)$. The color on a node g^l indicates the belief $\textsc{Val}_t^i(g^l)$ that the object is located within g^l. The highlighted grid cells indicate parent-child relationship between a grid cell at resolution level $l = 1$ (parent) and one at level $l = 0$.

An octree belief consists of an octree and a normalizer. An octree is a tree where every node has 8 children. In our context, a node represents a grid cell $g^l \in G^l$, where l is the resolution level, such that g^l covers a cubic volume of $(2^l)^3$ ground-level grid cells; the ground resolution level is given by $l = 0$. The 8 children of the node equally subdivide the volume at g^l into smaller volumes at resolution level $l - 1$ (Figure 4.4). Each node stores a value $\textsc{Val}_t^i(g^l) \in \mathbb{R}$, which represents the unnormalized belief that $s_i^l = g^l$, that is, object i is located at grid cell g^l. We denote the set of nodes at resolution level $k < l$ that reside in a subtree rooted at g^l by $\textsc{Ch}^k(g^l)$. By definition, $b_t^i(g^l) = \Pr(g^l|h_t) = \sum_{c\in\textsc{Ch}^k(g^l)} \Pr(c|h_t)$. Thus, with a normalizer $\textsc{Norm}_t = \sum_{g\in G} \textsc{Val}_t^i(g)$, we can rewrite the normalized belief as:

$$b_t^i(g^l) = \frac{\textsc{Val}_t^i(g^l)}{\textsc{Norm}_t} = \sum_{c\in\textsc{Ch}^k(g^l)} \left(\frac{\textsc{Val}_t^i(c)}{\textsc{Norm}_t}\right), \tag{4.1}$$

which means $\textsc{Val}_t^i(g^l) = \sum_{c\in\textsc{Ch}^k(g^l)} \textsc{Val}_t^i(c)$. In words, the value stored in a node is the sum of values stored in its children. The normalizer equals to the sum of values stored in the nodes at the ground resolution level.

The octree does not need to be constructed fully in order to query the probability at any grid cell. This can be achieved by setting a default value $\textsc{Val}_0^i(g) = 1$ for all ground grid cells $g \in G$ not yet present in the octree. Then, any node corresponding to g^l has a default value of $\textsc{Val}_0^i(g^l) = \sum_{c\in\textsc{Ch}^0(g^l)} \textsc{Val}_0^i(c) = |\textsc{Ch}^0(g^l)|$.

For clarity (*e.g.*, in Section 5.1.2), we provide the definition of *default value* and *initial value* in an octree belief node as follows:

Definition 1 (Default value). *The* default value *of an octree belief node* $\text{VAL}_0^i(g^l)$ *is the value* before *the node is present in the octree.*

Definition 2 (Initial value). *The* initial value *of an octree belief node* $\text{VAL}_1^i(g^l)$ *is the value* when *the node is inserted into the octree.*

Note that the notation of initial value is consistent with that of initial belief as introduced in Section 2.2.1, page 25.

4.4.1 Belief Update

We have defined a per-voxel observation model for $\Pr(o_i|s', a)$ that reduces to $\Pr(d(s'_i)|s', a)$ if $s'_i \in V_i$, or a uniform distribution if $s'_i \notin V_i$. This suggests that the belief update need only happen for voxels that are inside the FOV to reflect the information in the observation.

Upon receiving observation o_i within the FOV V_i, belief is updated according to Algorithm 2. This algorithm updates the value of the ground-level node g corresponding to each voxel $v \in V_i$ as $\text{VAL}^i_{t+1}(g) = \Pr(d(v)|s', a)\text{VAL}^i_t(g)$. The normalizer is updated to make sure b^i_{t+1} is normalized

Lemma 1. *The normalizer* NORM_t *at time* t *can be correctly updated by adding the incremental update of values as in Algorithm 2.*

Proof. The normalizer must be equal to the sum of node values at the ground level for the next belief b^i_{t+1} to be valid (Equation 4.1). That is, $\text{NORM}_{t+1} = \sum_{s_i \in G} \text{VAL}^i_{t+1}(s_i)$. This sum can be decomposed into two cases where the object i is inside of V_i and outside of V_i; For object locations $s_i \notin V_i$, the *unnormalized* observation model is uniform, thus $\text{VAL}^i_{t+1}(s_i) = \Pr(d(s_i)|s', a)\text{VAL}^i_t(s_i) = \text{VAL}^i_t(s_i)$. Therefore, $\text{NORM}_{t+1} = \sum_{s_i \in V_i} \text{VAL}^i_{t+1}(s_i) + \sum_{s_i \notin V_i} \text{VAL}^i_t(s_i)$. Note the set $\{s_i | s_i \notin V_i\}$ is equivalent as $\{s_i | s_i \in G \setminus V_i\}$. Using this fact and the definition of NORM_t, we obtain $\text{NORM}_{t+1} = \text{NORM}_t + \sum_{s_i \in V_i} \left(\text{VAL}^i_{t+1}(s_i) - \text{VAL}^i_t(s_i)\right)$ which proves the lemma. □

This belief update is therefore exact since the objects are static. The complexity of this algorithm is $O(|V| \log(|G|)$; Inserting nodes and updating values of nodes can be done by traversing the tree depth-wise.

4.4.2 Sampling

Octree belief affords exact belief sampling at any resolution level in logarithmic time complexity with respect to the size of the search space $|G|$, despite not being completely built.

Algorithm 2: OctreeBeliefUpdate $(b_t^i, a, o_i) \rightarrow b_{t+1}^i$

input : b_t^i: octree belief for object i; a: action taken by robot;
$o_i = \{(v, d(v)|v \in V_i\}$: factored observation for object i
output: b_{t+1}^i: updated octree belief
// Let $\Psi(b_i^t)$ denote the octree underlying b_t^i.
for $v \in V_i$ **do**
  $s_i \leftarrow v$; // State at grid cell corresponding to voxel v
  if $s_i \notin \Psi(b_i^t)$ **then**
    Insert node at s_i to $\Psi(b_i^t)$;
  end
  $\text{VAL}_{t+1}^i(s_i) \leftarrow \Pr(d(v)|s', a)\text{VAL}_t^i(s_i)$;
  $\text{NORM}_{t+1} \leftarrow \text{NORM}_t + \text{VAL}_{t+1}^i(s_i) - \text{VAL}_t^i(s_i)$;
end

Given resolution level l, we sample from $\mathcal{S}_i^l$ by traversing the octree in a depth-first manner. Let l_{max} denote the maximum resolution level for the search space. Let l_{des} be the *desired* resolution level at which an object state is sampled.[3] If $s_i^{l_{des}}$ is sampled, then all nodes in the octree that cover $s_i^{l_{des}}$, i.e, $s_i^{l_{max}}, \cdots, s_i^{l_{des}+2}, s_i^{l_{des}+1}$, must also be implicitly sampled, Also, the event that s_i^{l+k} is sampled is independent from other samples given that s_i^{l+k+1} is sampled. Hence,

$$\begin{aligned} &\Pr(s_i^{l_{des}}|h_t) \\ &= \Pr(s_i^{l_{max}}, \cdots, s_i^{l_{des}+2}, s_i^{l_{des}+1}, s_i^{l_{des}}|h_t) \qquad (4.2) \\ &= \Pr(s_i^{l_{des}}|s_i^{l_{des}+1}, h_t) \times \Pr(s_i^{l_{des}+1}|s_i^{l_{des}+2}, h_t) \times \cdots \times \Pr(s_i^{l_{max}-1}|s_i^{l_{max}}, h_t) \qquad (4.3) \end{aligned}$$

Therefore, the task of sampling $s^{l_{des}}$ is translated into sampling a sequence of samples $s_i^{l_{max}}, \cdots, s_i^{l_{des}+2}, s_i^{l_{des}+1}, s_i^{l_{des}}$, each according to the distribution $\Pr(s_i^l|s_i^{l+1}, h_t)$. We can show that this distribution can be efficiently obtained using octree belief itself as follows:

$$\begin{aligned} \Pr(s_i^l|s_i^{l+1}, h_t) &= \frac{\Pr(s_i^l, s_i^{l+1}|h_t)}{\Pr(s_i^{l+1}|h_t)} \qquad (4.4) \\ &= \frac{\Pr(s_i^l|h_t)}{\Pr(s_i^{l+1}|h_t)} \qquad (4.5) \\ &= \frac{\text{VAL}_t^i(s_i^l)/\text{NORM}_t}{\text{VAL}_t^i(s_i^{l+1})/\text{NORM}_t} \qquad (4.6) \\ &= \frac{\text{VAL}_t^i(s_i^l)}{\text{VAL}_t^i(s_i^{l+1})} \qquad (4.7) \end{aligned}$$

Step 4.5 holds because sampling both s_i^l and s_i^{l+1} is equivalent to sampling just s_i^l since the latter (the event that s_i^{l+1} is sampled) is deterministic when the former (the event that s_i^l is

3. Recall that sampling a object state s_i^l here means that the object is considered to be located at s_i^l.

sampled) happens. Sampling from this probability distribution is efficient, as the sample space, i.e. the children of node s_i^{l+1} is only of size 8. Therefore, this sampling scheme yields a sample $s^{l_{des}}$ exactly according to $b_t^i(s^{l_{des}})$ with time complexity $O(\log(|G|))$.

4.5 Using Octree Belief for Multi-Resolution Planning

POUCT expands an MCTS tree using a *generative function* $(s', o, r) \sim \mathcal{G}(s, a)$, which is straightforward to acquire since we explicitly define the 3D-MOS models. However, directly applying POUCT is subject to high branching factor due to the large observation space in our domain.

Our intuition is that octree belief imposes a spatial state abstraction, which can be used to derive an abstraction over observations, reducing the branching factor for planning. Below, we formulate an *abstract 3D-MOS* with smaller spaces, and propose our multi-resolution planning algorithm.

4.5.1 Abstract 3D-MOS

We adopt the abstraction scheme in L. Li et al. (2006) where in general, the abstract transition and reward functions are weighted sums of the original problem's transition and reward functions, respectively with weights that sum up to 1. We define an abstract 3D-MOS $\langle \hat{\mathcal{S}}, \hat{\mathcal{A}}, \hat{\mathcal{O}}, \hat{T}, \hat{O}, R, \gamma, l \rangle$ at resolution level l as follows.

State space $\hat{\mathcal{S}}$. For each object i, an abstraction function $\phi_i : \mathcal{S}_i \to \mathcal{S}_i^l$ transforms the ground-level object state s_i to an abstract object state s_i^l at resolution level l. The abstraction of the full state is $\hat{s} = \phi(s) = \{s_r\} \cup \bigcup_i \phi_i(s_i)$ where the robot state s_r is kept as is. The *inverse image* $\phi_i^{-1}(s_i^l)$ is the set of ground states that correspond to s_i^l under ϕ_i (L. Li et al., 2006).

Action space $\hat{\mathcal{A}}$. Since state abstraction lowers the resolution of the search space, we consider macro move actions that move the robot over longer distance at each planning step. Each macro move action MOVEOP(s_r, g) is an *option* (Sutton et al., 1999) that moves s_r to goal location g using multiple MOVE actions. The primitive LOOK and FIND actions are kept.

Transition function $\hat{T}$. Targets and obstacles are still static, and the robot state still transitions according to the ground-level transition function. However, the transition of the found set from $\mathcal{F}$ to $\mathcal{F}'$ is special since the action FIND(i, g) operates at the ground level while s_i^l has a lower resolution ($l > 0$). Let f_i be the binary state variable that is true if and only if object $i \in \mathcal{F}$. Because the action FIND(i, g) affects f_i based only on whether object i is located at g, and that the problem is no longer Markovian due to state abstraction (Bai

et al., 2016), f_i transitions to f_i' following

$$\Pr(f_i'|f_i, s_i^l, h_t, \textsc{Find}(i, g)) \tag{4.8}$$

$$= \sum_{s_i \in \phi_i^{-1}(s_i^l)} \Pr(f_i'|s_i, f_i, \textsc{Find}(i, g)) \Pr(s_i|s_i^l, h_t). \tag{4.9}$$

The above is consistent with the abstract transition function in the works (L. Li et al., 2006; Bai et al., 2016) where the first term corresponds to the ground-level deterministic transition function and the second term $\Pr(s_i|s_i^l, h_t)$, stored in the octree belief, is the *weight* that sums up to 1 for all $s_i \in \mathcal{S}_i$.

Observation space $\hat{\mathcal{O}}$ and function $\hat{O}$. For the purpose of planning, we again use the assumption that an object is contained within a single voxel (yet at resolution level l). Then, given state $\hat{s}'$, the abstract observation o_i^l is regarded as a voxel-label pair $(s_i^l, d(s_i^l))$. Since it is computationally expensive to sum out all object states, we approximate the observation model by ignoring objects other than i:

$$\Pr(o_i^l|\hat{s}', a, h_t) = \Pr(d(s_i^l)|\hat{s}', a, h_t) \tag{4.10}$$

$$\approx \Pr(d(s_i^l)|s_i^l, s_r, a, h_t) \tag{4.11}$$

$$= \sum_{s_i \in \phi_i^{-1}(s_i^l)} \Pr(d(s_i^l)|s_i, s_r, a) \Pr(s_i|s_i^l, h_t). \tag{4.12}$$

This resembles the abstract transition function, where $\Pr(d(s_i^l)|s_i, s_r, a)$ is the ground observation function, and $\Pr(s_i|s_i^l, h_t)$ is again the weight.

For practical POMDP planning, it can be inefficient to sample from this abstract observation model if l is large. In our implementation, we approximate this distribution by Monte Carlo sampling (Shapiro, 2003): We sample k ground states from $\phi_i^{-1}(s_i^l)$ according to their weights.[4] Then we set $d(s_i^l) = i$ if the majority of these samples have $d(s_i) = i$, and $d(s_i^l) = \textsc{Free}$ otherwise. A similar approach is used for sampling from the abstract transition model.

Reward function R. The reward function is the same as the one in ground 3D-MOS, since computing the reward only depends on the robot state which is not abstracted and the abstract action space consists of the same primitive actions as 3D-MOS. Therefore, solving an abstract 3D-MOS is solving the same task as the original 3D-MOS.

4.5.2 Multi-Resolution Planning Algorithm

Abstract 3D-MOS is smaller than the original 3D-MOS which may provide benefit in online planning. However, it may be difficult to define a single resolution level, due to the uncertainty of the size or shape of objects, and the unknown distance between the robot and these objects.

4. We tested $k = 10$ and $k = 40$ and observed similar search performance. We used $k = 10$ in our experiments.

Algorithm 3: MR-POUCT $(\mathcal{P}, b_t, d) \rightarrow \hat{a}$

input : $\mathcal{P}$: a set of abstract 3D-MOS instances at different resolution levels; b_t: belief at time t; d: planning depth
output: $\hat{a}$: an action in the action space of some $P_l \in \mathcal{P}$
procedure Plan(b_t)
  foreach $P_l \in \mathcal{P}$ *in parallel* **do**
    // Recall that $P_l = \langle \hat{\mathcal{S}}, \hat{\mathcal{A}}, \hat{\mathcal{O}}, \hat{T}, \hat{O}, R, \gamma, l \rangle$
    $\mathcal{G} \leftarrow$ GenerativeFunction(P_l);
    $Q_P(b_t, \hat{a}) \leftarrow$ POUCT($\mathcal{G}, h_t, d$);
  end
  $\hat{a} \leftarrow \text{argmax}_{\hat{a}}\{Q_P(b_t, \hat{a}) | P \in \mathcal{P}\}$;
  return $\hat{a}$

Therefore, we propose to solve a number of abstract 3D-MOS problems in parallel, and select an action from $\hat{\mathcal{A}}$ with the highest value for execution. The algorithm is formally presented in Algorithm 3. The set of abstract 3D-MOS problems, $\mathcal{P}$, can be defined based on the dimensionality of the search space and the particular object search setting. Then, it is straightforward to define a *generative function* $\mathcal{G}(\hat{s}, \hat{a}) \rightarrow (\hat{s}', \hat{o}, r)$ from an abstract 3D-MOS instance P using its transition, observation and reward functions. POUCT uses $\mathcal{G}$ to build a search tree and plan the next action. Thus, all problems in $\mathcal{P}$ are solved online in parallel, each by a separate POUCT. The final action with the highest value $Q_P(b_t, \hat{a})$ in its respective POUCT search tree is chosen as the output (see (Silver & Veness, 2010) for details on POUCT). We call this algorithm Multi-Resolution POUCT (MR-POUCT).

Next, we describe evaluation for this proposed multi-resolution planning algorithm. We assess the hypothesis that our approach, MR-POUCT, improves the robot's ability to efficiently and successfully find objects especially in large search spaces. We conduct a simulation evaluation (Section 4.5.3) and a study on a real robot (Section 4.5.4).

4.5.3 Evaluation in Simulation

Setup

We implement our approach in a simulated environment designed to reflect the essence of the 3D-MOS domain (Figure 4.2). Each simulated problem instance is defined by a tuple (m, n, d), where the search region G has size $|G| = m^3$ with n randomly generated, randomly placed objects. The on-board camera projects a viewing frustum with 45 degree FOV angle, an 1.0 aspect ratio, a minimum range of 1 grid cell, and a maximum range of d grid cells. Hence, we can increase the difficulty of the problem by increasing m and n, or by reducing the percentage of voxels covered by a viewing frustum through reducing the FOV range d. Occlusion is simulated using perspective projection and treating each grid cell as a point.

There are two primitive MOVE actions per axis (e.g. $+z$, $-z$) that each moves the robot along that axis by one grid cell. There are two LOOK actions per axis, one for each direction. Finally, a FIND action is defined that declares all not-yet-found objects within the viewing frustum as found. Thus, the total number of primitive actions is 13. MOVE and LOOK actions have a step cost of -1. A successful FIND receives +1000 while a failed attempt receives -1000. A FIND action is successful if part of a new object lies within the viewing frustum. If multiple new objects are present within one viewing frustum when the FIND is taken, only the maximum reward of $+1000$ is received. The task terminates either when the total planning time limit is reached or n FIND actions are taken.

Baselines

We compare our approach (*MR-POUCT*) with the following baselines: *POUCT* uses the octree belief but solves the ground POMDP directly using the original POUCT algorithm. *Options+POUCT* uses the octree belief and a resolution hierarchy, but only the motion action abstraction (i.e. MOVEOP options) is used, meaning that the agent can move for longer distances per planning step but do not make use of state and observation abstractions. *POMCP* uses a particle belief representation which is subject to particle deprivation. Uniform random rollout policy is used for all POMDP-based methods. *Exhaustive* uses a hand-coded exhaustive policy, where the agent traverses every location in the search environment. At every location, the agent takes a sequence of LOOK actions, one in each direction. Finally, *Random* executes actions at uniformly at random.

Each algorithm begins with uniform prior and is allowed a maximum of 3.0s for planning each step. The total amount of allowed planning time plus time spent on belief update is 120s, 240s, 360s, and 480s for environment sizes (m) of 4, 8, 16, or 32, respectively. Belief update is not necessary for *Exhaustive* and *Random*. The maximum number of planning steps is 500. The discount factor γ is set to 0.99. For each (m, n, d) setting, 40 trials (with random world generation) are conducted.

Results

We evaluate the scalability of our approach with 4 different settings of search space size $m \in \{4, 8, 16, 32\}$ and 3 settings of number of objects $n \in \{2, 4, 6\}$, resulting in 12 combinations. The FOV range d is chosen such that the percentage of the grids covered by one projection of the viewing frustum decreases as the world size m increases.[5] The sensor is assumed to be near-perfect, with $\alpha = 10^5$ and $\beta = 0$. We measure the discounted cumulative reward, which reflects both the search efficiency and effectiveness, as well as the number of objects found per trial.

Results are shown in Figure 4.5. Particle deprivation happens quickly due to large observation space, and the behavior degenerates to a random agent, causing POMCP to

5. The maximum FOV coverage for $m = 4, 8, 16$, and 32 is $17.2\%(d = 4), 8.8\%(d = 6), 4.7\%(d = 10)$, and $2.6\%(d = 16)$, respectively.

perform poorly. In small-scale domains, the *Exhaustive* approach works well, outperforming the POMDP-based methods. We find that in those environments, the FOV can capture a significant portion of the environment, making exhaustive search desirable. The POMDP-based approaches are competitive or better in the two largest search environments ($m = 16$ and $m = 32$). In particular, MR-POUCT outperforms *Exhaustive* in all test cases in the larger environments, with greater margin in discounted cumulative reward; *Exhaustive* takes more search steps but is less efficient. When the search space contains fewer objects, MR-POUCT and POUCT show more resilience than Options+POUCT, with MR-POUCT performing consistently better. This demonstrates the benefit of planning with the resolution hierarchy in octree belief especially in large search environments.

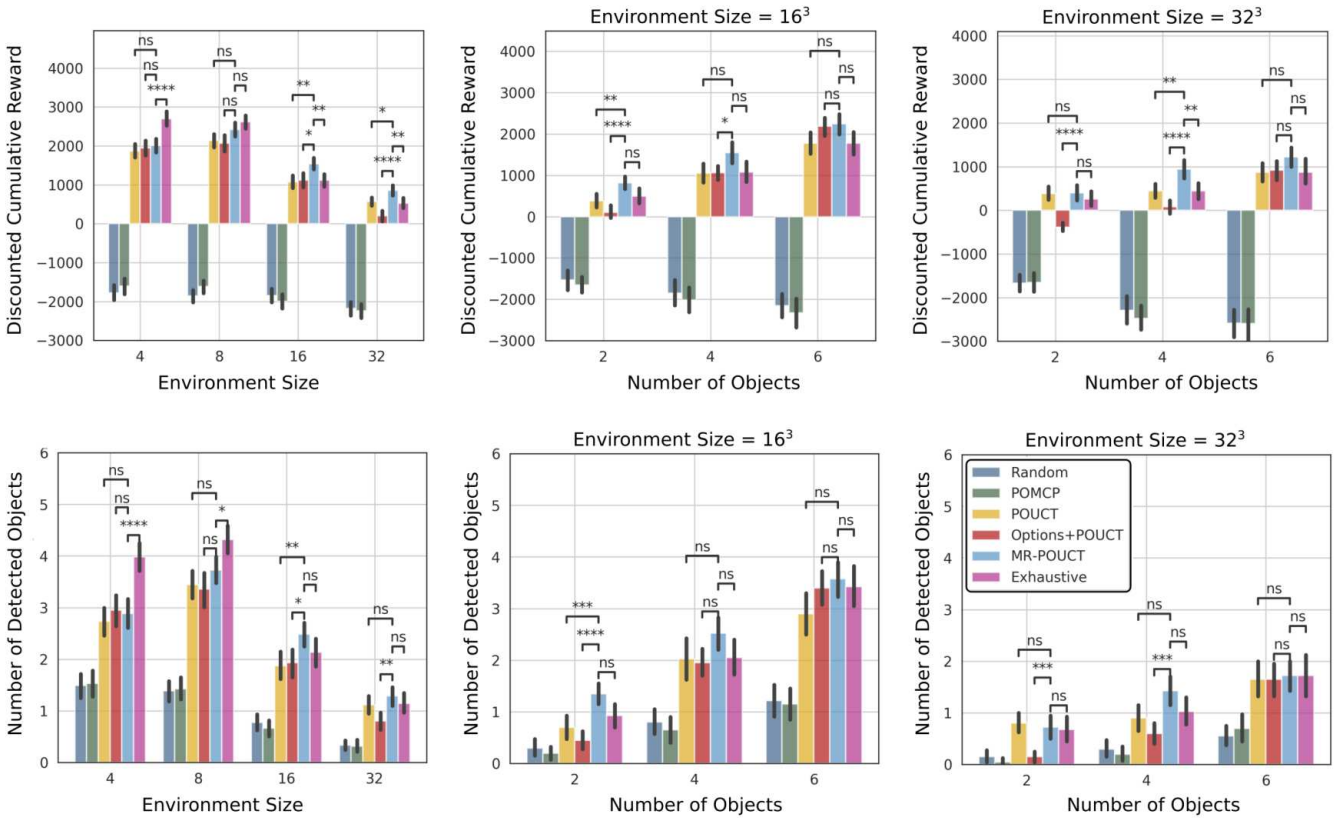


Figure 4.5: Discounted cumulative reward and number of detected objects as the environment size (m) increases and as the of number of objects (n) increases. Exhaustive search performs well in small-scale environments (4 and 8) where exploration strategy is not taken advantage of. In large environments, our method MR-POUCT performs better than the baselines in most cases. The error bars are 95% confidence intervals. The level of statistical significance is shown, comparing MR-POUCT against POUCT, Options+POUCT, and Exhaustive, respectively, indicated by ns ($p > 0.05$), * ($p \leq 0.05$), ** ($p \leq 0.01$), *** ($p \leq 0.001$), **** ($p \leq 0.0001$).

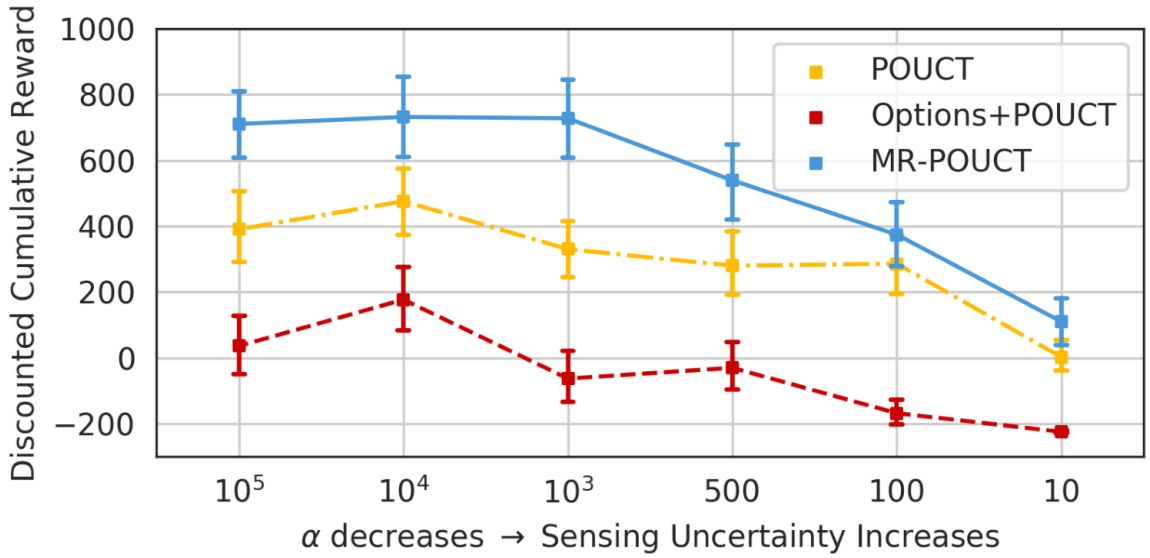


Figure 4.6: Discounted cumulative reward with 95% confidence interval as the sensing uncertainty increases, aggregating over the β settings.

We then investigate the performance of our method with respect to changes in sensing uncertainty, controlled by the parameters α and β of the observation model. According to the belief update algorithm in Section 4.4.1, a noisy but functional sensor should increase the belief $\text{VAL}_t^i(g)$ for object i if an observed voxel at g is labeled i, while decrease the belief if labeled FREE. This implies that a properly working sensor should satisfy $\alpha > 1$ and $\beta < 1$. We investigate on 5 settings of $\alpha \in \{10, 100, 500, 10^3, 10^4, 10^5\}$ and 2 settings of $\beta \in \{0.3, 0.8\}$. A fixed problem difficulty of $(16, 2, 10)$ is used to conduct this experiment. Results in Figure 4.6 show that MR-POUCT is consistently better in all parameter settings. We observe that β has almost no impact to any algorithm's performance as long as $\beta < 1$, whereas decreasing α changes the agent behavior such that it must decide to LOOK multiple times before being certain.

4.5.4 Demonstration on Real-Robot

We demonstrate that our approach is scalable to real world settings by implementing the 3D-MOS problem as well as MR-POUCT for a mobile robot setting. We use the Kinova MOVO Mobile Manipulator robot, which has an actuated torso with an extension range between around 0.05m and 0.5m, which facilitates a 3D action space. The robot operates in a lab environment, which is decomposed into two *search regions* G_1 and G_2 of size roughly $10\text{m}^2 \times$ 2m (Figure. 4.7), each with a semantic label ("shelf-area" for G_1 and "whiteboard-area" for G_2). The robot is tasked to look for n_{G_1} and n_{G_2} objects in each search region sequentially, where objects are represented by paper AR tags that could be in clutter or not detectable at an angle. The robot instantiates an instance of the 3D-MOS problem once

it navigates to a search region. In this 3D-MOS implementation, the MOVE actions are implemented based on a topological graph on top of a metric occupancy grid map. The

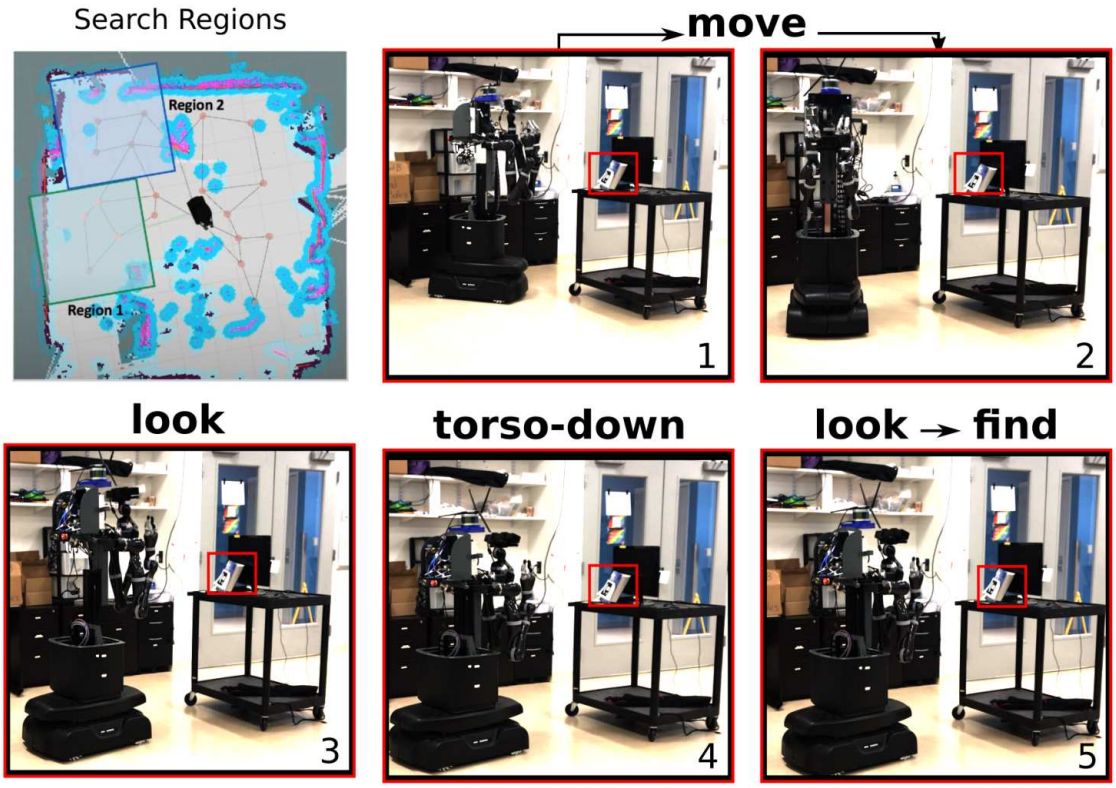


Figure 4.7: Example action sequence produced by the proposed approach that enables a Kinova MOVO robot to perform 3D object search in two search regions separately (top left). The mobile robot first navigates in front of a portable table (1-2). It then takes a LOOK action to observe the space in front (3), and no target is observed since the torso is too high. The robot then decides to lower its torso (4), takes another LOOK action in the same direction, and then FIND to mark the object as found (5). This sequence of actions demonstrate that our algorithm can produce efficient search strategies in real world scenarios.

neighbors of a graph node form the motion action space when the robot is at that node. The robot can take LOOK action in 4 cardinal directions in place and receive volumetric observations; A volumetric observation is a result of downsampling and thresholding points in the corresponding point cloud. The robot was able to find 3 out of 6 total objects in the two search regions in around 15 minutes. One sequence of actions (Figure 4.7) shows that the robot decides to lower its torso in order to LOOK and FIND an object.[6] A failure mode is that the object may not be covered by any viewpoint and thus not detected; this can be

6. Video footage with visualization of volumetric observations and octree belief update is available at https://zkytony.github.io/3D-MOS/.

improved with a denser topological map, or by considering destinations of MOVE actions sampled from the continuous search region.

In the next chapter, we present a system for generalized 3D multi-object search, the first of its kind, and discuss its integration with different robots performing object search in different environments.

THIS IS THE END OF THIS CHAPTER.

CHAPTER 5

GenMOS: A System for Generalized 3D Multi-Object Search

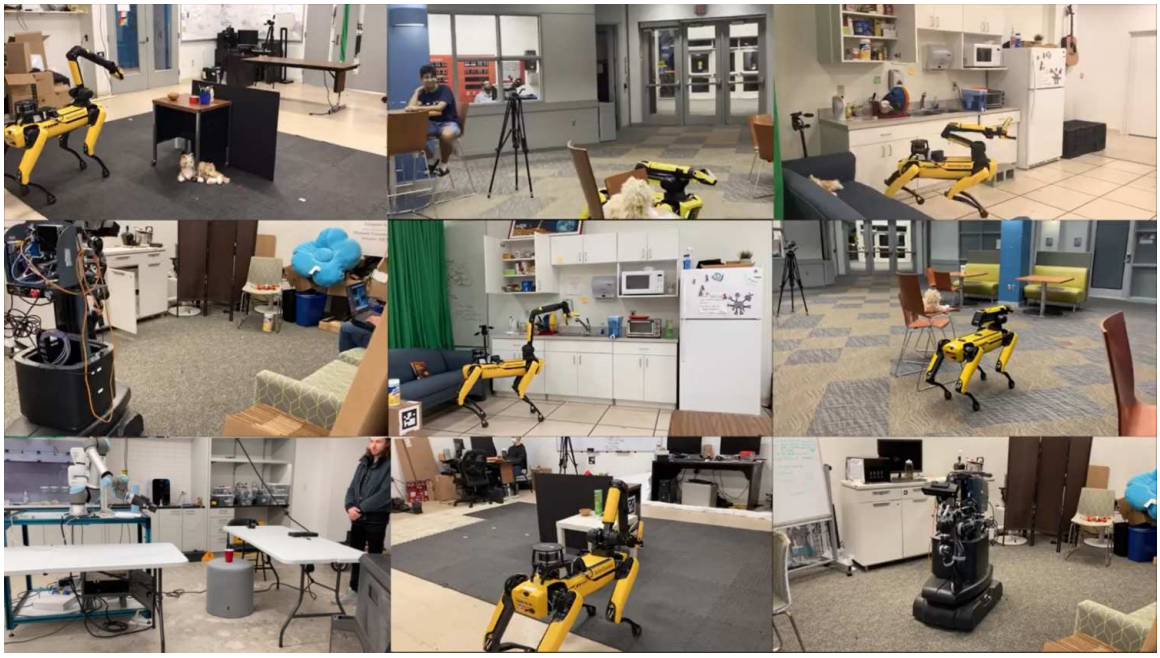


Figure 5.1: GenMOS enables different robots to search for objects in various 3D regions.

5.1 The GenMOS System

TOWARDS the goal of making object search an off-the-shelf capability for any robot, we present GenMOS (Generalized Multi-Object Search), the first general-purpose object

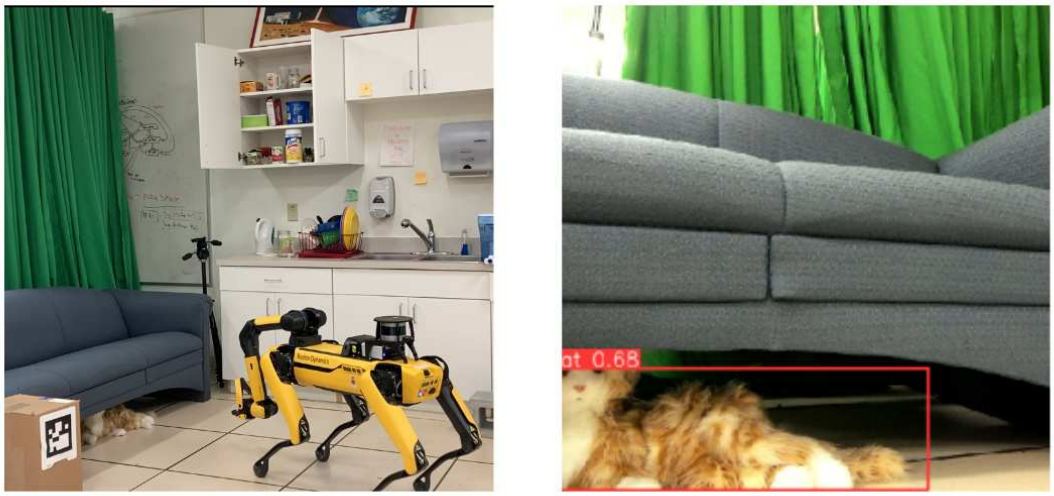


Figure 5.2: Our system, GenMOS, enables a Boston Dynamics Spot robot to successfully find a toy cat underneath the couch. The left image shows a third-person view of the scene. The right image shows the RGB image from the gripper camera, along with the object detection bounding box for the cat labeled.

search system that is robot-independent and environment-agnostic (Figure 5.1). GenMOS builds upon the methodology for 3D multi-object search described in the previous chapter, while significantly improving its practicality in the real world. Our system enables a Boston Dynamics Spot to find, for example, a cat underneath the couch, as shown in Figure 6.1.

This chapter begins with an overview of the system's design, illustrated in Figure 5.3. In particular, we describe three ways that point cloud observations are used in GenMOS:

(1) to simulate occlusion (Figure 5.4);
(2) to inform occupancy and initialize octree belief (Figure 5.5);
(3) to sample a belief-based graph of view positions (Figure 5.9, right column).

Then, we describe novel algorithmic contributions regarding (2) and (3): For (2), we propose an algorithm for initializing octree beliefs given arbitrary prior distributions over object locations; For (3), we propose an algorithm which samples a belief-dependent graph of view positions, allowing the output space of GenMOS to be the continuous space of reachable viewpoints. Subsequently, we describe the gRPC protocol of our implementation of GenMOS as well as a few useful parameters one can configure to adapt GenMOS to a specific scenario. Finally, we discuss our evaluation of GenMOS, first in a simulation domain, then integrated on three robot platforms: Boston Dynamics Spot, Kinova MOVO, and Universal Robotics UR5e.

Contributions. The contributions of this chapter were described in Section 4.2. We emphasize here that the algorithms and evaluation in this chapter serve to improve and demonstrate the practicality of the octree-based 3D multi-object search approach intro-

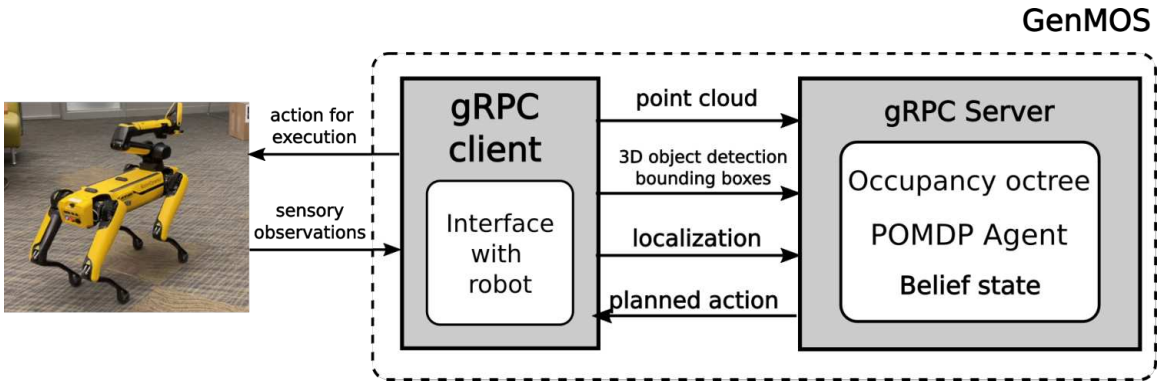


Figure 5.3: Overview of the GenMOS system. See Section 5.1.1 for description.

duced in the previous chapter, which was only evaluated in an idealistic simulation with cardinal action space and on a MOVO with a proof-of-concept system.

5.1.1 System Overview

A gRPC-based system. GenMOS is a client-server construct, designed and implemented based on gRPC (*gPRC Documentation*, n.d.), a high-performance, cross-platform, and open source framework for remote procedural call (RPC). As a gRPC-based system (Figure 5.3), GenMOS is independent of, thus integrable to any particular robot middleware such as ROS (Quigley et al., 2009) or ROS 2 (Macenski et al., 2022).

Inputs and outputs. GenMOS considers perceptual inputs including point cloud observations of the local region, 3D object detection bounding boxes (if detection occurs), and localization of robot camera pose, and it outputs a viewpoint to move to as the result of sequential online planning.

Server

Here, I describe several important aspects of the GenMOS server.

3D-MOS. The server internally maintains a POMDP model of the search task, which is a 3D-MOS with an instantiation of the action space based on a graph of view positions (Section 5.1.3). The definition and implementation of this model, based on `pomdp_py` (Zheng & Tellex, 2020), is general and does not depend on any particular environment. Importantly, the server handles coordinate conversion: the client only needs to send data in the metric world frame and the server properly converts them into the POMDP's frame.

Occupancy octree from point cloud. Internally, the server maintains an octree representation of the search region's occupancy, used to simulate occlusion-enabled observations for belief update. Point cloud observations can be sent to the server to update the server's model of the search region. Specifically, the server converts the point cloud into an *occu-*

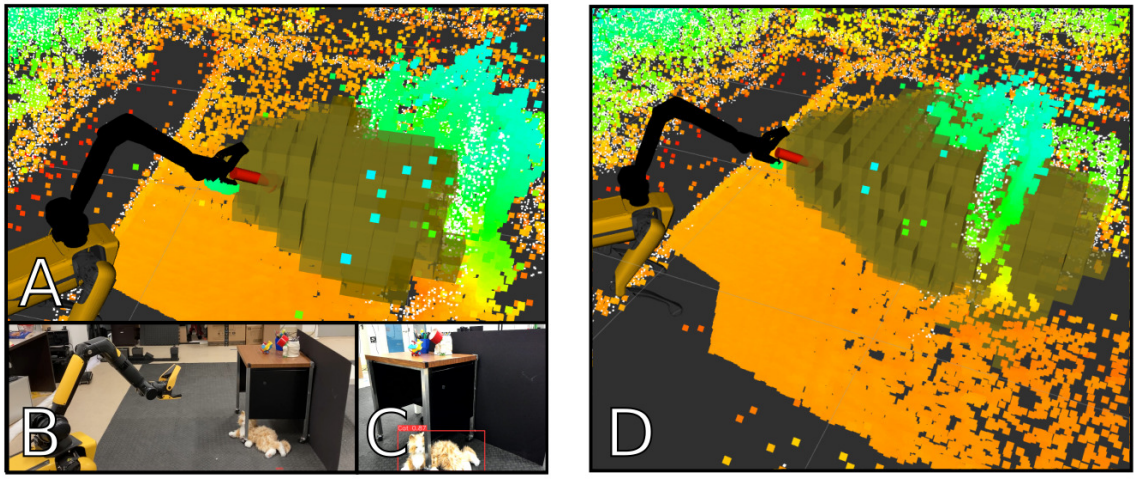


Figure 5.4: For belief update, GenMOS samples a volumetric observation (a set of labeled voxels within the viewing frustum) that considers occlusion based on the occupancy octree dynamically built from point cloud (A). Not enabling occlusion (D) leads to mistaken invisible locations as free. The robot is looking at a table corner (B) with its view blocked by the table and the board (C).

pancy octree (similar to OctoMap (Hornung et al., 2013)), where a leaf node in the tree has an associated value of occupancy (0 for free, and 1 for occupied). The occupancy octree is used by the server for both sampling the view positions graph to avoid collision, as well as for constructing volumetric observations during belief update, where occupied nodes block the FOV and cause occlusion (Figure. 5.4).

Additional uses of point cloud. A key aspect of GenMOS is how point cloud observations are used in three ways: (1) to simulate occlusion; (2) to inform occupancy and initialize octree belief; and (3) to sample a belief-based graph of view positions. We have illustrated (1) above. For (2) and (3), we discuss in more detail in Section 5.1.2 and Section 5.1.3, respectively.

Object detection. The server can also take in 3D object detection bounding boxes, which represent the output of a generic object detector or perception pipeline capable of estimating the 3D locations of detected objects. The bounding box's size plays a role in the octree belief update, as it influences the volumetric observation, where voxels overlapping with the bounding box are labeled by the detected object and leads to an increase in the octree belief at the corresponding locations.

When 3D object detection is not available on the robot, the system can also consume label-only detections based on just images. Such label-only detections essentially correspond to a volumetric observation within the FOV where all voxels are labeled by the object, which usually covers a sizable volume. This is still useful for search, as subsequent search steps can reduce uncertainty by looking from different viewpoints.

Server requests. The server may also actively request information (such as additional observation about the search region's occupancy), which enables our implementation of hierarchical planning in Section 5.2.2; there, 3D local search is triggered by a high-level action to "search locally" and the server would request point cloud data on the fly in order to instantiate 3D-MOS.

Client

Here, I describe several important aspects of the GenMOS client.

Client's role. The client sends to the server configurations of the POMDP agent, perception data, and planning requests, and executes the action returned by the server (Figure 5.3). All data transmitted between the client and the server are represented as Protocol Buffer (protobuf) messages (Varda, n.d.) of generic, robot-independent message types (*e.g.*, point cloud, 3D bounding box, 6D pose, etc.). The client is responsible for integrating with the robot hardware, obtaining sensor data and converting them into the protobuf message types, and physically executing the actions to reach the planned viewpoints. This makes the server code independent of any specific robot.

Planning requsts. When planning requests are sent from the client, the server performs online planning using an asymptotically optimal, Monte Carlo Tree Search-based online POMDP planning algorithm called POUCT (Silver & Veness, 2010).[1] The server converts the planned camera viewpoint $q' \in \mathcal{R}$ to metric coordinates in the frame of the search region. The client then handles moving the robot to that viewpoint. If the server plans a FIND action, the client should send back the detected target objects (if any)[2] The client is also responsible for sending new observations upon action completion.

Next, I follow through with explaining the algorithmic contributions of this chapter that enable the two additional uses of point cloud in GenMOS: belief initialization and view position graph sampling.

5.1.2 Prior Initialization of Octree Belief

Octree belief covers, by definition, a cubic volume; However, the actual feasible search region is likely not cubic, and often irregular. This causes the robot to believe constantly that the target objects are outside of the actual search region, at places imperceivable by the robot, which can impact search behavior.[3]

To address this problem, I propose an efficient algorithm for initializing an octree belief over an arbitrary search region, presented in Algorithm 4. Recall that G denotes the entire

1. See Section 2.2.4, page 28 for an introduction of POUCT.
2. The client may choose to control the robot to physically signal when FIND is taken. For example, with Spot, I let the robot close and reopen its gripper, indicating the robot's commitment to the found location.
3. This is an issued I observed with the proof-of-concept system in Chapter 4.

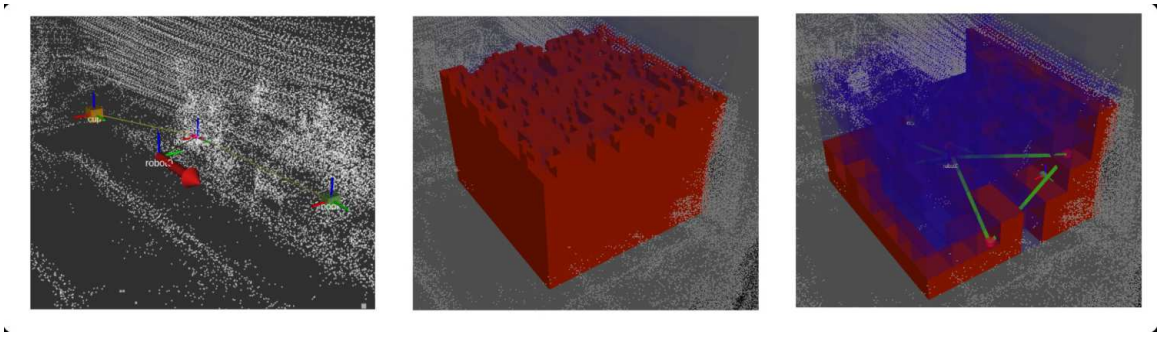


Figure 5.5: Left: A simulation environment where the pose of the robot's viewpoint is represented by the red arrow, and the two target objects are represented by orange and green cubes. Middle: initialized octree belief given uniform prior within a 10.2m$^2\times$2.4m region; Right: initialized octree belief within the same region, given occupancy-based prior constructed from point cloud. Colors indicate strength of belief, from red (high) to blue (low).

3D grid map at ground resolution level underlying an octree belief. Suppose $G_* \subseteq G$ is the subset of grids in G that make up the search region.[4] Recall the definition of *default value* and *initial value* in Definition 1 and Definition 2, respectively (Section 4.4, page 45). The high-level idea of the proposed algorithm is as follows:

1. First, set the default value of all ground-level nodes in the octree belief to 0.

2. Then, through a sample-based procedure (with N samples), ground-level nodes whose 3D positions lie within the given search region G_* have their default values changed to 1.

This effectively reduces the sample space of the octree belief to be within the search region G_*. Besides reducing the sample space, if we are given a prior distribution $\textsc{PriorVal}^i : G_*^l \to \mathbb{R}$, we can initialize the octree belief accordingly as follows, during step 2 above:

3. If a prior probability $\textsc{PriorVal}^i(g^l)$ is defined at octree belief node $g^l \in G_*^l$, and g^l is the parent (or self) of some ground level node $g \in G$ sampled during step 2, then $\textsc{Val}_1^i(g^l)$, the initial value at g^l is set as $\textsc{Val}_1^i(g^l) \leftarrow \textsc{PriorVal}^i(g^l)$.

The algorithm is given below in Algorithm 4. The assignment of initial value at a node is done in lines 8-10 and the tree can be pruned as in lines 12-14. This proposed algorithm has a complexity of $O(N(\log(|G|))^2)$.

4. G_* could be an arbitrary subset, not necessarily forming, for example, a cuboid.

Algorithm 4: Initialize Octree Belief $(m, G_*, \text{PRIORVAL}^i) \rightarrow b_1^i$

input : m: octree dimension (power of 2, such that $|G| = m^3$);
G_*: the actual search region, satisfying $G_* \subseteq G$;
PRIORVAL^i: Prior distribution of octree node values.
param: N: number of samples; B: a 3D box, satisfying $G_* \subseteq B \subseteq G$.
output: b_1^i: the initialized octree belief.

Initialize octree $\Psi(b_0^i)$; Set $\text{VAL}_0^i(g) = 0$ (instead of 1);
for $i \in \{1, \cdots, N\}$ **do**
  Set $l = 0$; Sample $g^l \sim B$; // ground resolution location
  while $l \leq \log_2 m$ **do**
    if $g^l \in G_*^l$ **then**
      Add g^l to $\Psi(b_1^i)$; // insert g^l to octree underlying b_1^i
      Set default value $\text{VAL}_0^i(g^l) = |\text{CH}^0(g^l)|$;
      if $g^l \in \text{PRIORVAL}^i$ **then**
        Set initial value $\text{VAL}_1^i(g^l) \leftarrow \text{PRIORVAL}^i(g^l)$;
        // otherwise $\text{VAL}_1^i(g^l) \leftarrow \text{VAL}_0^i(g^l)$
      end
      // ensure parent value is sum of children
      Update values of all parent nodes at $g^{l+1} \cdots g^m$;
      if $\text{VAL}_1^i(g^l) = \text{VAL}_0^i(g^l)$ **then**
        remove children of g^l ; // Pruning
      end
    end
    $l \leftarrow l + 1$;
  end
end
$\text{NORM}_1 \leftarrow \text{VAL}_1^i(g^m)$; // normalizer set to root node's value

In practice, the server can optionally determine the search region G^* based on the occupancy octree constructed from point cloud observations. This avoids believing that the objects lie in midair. In our experiments, we assign a prior value of $100 \times ((2^k)^3)$ to occupied nodes in the octree at the resolution level $k = 2$, and we set the number of samples $N = 3000$. Figure 5.5 visualizes an octree belief with occupancy-based prior.

5.1.3 Belief-based Sampling for View Position Graph

The evaluation in Sections 4.5.3 and Sections 4.5.4 of the previous chapter only considers moving the camera in cardinal directions, or over a fixed topological map. To enable planning over the continuous space of viewpoints $\mathcal{R} \subseteq \mathcal{P} \times \text{SO}(3)$, GenMOS samples a view position graph $\mathcal{G}_t = (\mathcal{P}_V, \mathcal{E}_M)$ based on the current octree belief. At a high level, given an occupancy octree, we first sample a set of non-occupied positions $\mathcal{P}_V$ from $\mathcal{P}$ with a minimum separation threshold, (*e.g.*, 0.75m) and associate with each position the *belief* by

querying the octree belief at that position at a higher resolution level to cover more space. Then, we select top-K (*e.g.*, $K = 10$) nodes ranked by their beliefs and insert edges such that each node has a limited degree. A $\textsc{Move}(s_r, p_v)$ action then moves the robot to a viewpoint position $p_v \in \mathcal{P}_v$ on the graph. We implicitly enforce a $\textsc{Look}(\phi)$ action after a MOVE action through the transition model where ϕ is the orientation facing the an unfound object (contained in s, input to the transition model). At time $t + 1$, the graph is resampled *if* the sum of the probability covered by positions in $\mathcal{G}_t$ is below a threshold. (*e.g.*, 0.4).

5.1.4 The gRPC Protocol in GenMOS

In the gRPC framework, remote procedural calls (RPCs) are defined as Protocol Buffer messages (Varda, n.d.). In particular, the key RPCs in GenMOS are as follows:

- `CreateAgent`: Upon receiving the POMDP agent configurations from the client, the server prepares for agent creation pending the first `UpdateSearchRegion` call.
- `UpdateSearchRegion`: The client sends over a point cloud of the local search region, and the server creates or updates the occupancy octree about the search region.
- `ProcessObservation`: The client requests belief update by sending observations such as object detection and robot pose estimation.
- `CreatePlanner`: The client provides hyperparameters of the planner, and the server creates a planner instance accordingly (*e.g.*, POUCT planner in `pomdp_py` (Zheng & Tellex, 2020)).
- `PlanAction`: The client requests the server to plan an action for an agent. An action is planned only if the last planned action has been executed successfully.
- `ListenServer`: This is a bidirection streaming RPC that establishes a channel of communication of messages or status between the client and the server.

5.1.5 Example Configuration Parameters

The table below lists some parameters that the GenMOS server is able to handle.

`octree_size`	Dimension of the octree representing the search region (*e.g.*, 32 means the octree occupies a 32^3 grid)
`res`	Resolution of a grid, *i.e.*, length of the grid's side in meters (*e.g.*, 0.1)
`region_size`	Defines the dimensions of a box (w, ℓ, h) in meters (*e.g.*(4.0, 3.0, 1.5))
`center`	Defines the XYZ location (metric)of the search region's center (*e.g.*(-0.5, -1.65, 0.25)))
`prior_from_occupancy`	"True" to use occupancy-based prior

`occupancy_fill_height`	"True" to consider the space below obstacles into search space
`num_nodes`	maximum number of view positions on graph (*e.g.*, 10)
`sep`	minimum separation between nodes (in meters) (*e.g.*, 0.4)
`inflation`	radius to blow up obstacles for view position sampling
`num_sims`	number of samples for MCTS-based online POMDP planning.

Table 5.1: Example configuration parameters in GenMOS

5.2 Evaluation of GenMOS

We implemented the gRPC protocols of the GenMOS system described in Section 5.1.4. The result is a single package for multi-object search in 3D regions that can provide the object search functionality as long as the perception inputs are given, which are generic point cloud and object detection results that a robot typically should be able to provide.

There are two hypotheses that we test through our evaluation: (1) The octree belief-based planning algorithm that the package implements is effective for 3D object search; (2) The package does enable real robots to search for and find objects in 3D regions in different environments within a reasonable time budget.

To test the first hypothesis, we conduct an experiment in simulation (Section 5.2.1). To test the second hypothesis, we deploy our system for object search with a Boston Dynamics Spot robot in two different local regions: a region of arranged tables and a kitchen region (Section 5.2.2), and we also implement a preliminary hierarchical planning algorithm for a demonstration over a larger lobby area (Section 5.2.2). We further integrate GenMOS with Kinova MOVO and Universal Robotics UR5e robotic arm and test search behavior enabled by GenMOS.

5.2.1 Evaluation in Simulation

We tasked a simulated robot (represented as an arrow for its viewpoint) to search for two virtual objects (cubes) with volume 0.002m^3 each uniformly randomly placed in a region of size 10.2m$^2\times$ 2.4m. The robot's frustum camera model had a FOV angle of 60 degrees, minimum range of 0.2m and maximum range of 2.0m.

We experimented with three types of priors, groundtruth, uniform, and occupancy-based prior, at two different resolution levels, $0.001m^3$ (octree size $32\times32\times32$) and $0.008m^3$ (octree size $16\times16\times16$) representing search granularity. For the best-performing setting (non-groundtruth), we also compared the use of the POUCT planner against two baselines: Random moves to a uniformly sampled view position graph node (Section 5.1.3), and Greedy is a next-best view planner that moves to the view position graph node that is

Prior type (resolution) with POUCT	Length (m)	Planning time (s)	Total time (s)	success rate
Uniform (0.008m^3)	22.13	24.28	166.18	50%
Occupancy (0.008m^3)	23.89	22.66	159.10	60%
Uniform (0.001m^3)	6.42	10.47	99.66	90%
Occupancy (0.001m^3)*	3.22	7.42	64.12	100%
Groundtruth	0.44	1.97	17.82	100%
*with Random	12.18	0.19	167.20	55%
*with Greedy	3.48	0.12	81.80	85%

Table 5.2: Simulation results. We compare the search performance between different prior belief and resolution settings. The results for the first three colums are averaged over 20 trials.

closest to the highest belief location for some target. Both baseline planners take FIND upon target detection.

We evaluate the search performance by four metrics: total path length traversed during search (Length), *total* time used for POMDP planning (Planning time), total system time (Total time), and success rate. Total system time included time for planning, executing navigation actions, receiving observations, belief update and visualization; the simulated robot has a translational velocity of 1.0m/s, and a rotational velocity of 0.87rad/s.

We perform 20 search trials per method and report the average of each metric in Table 5.2.[5] Each trial was allowed 180s total system time (excluding the time for visualization). Results indicate that the system achieved high success rate especially at high resolution under occupancy-based prior. We observed that searching with a resolution level more coarse than the target size hurts performance, while having occupancy-based prior improves. Additionally, Greedy was much faster than POUCT in planning time yet lead to lower success rate within the time budget and longer total time than using POUCT. Our intuition is that, while Greedy prioritizes looking at a location with the highest belief, POUCT considers the search of multiple objects in a sequence.

5.2.2 Deployment on the Boston Dynamics Spot

We deploy our system to the Boston Dynamics Spot (*Boston Dynamics Spot*, 2019) by writing a client for GenMOS that interfaces with the Spot SDK.[6] Spot is a mobile robot that is robust at navigation while avoiding obstacles. Our Spot robot is equipped with an arm that has a gripper with an RGB-D camera, which has a depth range of around 1.5m. However,

5. Simulation experiments were run on a computer with i7-8700 CPU.
6. We integrated Spot SDK with ROS (Quigley et al., 2009) to use RViZ (Kam et al., 2015); Our computer that ran GenMOS for Spot has an i7-9750H CPU with an RTX 2060 GPU.

motion planning of the arm does not have collision checking. Nevertheless, our package is able to output viewpoints that are of safe distances from obstacles to enable collision-free search, leveraging the point cloud received from the Spot's on-board cameras. We use Spot's off-the-shelf GraphNav service to map the search region (without the presence of the target objects) and then localize the robot within it.

Figure 5.6: Candidate target objects in our evaluation. From left to right, the object labels are: Columbia Book, Robot Book, Bowl, Lysol, ToyPlane, Pringles, and Cat.

We task the robot to search in 2 different local regions in different rooms of our lab (Figure 5.7). The first region (of size $9\text{m}^2 \times 1.5\text{m}$) consists of two tables and a separation board which creates occlusion; The target objects can be on the floor, or on or under tables; Note that our system is given only point cloud observations to infer potential target locations. The second region (of size $7.5\text{m}^2 \times 2.2\text{m}$) is a kitchen area, where target objects can be on the countertop, on or underneath the couch, on the shelf, or in the sink. In both environments, the resolution of the octree belief is set to $0.001m^3$ with a size of $32\times32\times32$. The robot is given at most 10 minutes to search. We collected a dataset of 230 images and trained a YOLOv5 detector (Jocher et al., 2020) with 1.9 million parameters for the objects of interest (Figure 5.6). We project the 2D bounding box to 3D using the depth image from the gripper camera.

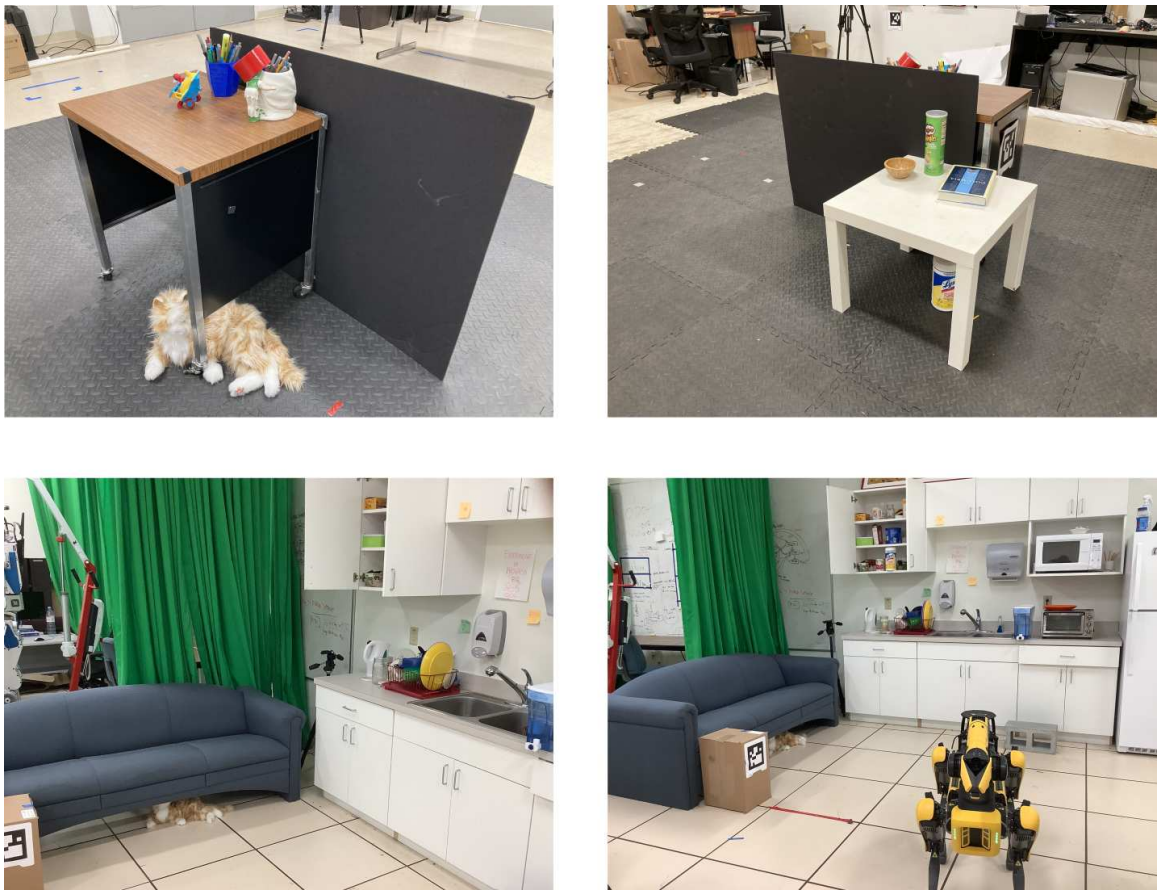


Figure 5.7: Local regions in our evaluation with Spot. Upper two: two views of the arranged tables region. A black board separates the two tables to block the view from one side to the other. Bottom two: two views of the kitchen region, with a couch, a countertop, and a shelf.

Figure 5.8 contains illustrations of several key frames during the search trials in both regions. Video footages of the search together with belief state visualization are available in the supplementary video. In the arranged tables region, our system enables Spot to simultaneously search for four objects (Cat, Pringles, Lysol, and ToyPlane), and successfully find three objects in 6.5 minutes. In the kitchen region, our system enables Spot to find a Cat placed underneath the couch within one minute. However, we do observe that search success deteriorates due to false negatives from the object detector, as well as conservative viewpoint sampling for obstacle avoidance, which prevents the robot to plan top-down views from above the countertop, for example. Overall, our system enables the robot to search for objects in different environments within a moderate time budget.

Figure 5.8: Key frames from local region search trials. Each frame consists of three images: a third-person view (top), an image from Spot's gripper camera with object detection (bottom left), and a combined visualization of the octree belief, viewpoint graph, and local point cloud observations. Green boxes indicate successfully finding the marked object. Red boxes indicate failure of finding the object due to false negatives in object detection. The yellow or white box on the right of each frame indicates the amount of time passed since the start of the search. Frames at the top row belong to a single trial in the table region, while frames at the bottom row belong to distinct trials in the kitchen region. The top row (1-4) shows that GenMOS enables Spot to successfully find multiple objects in the table region: Lysol under the white (2), Pringles on the white table (3), and the Cat on the fllor under the wooden table (4). The bottom row shows that GenMOS enables Spot to find a Cat underneath the couch (5), and the Pringles at the countertop corner (6). (7-8) shows a failure mode, where the GenMOS plans a reasonable viewpoint, while the object detector fails to detect the object (Cat) on the shelf or in the sink. Video: https://youtu.be/TfCe2ZVwypU

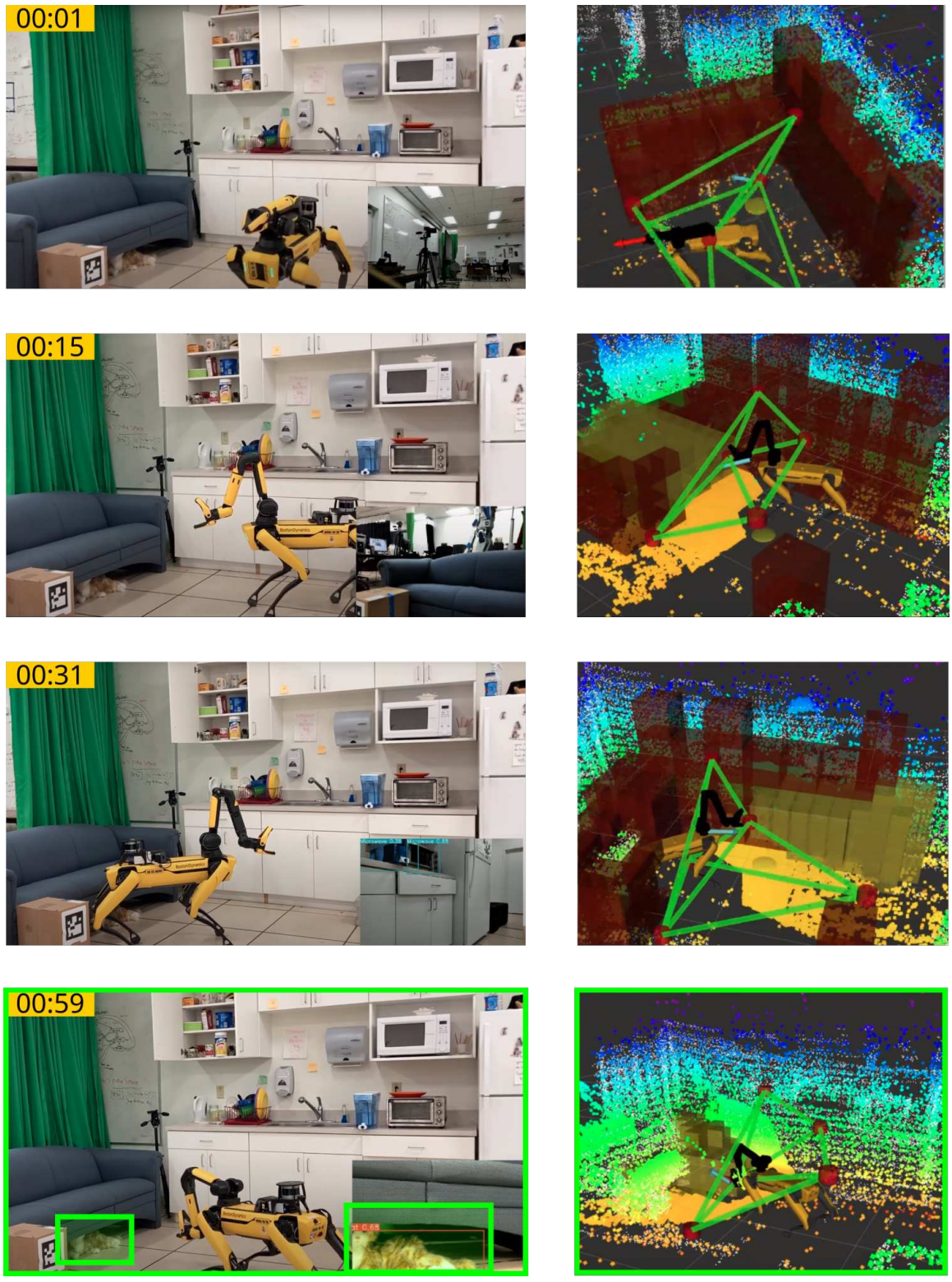


Figure 5.9: Sequence of frames from the search trial where Spot is tasked to find the toy cat under the couch. GenMOS enables Spot to find the hidden cat under one minute. Red boxes represent octree belief, initialized based on occupancy.

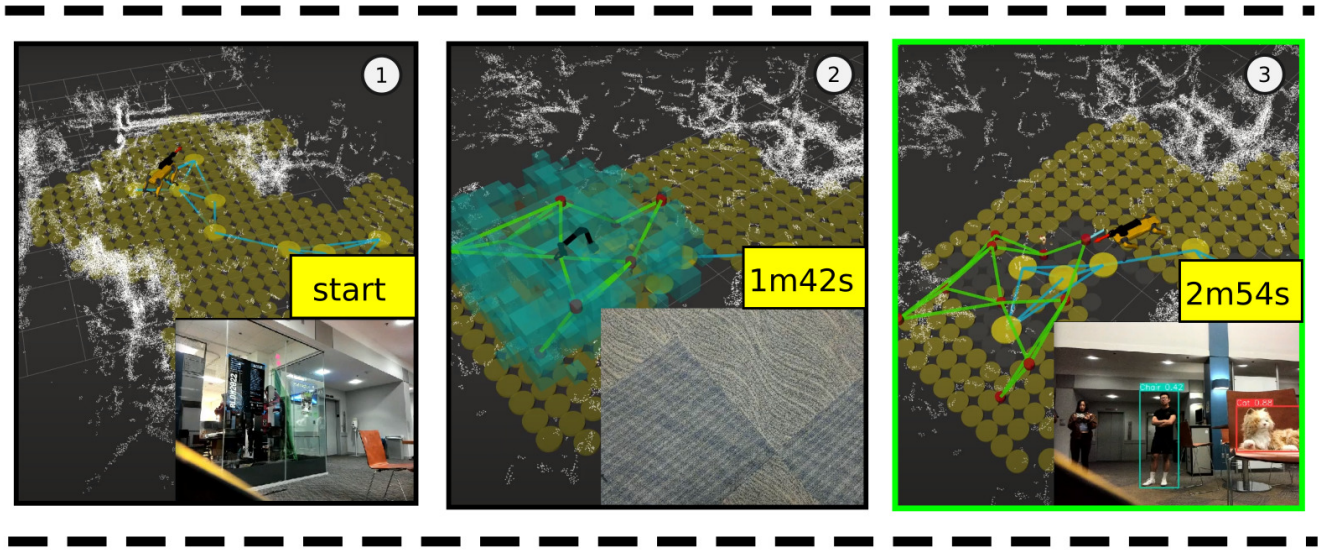


Figure 5.10: Demonstration of hierarchical planning where a 2D global search is integrated with 3D local search through the *stay* action (Zheng et al., 2022). This system enables the Spot robot to find a Cat in a lobby area within 3 minutes. (1) Initial state; (2) searching in a 3D local region; (3) the robot detects the Cat and the search finishes.

Extension to Hierarchical Planning

We envision the integration of our 3D local search algorithm with a global search algorithm so that a larger search space can be handled. To this end, we implemented a hierarchical planning algorithm that contains a 2D global planner (with the same multi-object search POMDP model as in Chapter 4 but in 2D), where the global planner has a *stay* action (no viewpoint change) which triggers the initialization of a 3D local search agent. In particular, our implementation uses the `ListenServer` streaming RPC; when the planner *decides* to search locally, we let the server send a message that triggers the client to send over an `UpdateSearchRegion` request to initialize the local 3D search agent.

The starting belief of the 3D local agent is initialized based on the 2D global belief; the 2D global belief is in turn updated by projecting the 3D field of view down to 2D. We set the resolution of 2D search to be $0.09m^2$, and the resolution of 3D search to be $0.001m^3$. We test this system in a lobby area of size $25\text{m}^2 \times 1.5\text{m}$, where the robot is tasked to find the toy cat on a tall chair (Figure 5.10). The search succeeded within three minutes, covering roughly 15m^2.

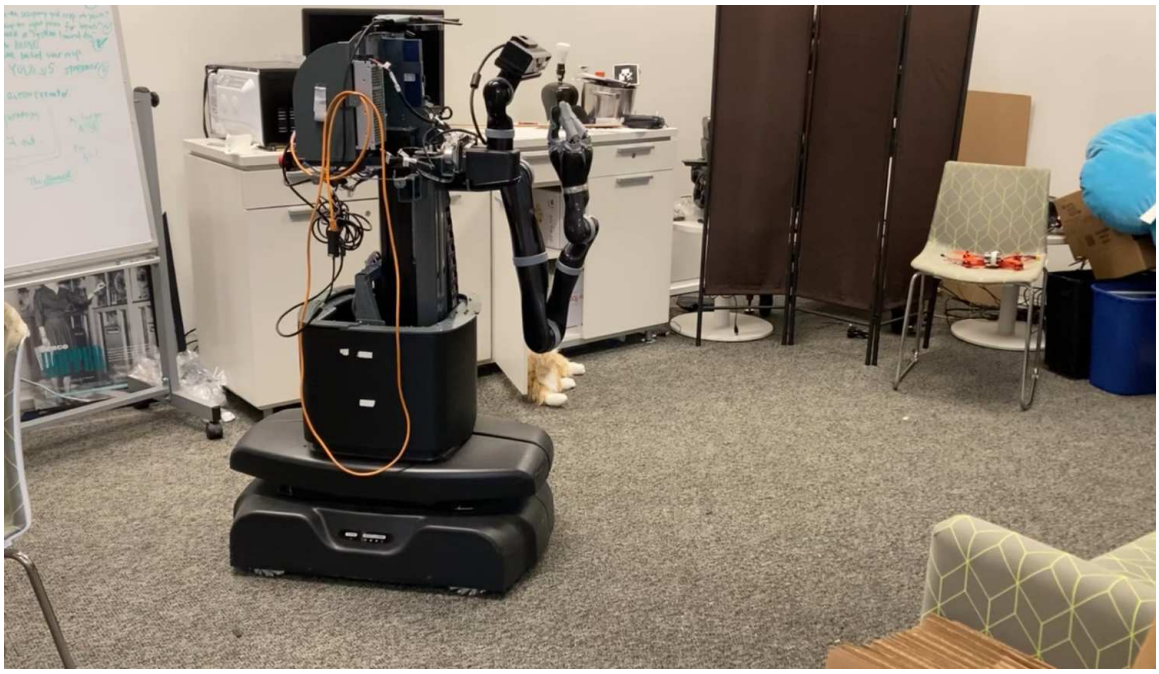


Figure 5.11: Test environment for object search with MOVO using GenMOS. The target object is, again, the toy cat. In this case, it is lying on the floor next to the opened door.

5.2.3 Deployment on the Kinova MOVO Robot

We additionally deployed GenMOS to the Kinova MOVO mobile manipulator, a robot with a mobile base, an extensible torso, and a head that can pan and tilt, and it is equipped with a Kinect V2 RGBD camera. Similar to Spot, we deployed GenMOS to MOVO by integrating the GenMOS gRPC client with the perception, navigation and control stacks of MOVO, which is based on ROS Kinetic. Since the maintenance of MOVO by Kinova has terminated since 2019, deploying GenMOS on MOVO poses a greater challenge compared to Spot. Nevertheless, through Docker (Merkel, 2014), GenMOS was successfully integrated through implementing a client for MOVO, and it enabled MOVO to do object search.

We evaluated the resulting object search system in a small living room environment (Figures 5.12 and 5.13). The robot is able to perform search and successfully finds a toy cat on the floor in around 2 minutes. Compared to Spot, however, MOVO is less agile and prone to collision with obstacles while navigating between viewpoints during the search.

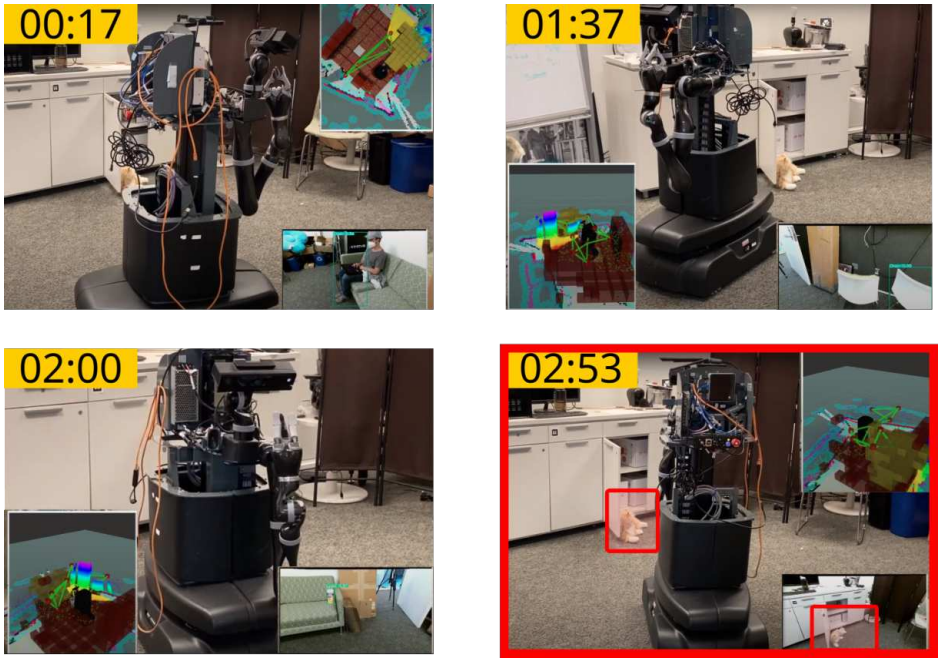


Figure 5.12: Here, the toy cat lies on the floor next to the opened door. MOVO eventually looked in the right direction, but the object detector failed to recognize it.

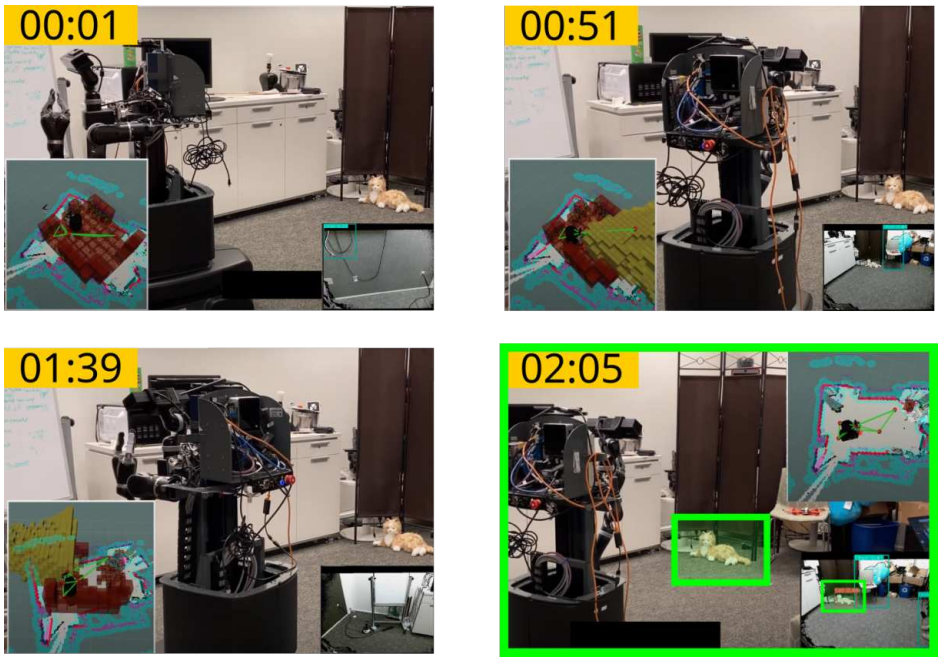


Figure 5.13: Here, the toy cat is in front of the room divider. Although the detector failed at first (2), MOVO recovered and found the target on the second try (4).

Figure 5.14: Test environment for UR5e. The robot's gripper had a camera. The target object is a cup placed either on or slightly under the farther table, initially out of sight for the gripper camera.

5.2.4 Deployment on the Universal Robotics UR5e Robotic Arm

Finally, we integrate GenMOS with the UR5e robotic arm on yet a different middleware, Viam.[7] The UR5e arm is mounted on a table and it is equipped with a camera on its gripper. We applied an off-the-shelf RGB object detection model trained on MS-COCO (T.-Y. Lin et al., 2014) and tasked the arm to find a red cup. The cup is initially out of sight, either on or below a different table. Since object detection lacks depth, we considered label-only detection (*i.e.*, discarding the 2D bounding box and only keeping the event that a certain object is detected). As discussed in Section 5.1.1 (page 58), GenMOS can accommodate to such a scenario as it is expected to look from different viewpoints to reduce the region of uncertainty, once a detection is made. Indeed, we observed this type of behavior on UR5e (see Figure 5.15).

A few caveats should be noted about this particular system. I expected the arm to be able to reliably motion plan to viewpoints produced by GenMOS. However, in practice, motion planning frequently fails.[8] As a work-around that still tests GenMOS's object search planning ability, I predefined a set of viewpoints for which motion planning works, and when a viewpoint is planned by GenMOS, the arm is moved to the closest viewpoint in this set. Another issue that I worked around is that the detector works poorly if the image is rotated due to gripper rotation. So, I made the gripper automatically level every time it reaches a destination view pose, which was done by commanding the last joint to offset the end effector's rotation around the wrist.

7. As an alternative to ROS, Viam (https://www.viam.com/) aims to provide fast configuration of robot hardware and distributed robot systems through an extensive web interface that accelerates collaboration as well as standardized services for vision and motion planning as building blocks.
8. This is in part due to the fact that motion services with Viam were in early phases of development when I visited, but also that GenMOS does not internally consider kinematic constraints; it works if motion planning fails a few times as GenMOS would simply replan, but the failure was too frequent at the time to keep trying motion planning to any viewpoint produced by GenMOS .

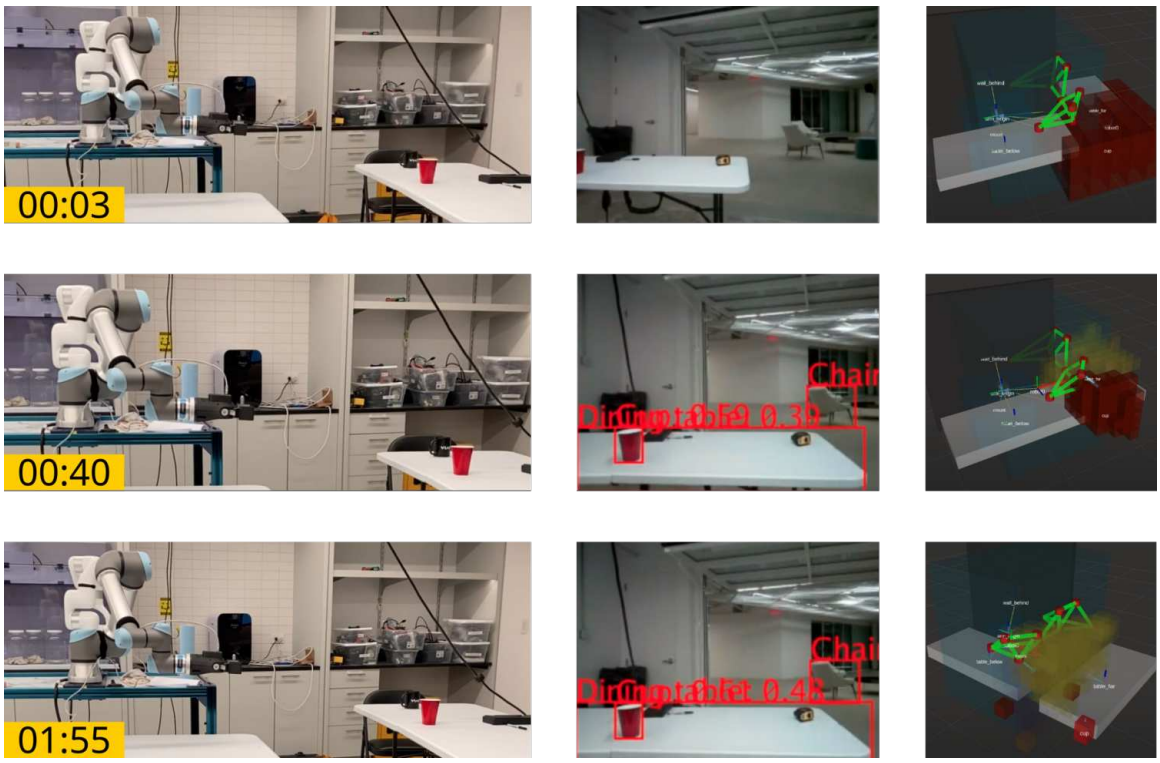


Figure 5.15: The UR5e arm moves back and forth, reducing uncertainty to a few grids.

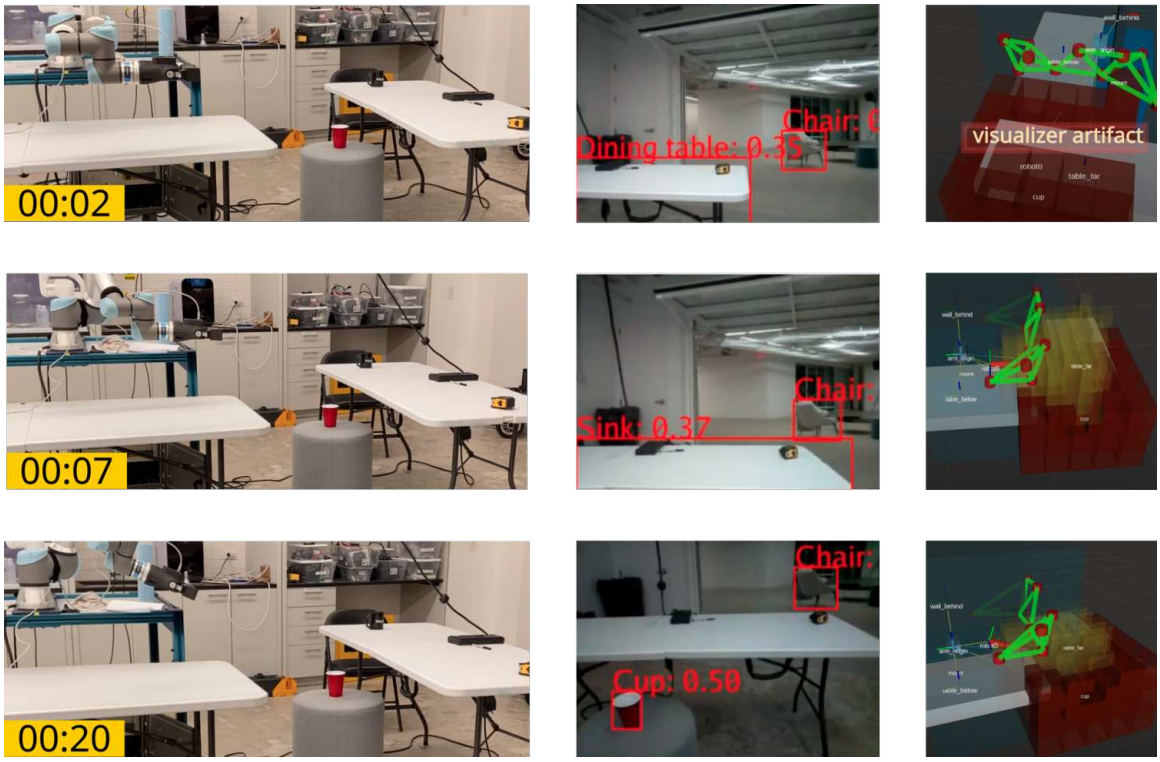


Figure 5.16: The UR5e arm looks down; yet belief update missed the detection.

CHAPTER 6

Correlational Object Search

6.1 Motivation - Finding Hard-to-Detect Objects

OBJECT search can make a difference in many applications including domestic services (Sprute et al., 2017; Zeng et al., 2020), search and rescue (Eismann et al., 2009; Sun et al., 2016), and elderly care (Idrees et al., 2020; Loghmani et al., 2018). In realistic settings, however, the object being searched for will often be small, outside the current field of view, and hard to detect. For example, a household robot must be very close to a fork in order to be able to detect it; likewise, a warehouse robot may have to locate a particular machine within a very large factory. To be effective, the robot must generate efficient search strategies that require as few timesteps as possible.

6.1.1 Why Correlations?

In such settings, when the target object is hard to detect, *correlational information* can be extremely useful. Specifically, suppose the robot is equipped with a prior about the relative spatial locations of object types (*e.g.*, refrigerators tend to be near forks). Then, it can leverage this information as a powerful heuristic to narrow down or "focus" the search space, by first focusing its efforts on locating easier-to-detect objects that are highly correlated with the target object, and only then focusing on locating the target object itself. Doing so has the potential to greatly improve search efficiency, as the robot no longer needs to waste time considering strategies that, *e.g.*, search for a fork in a bathroom. Unfortunately, previous approaches to object search with correlational information tend to resort to ad-hoc or greedy search strategies (Aydemir et al., 2013; Kollar & Roy, 2009; Zeng et al., 2020), which may not scale well to complex environments.

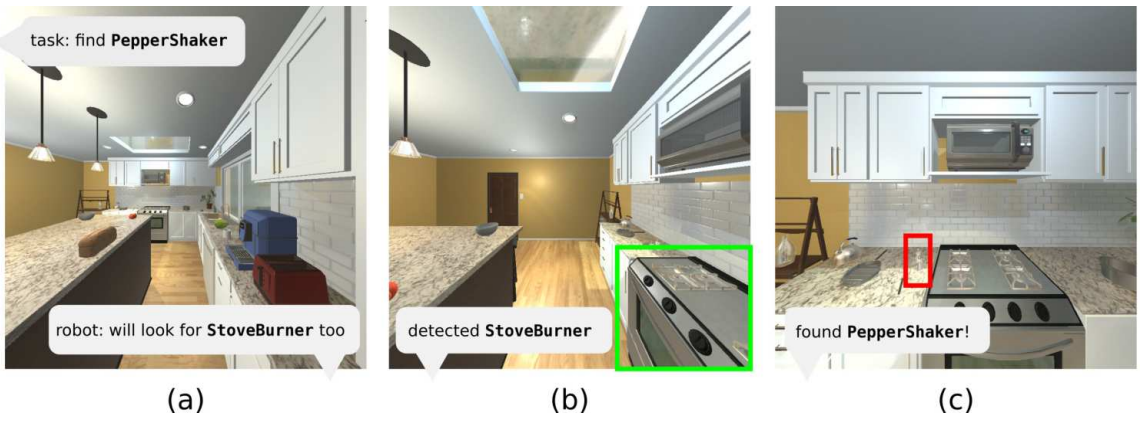


Figure 6.1: We study the problem of object search using correlational information about spatial relations between objects. This example illustrates a desirable search behavior in an AI2-THOR scene, where the robot leverages the detection of a `StoveBurner` to more efficiently find a hard-to-detect `PepperShaker`.

6.1.2 Remark on Previous Work

We follow a long line of work that models the object search problem as a partially observable Markov decision process (POMDP) (Aydemir et al., 2013; J. K. Li et al., 2016; Xiao et al., 2019; Wandzel et al., 2019; Zheng et al., 2021a). This formalization is useful because object search over long horizons is naturally a sequential, partially observed decision-making problem: (1) the robot must search for the target object by visiting multiple viewpoints in the environment sequentially, and (2) the robot must maintain and update a measure of uncertainty over the location of the target object, via its belief state.

Garvey (1976) and Wixson & Ballard (1994) pioneered the paradigm of *indirect search*, where an intermediate object (such as a desk) that is typically easier to detect is located first, before the target object (such as a keyboard). More recently, probabilistic graphical models have been used to model object-room or object-object spatial correlations (Aydemir et al., 2013; Zeng et al., 2020; Kollar & Roy, 2009; Lorbach et al., 2014). In particular, Zeng et al. (2020) proposed a factor graph representation for different types of object spatial relations. Their approach produces search strategies in a greedy fashion by selecting the next-best view to navigate towards, based on a hybrid utility of navigation cost and the likelihood of detecting objects. In our evaluation, we compare our sequential decision-making approach with a greedy, next-best view baseline based on that work (Zeng et al., 2020).

Recently, the problem of semantic visual navigation (Y. Zhu et al., 2017; Batra et al., 2020; Wortsman et al., 2019; Qiu et al., 2020; Mayo et al., 2021) received a surge of interest in the deep learning community. In this problem, an embodied agent is placed in an unknown environment and tasked to navigate towards a given semantic target (such as "kitchen" or "chair"). The agent typically has access to behavioral datasets for training on the order of millions of frames and the challenge is typically in generalization. Our

work considers the standard evaluation metric (SPL (Anderson et al., 2018)) and task success criteria (object visibility and distance threshold (Batra et al., 2020)) from this body of work. However, our setting differs fundamentally in that the search strategy is not a result of training but a result of solving an optimization problem.

6.2 Contributions

In this work, we make the following contributions:

- We propose COS-POMDP, which contains a correlation-based observation model that captures spatial relations between objects
- We prove that COS-POMDPs produce equivalent solutions to a naive formulation where the state space is a joint over all object locations, despite COS-POMDPs having a much smaller state space;
- We address scalability by proposing a hierarchical planning algorithm, where a high-level COS-POMDP plans subgoals, each fulfilled by a low-level planner that plans with low-level actions (*i.e.*, given primitive actions); both levels plan online based on a shared and updated COS-POMDP belief state, enabling closed-loop planning;
- We investigate the influence of correlational information when searching for hard-to-detect targets, and the benefit of optimizing for a sequence of actions as opposed to selecting the next-best view.
- We conduct experiments in AI2-THOR (Kolve et al., 2017), a realistic simulator of household environments, and use YOLOv5 (Redmon et al., 2016; Jocher et al., 2020) as the object detector.
    - Our results show that, when the given correlational information is accurate, COS-POMDP leads to more robust search perfomance when the target object is hard to detect. In particular, for target objects with a true positive detection rate below 40%, COS-POMDP improves the POMDP baseline that ignores correlational information by 70% and a greedy, next-best view baseline by 170%, in terms of the SPL (Anderson et al., 2018) metric, commonly used for evaluating navigation agents in simulated environments (Wortsman et al., 2019; Qiu et al., 2020; Batra et al., 2020).

6.3 Problem Formulation

We formulate correlational object search as a planning problem, where a robot must search for a target object given correlational information with other objects in the environment. We begin by describing the underlying search environment and the capabilities of the robot. Then we define the inputs to the robot and the solution expected to be produced by the robot to solve this problem.

6.3.1 Search Environment and Robot Capabilities

The search environment contains a target object and n additional static objects. The set of possible object locations is discrete, denoted as $\mathcal{X}$. The locations of the target object $x_{\text{target}} \in \mathcal{X}$ and other objects $x_1, \ldots, x_n \in \mathcal{X}$ are unknown to the robot, and follow a latent joint distribution $\Pr(x_1, \ldots, x_n, x_{\text{target}})$. The robot is given as input a factored form of this distribution, defined later in Sec. 6.3.2.

The robot can observe the environment from a discrete set of viewpoints, where each viewpoint is specified by the position and orientation of the robot's camera. These viewpoints form the necessary state space of the robot, denoted as $\mathcal{S}_{\text{robot}}$. The initial viewpoint is denoted as $s^{\text{init}}_{\text{robot}}$. By taking a primitive move action a from the set $\mathcal{A}_m$, the robot changes its viewpoint subject to transition uncertainty $T_m(s'_{\text{robot}}, s_{\text{robot}}, a) = \Pr(s'_{\text{robot}} | s_{\text{robot}}, a)$. Also, the robot can decide to finish a task at any timestep by choosing a special action `Done`, which deterministically terminates the process.

At each timestep, the robot receives an observation z factored into two independent components $z = (z_{\text{robot}}, z_{\text{objects}})$. The first component $z_{\text{robot}} \in \mathcal{S}_{\text{robot}}$ is an estimation of the robot's current viewpoint following the observation model $O_{\text{robot}}(z_{\text{robot}}, s_{\text{robot}}) = \Pr(z_{\text{robot}} | s_{\text{robot}})$. The second component $z_{\text{objects}} = (z_1, \ldots, z_n, z_{\text{target}})$ is the result of performing object detection. Each element, $z_i \in \mathcal{X} \cup \{\texttt{null}\}$, $i \in \{1, \ldots, n, \text{target}\}$, is the detected location of object i within the field of view, or `null` if not detected. The observation z_i about object i is subject to limited field of view and sensing uncertainty captured by a *detection model* $D_i(z_i, x_i, s_{\text{robot}}) = \Pr(z_i | x_i, s_{\text{robot}})$; Here, a common conditional independence assumption in object search is made (Zeng et al., 2020; Wandzel et al., 2019), where z_i is conditionally independent of the observations and locations of all other objects given its location and the robot state s_{robot}. The set of detection models for all objects is $\mathcal{D} = \{D_1, \ldots, D_n, D_{\text{target}}\}$. In our experiments, we obtain parameters for the detection models based on the performance of the vision-based object detector (Sec. 6.6.1).

6.3.2 The Correlational Object Search Problem

Although the joint distribution of object locations is latent, the robot is assumed to have access to a factored form of that distribution, that is, n conditional distributions, $\mathcal{C} = \{C_1, \ldots, C_n\}$ where $C_i(x_i, x_{\text{target}}) = \Pr(x_i | x_{\text{target}})$ specifies the spatial correlation between the target and object i. We call each C_i a *correlation model*. This model can be learned from data or specified based on environment-specific knowledge.

The robot performs search by taking a sequence of move actions to observe different parts of the environment, and terminates the search by taking `Done`. We are now ready to define the *correlational object search* problem:

Problem 1 (Correlational Object Search). Given as input a tuple

$$(\mathcal{X}, \mathcal{C}, \mathcal{D}, s^{\text{init}}_{\text{robot}}, \mathcal{S}_{\text{robot}}, O_{\text{robot}}, \mathcal{A}_m, T_m),$$

the robot must perform a sequence of actions, $a_{1:T} = (a_1, \ldots, a_T)$ of length $T \geq 1$, where $a_1, \ldots, a_{T-1} \in \mathcal{A}_m$ and a_T is Done. The action sequence $a_{1:T}$ is called a *solution*. A solution is *successful* if the robot state sequence and the target location satisfy certain criteria upon taking the Done action. In our evaluation in AI2-THOR, we use the success criteria recommended by Batra et al. (2020) and the commonly-used SPL metric proposed by Anderson et al. (2018) to measure the efficiency of successful searches. The objective is to produce a successful solution that reaches the target object while minimizing the total distance traveled by the robot.

6.4 Correlational Object Search as a POMDP

The POMDP is an extensively studied framework for optimizing sequential decisions under partial observability and uncertainty in motion and sensing. Both challenges (partial observability and uncertainty) considered in POMDP arise naturally in object search. In addition, the objective of minimizing the navigation distance while successfully finding the target can be represented by the POMDP objective of maximizing the discounted cumulative rewards. Therefore, we model the correlational object search task as a POMDP.

We first provide a condensed review of POMDPs; for more information, we refer the reader to (Kaelbling et al., 1998; Shani et al., 2013; Somani et al., 2013; Silver & Veness, 2010). Then, we present the COS-POMDP, a POMDP formulation that addresses the correlational object search problem, followed by a discussion on its optimality. COS-POMDP expands the observation space of the overarching object search POMDP (Chapter 3) by considering observations about objects other than the target object, and formulates the corresponding observation model using spatial correlation.

6.4.1 COS-POMDP

Given an instance of the correlational object search problem defined in Sec. 6.3.2, we define the **C**orrelational **O**bject **S**earch POMDP (COS-POMDP) as follows:

- **State space.** The state space $\mathcal{S}$ is factored to include the robot state $s_{\text{robot}} \in \mathcal{S}_{\text{robot}}$ and the target state $x_{\text{target}} \in \mathcal{X}$. A state $s \in \mathcal{S}$ can be written as $s = (s_{\text{robot}}, x_{\text{target}})$. Importantly, no other object state is included in $\mathcal{S}$.
- **Action space.** The action space is $\mathcal{A} = \mathcal{A}_m \cup \{\text{Done}\}$.
- **Observation space.** The observation space $\mathcal{Z}$ is factored over the objects, and each $z \in \mathcal{Z}$ is written as $z = (z_{\text{robot}}, z_{\text{objects}})$, where $z_{\text{objects}} = (z_1, \ldots, z_n, z_{\text{target}})$.
- **Transition model.** The objects are assumed to be static. Actions $a_m \in \mathcal{A}_m$ change the robot state from s_{robot} to s'_{robot} according to T_m, and taking the Done action terminates the task deterministically.

- **Observation model.** By definition of z, we have

$$\Pr(z|s) = \Pr(z_{\text{robot}}|s_{\text{robot}}) \Pr(z_{\text{objects}}|s) \tag{6.1}$$
$$= O_{\text{robot}}(z_{\text{robot}}, s_{\text{robot}}) \Pr(z_{\text{objects}}|s) \tag{6.2}$$

Under the conditional independence assumption in Sec. 6.3, $\Pr(z_{\text{objects}}|s)$ can be compactly factored as:

$$\Pr(z_{\text{objects}}|s) = \Pr(z_1, \ldots, z_n, z_{\text{target}}|x_{\text{target}}, s_{\text{robot}}) \tag{6.3}$$
$$= \Pr(z_{\text{target}}|x_{\text{target}}, s_{\text{robot}}) \prod_{i=1}^{n} \Pr(z_i|x_{\text{target}}, s_{\text{robot}}) \tag{6.4}$$

The first term in Eq (6.4) is defined by D_{target}, and each $\Pr(z_i|x_{\text{target}}, s_{\text{robot}})$ is called a *correlational observation model*, written as:

$$\Pr(z_i|x_{\text{target}}, s_{\text{robot}}) = \sum_{x_i \in \mathcal{X}} \Pr(x_i, z_i|x_{\text{target}}, s_{\text{robot}}) \tag{6.5}$$
$$= \sum_{x_i \in \mathcal{X}} \Pr(z_i|x_i, s_{\text{robot}}) \Pr(x_i|x_{\text{target}}) \tag{6.6}$$

where the two terms in Eq (6.6) are the detection model $D_i \in \mathcal{D}$ and correlation model $C_i \in \mathcal{C}$, respectively.

- **Reward function.** The reward function, $R(s, a) = R(s_{\text{robot}}, x_{\text{target}}, a)$, is defined as follows. Upon taking `Done`, the task outcome is determined based on $s_{\text{robot}}, x_{\text{target}}$, which is successful if the robot orientation is facing the target and its position is within a distance threshold to the target. If successful, then the robot receives $R_{\text{max}} \gg 0$, and $R_{\text{min}} \ll 0$ otherwise. Taking a move action from $\mathcal{A}_m$ receives a negative reward which corresponds to the action's cost. In our experiments, we set $R_{\text{max}} = 100$ and $R_{\text{min}} = -100$. Each primitive move action (*e.g.*, `MoveAhead`) receives a step cost of -1.

6.4.2 Optimality of COS-POMDPs

The state space of a COS-POMDP involves only the robot and target object states. A natural question arises: have we lost any necessary information? In this section, we show that COS-POMDPs are optimal, in the following sense. If we imagine solving a "full" POMDP corresponding to the COS-POMDP, whose state space contains all object states, then the solutions to the COS-POMDP are equivalent. Note that a belief state in this "full" POMDP scales exponentially in the number of objects.

We begin by precisely defining the "full" POMDP, henceforth called the F-POMDP, corresponding to a COS-POMDP. The F-POMDP has identical action space, observation space, and transition model as the COS-POMDP. The reward function is also identical since it only depends on the target object state, robot state, and the action taken. F-POMDP differs in the state space and observation model:

- **State space:** The state is $s = (s_{\text{robot}}, x_{\text{target}}, x_1, \ldots, x_n)$.
- **Observation model:** Under the conditional independence assumption stated in Sec. 6.3, the model for observation z_i of object x_i involves just the detection model: $\Pr(z_i|s) = \Pr(z_i|x_i, s_{\text{robot}})$.

Since the COS-POMDP and the F-POMDP share the same action and observation spaces, they have the same history space as well. We first show that given the same policy, the two models have the same distribution over histories.

Theorem 1. Given any policy $\pi : h_t \to a$, the distribution of histories is identical between the COS-POMDP and the F-POMDP.

Proof. See Appendix 6.8.1 (page 91). □

Using Theorem 1, we are equipped to make a statement about the value of following a given policy in either the COS-POMDP or the F-POMDP.

Corollary 1. Given any policy $\pi : h_t \to a$ and h_t, the value $V_\pi(h_t)$ is identical between the COS-POMDP and the F-POMDP.

Proof. By definition, the value of a POMDP at a history is the expected discounted cumulative reward with respect to the distribution of future action-observation pairs. Theorem 1 states that the COS-POMDP and F-POMDP have the same distribution of histories given π. Furthermore, the reward function depends only on the states of the robot and the target object. Thus, this expectation is equal for the two POMDPs at any h. □

Finally, we can show that COS-POMDPs are optimal in the sense as discussed before.

Corollary 2. An optimal policy π^* for either the COS-POMDP or the F-POMDP is also optimal for the other.

Proof. Suppose, without loss of generality, that π^* is optimal for the COS-POMDP but not the F-POMDP. Let π' be the optimal policy for the F-POMDP. By the definition of optimality, for at least some history h we must have $V_{\pi'}(h) > V_{\pi^*}(h)$. By Corollary 1, for any such h the COS-POMDP also has value $V_{\pi'}(h)$, meaning π^* is not actually optimal for the COS-POMDP; this is a contradiction. □

6.5 Hierarchical Planning

Despite the optimality-preserving reduction of state space in a COS-POMDP, directly planning over the primitive move actions is not scalable to practical domains even for state-of-the-art online POMDP solvers (Silver & Veness, 2010). This is especially the case when

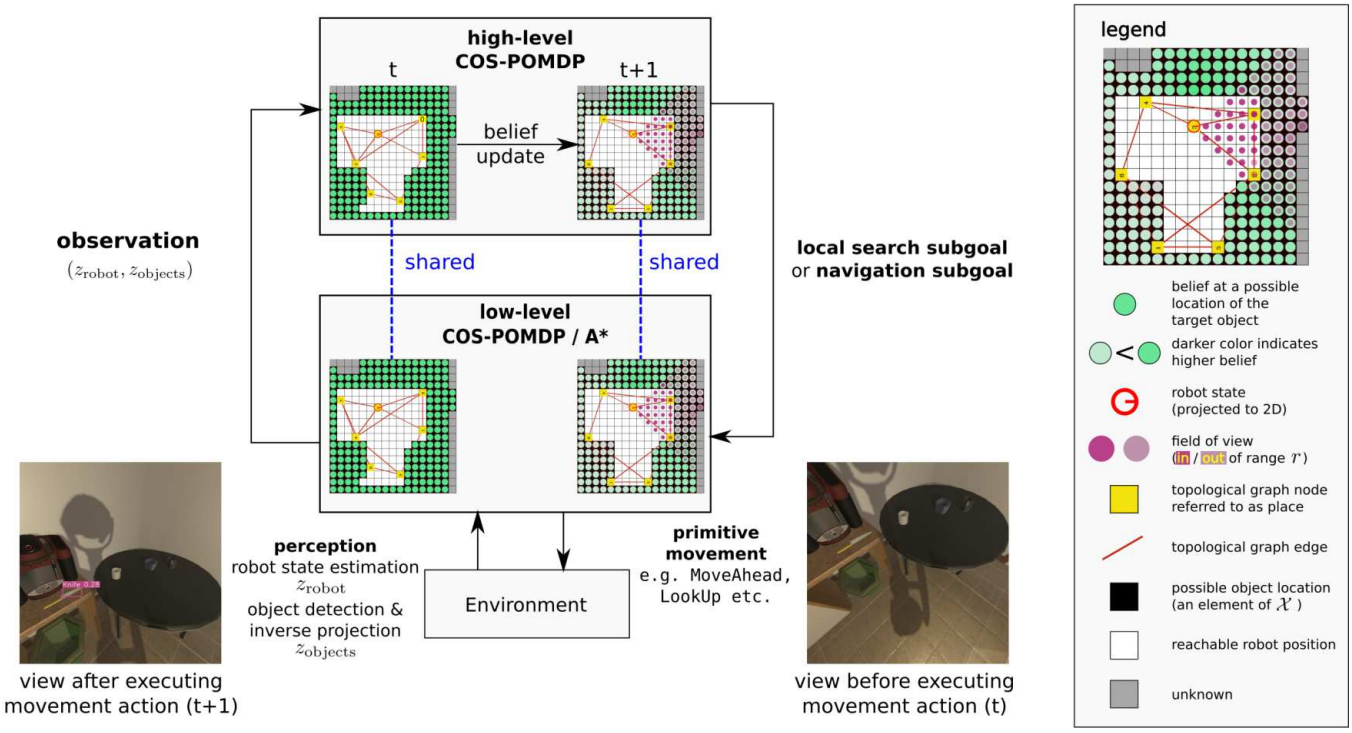


Figure 6.2: **Illustration of the Hierarchical Planning Algorithm.** A high-level COS-POMDP plans subgoals that are fed to a low-level planner to produce low-level actions. The belief state is shared across the levels. Both levels plan with updated beliefs at every timestep.

in-place rotation actions are considered, since identical viewpoints may be repeatedly visited at different depth levels in the search tree, limiting the size of the search region considered during planning. At the same time, however, planning POMDP actions at the low level has the benefit of controlling fine-grained movements, allowing goal-directed behavior to emerge automatically at this level. Therefore, we seek an algorithm that can reason about both searching over a large region as well as careful search in the area around the robot. This is practical because typical mobile robots can be controlled both at the low level of motor velocities and the high level of navigation goals (Zheng, 2021; Macenski et al., 2020).

Hence, we propose a hierarchical planning algorithm to apply COS-POMDPs in realistic domains. The algorithm is presented in Algorithm 5 and illustrated in Fig 7.2. To enable the planning of searching over a large region, we first generate a topological graph, where nodes are places accessible by the robot, and edges indicate navigability between places (Zheng & Pronobis, 2019). This is done by the SampleTopoGraph procedure (Appendix 6.8.2). In this procedure, the nodes are sampled based on the robot's current belief in the target location b^t_{target}, and edges are added such that the graph is connected and every node

Algorithm 5: OnlineHierarchicalPlanning

Input: $P = (\mathcal{X}, \mathcal{C}, \mathcal{D}, s^{\text{init}}_{\text{robot}}, \mathcal{S}_{\text{robot}}, O_{\text{robot}}, \mathcal{A}_m, T_m)$.
Parameter: maximum number of steps T_{max}.
Output: A solution $a_{1:T}$ (Problem 1).
$b^1_{\text{target}}(x_{\text{target}}) \leftarrow$ Uniform($\mathcal{X}$);
$b^1_{\text{robot}}(s_{\text{robot}}) \leftarrow \mathbf{1}(s_{\text{robot}} = s^{\text{init}}_{\text{robot}})$;
$b^1 \leftarrow (b^1_{\text{target}}, b^1_{\text{robot}})$;
$t \leftarrow 1$;
while $t \leq T_{\text{max}}$ and $a_{t-1} \neq$ Done **do**
  $(\mathcal{V}, \mathcal{E}) \leftarrow$ SampleTopoGraph($\mathcal{X}, \mathcal{S}_{\text{robot}}, b^t_{\text{target}}$);
  $P_{\text{H}} \leftarrow$ HighLevelCOSPOMDP($P, \mathcal{V}, \mathcal{E}, b^t$);
  subgoal $\leftarrow$ plan POMDP online for P_{H};
  if subgoal is *navigate to a node in* $\mathcal{V}$ **then**
    $s_{\text{robot}} \leftarrow \text{argmax}_{s_{\text{robot}}} b_{\text{robot}}(s_{\text{robot}})$;
    $a_t \leftarrow$ A*(subgoal, $s_{\text{robot}}, \mathcal{A}_m, T_m$);
  else if subgoal is *search locally* **then**
    $P_{\text{L}} \leftarrow$ LowLevelCOSPOMDP(P, b^t);
    $a_t \leftarrow$ plan POMDP online for P_{L};
  else if subgoal is *Done* **then**
    $a_t \leftarrow$ Done
  end
  $z_t \leftarrow$ execute a_t and receive observation;
  $b^{t+1} \leftarrow$ BeliefUpdate(b^t, a_t, z_t);
  $t \leftarrow t + 1$;
end

has an out-degree within a given range, which affects the branching factor for planning. An example output is illustrated in Fig 7.2.

Then, a high-level COS-POMDP P_{H} is instantiated. The state and observation spaces, the observation model, and the reward model, are as defined in Sec 6.4.1. The move action set and the corresponding transition model are defined according to the generated topological graph. Each move action represents a subgoal of *navigating* to another place, or the subgoal of *searching locally* at the current place. Both types of subgoals can still be understood as viewpoint-changing actions, except the latter keeps the viewpoint at the same location. For the transition model $T(s', g, s)$ where g represents the subgoal, the resulting viewpoint (*i.e.*, $s'_{\text{robot}} \in s'$) after completing a subgoal is located at the destination of the subgoal with orientation facing the target object location ($x_{\text{target}} \in s$). The Done action is also included as a dummy subgoal to match the definition of the COS-POMDP action space (Sec 6.4.1).

At each timestep, a subgoal is planned using an online POMDP planner, and a low-level planner is instantiated corresponding to the subgoal. This low-level planner then plans to

output an action a_t from the action set $\mathcal{A} = \mathcal{A}_m \cup \{\texttt{Done}\}$, which is used for execution. In our implementation, for *navigation* subgoals, an A* planner is used, and for *searching locally*, a low-level COS-POMDP P_{L} is instantiated with the primitive movements $\mathcal{A}_m$ in its action space. (We use PO-UCT (Silver & Veness, 2010) as the online POMDP solver in our experiments.)

Upon executing the low-level action a_t, the robot receives an observation $z_t \in \mathcal{Z}$ from its on-board perception modules for robot state estimation and object detection. This observation is used to update the belief of the high-level COS-POMDP, which is shared with the low-level COS-POMDP.

Finally, the process starts over from the first step of sampling a topological graph. If the high-level COS-POMDP plans a new subgoal different the current one, then the low-level planner is re-instantiated.

This algorithm plans actions for execution in an online, closed-loop fashion, allowing reasoning about viewpoint changes both at the level of places in a topological graph as well as fine-grained movements.

6.6 Evaluation

6.6.1 Experimental Setup

We test the following hypotheses through our experiments: (1) Leveraging correlational information with easier-to-detect objects can benefit the search for hard-to-detect objects; (2) Optimizing over an action sequence improves performance compared to greedily choosing the next-best view.

AI2-THOR

We conduct experiments in AI2-THOR (Kolve et al., 2017), a realistic simulator of in-household rooms. It has a total of 120 scenes divided evenly into four room types: *Bathroom*, *Bedroom*, *Kitchen*, and *Living room*. For each room type, we use the first 20 scenes for training a vision-based object detector and learning object correlation models (used in some experiments), and the last 10 for evaluating performance.

The robot can take primitive move actions from the set:

$\{\texttt{MoveAhead}, \texttt{RotateLeft}, \texttt{RotateRight}, \texttt{LookUp}, \texttt{LookDown}\}$.

`MoveAhead` moves the robot forward by 0.25m. `RotateLeft`, `RotateRight` rotate the robot in place by 45°. `LookUp`, `LookDown` tilt the camera up or down by 30°. The transition function of the robot's viewpoint when taking primitive move actions is deterministic. All methods (Sec 6.6.1) receive observations of the robot's viewpoint without noise, that is $O_{\text{robot}}(z_{\text{robot}}, s_{\text{robot}}) = \mathbf{1}(z_{\text{robot}} = s_{\text{robot}})$. To be successful, when the robot takes `Done`, the

robot must be within a Euclidean distance of 1.0m from the target object while the target object is visible in the camera frame. The maximum number of steps T_{max} is 100.

Object Detector

Unlike previous work in object search evaluated using a ground truth object detector (Qiu et al., 2020) or detectors with synthetic noise and detection ranges (Zeng et al., 2020), we use a vision-based object detector, YOLOv5 (Jocher et al., 2020), since it is more realistic and suitable for our motivation. We collect training data by randomly placing the agent in the training scenes. Table 6.2 and Table 6.3 contain detection statistics of the target objects and correlated objects in validation scenes, respectively. The pixel coordinates within the bounding boxes returned by YOLOv5 are downsampled and inverse projected to positions in the 3D world frame, using the provided depth image.

Detection Model. Vision detectors can sometimes detect small objects from far away. Therefore, we consider a line-of-sight detection model with a limited field of view angle to enable POMDP planning:

$$
\begin{aligned}
D(z_i, x_i, s_{\text{robot}}) &= \Pr(z_i | x_i, s_{\text{robot}}) \\
&= \begin{cases}
1.0 - \text{TP} & s_i \in \mathcal{V}(s_{\text{robot}}) \wedge z_i = \texttt{null} \\
\delta\text{FP}/|\mathcal{V}_E(r)| & s_i \in \mathcal{V}(s_{\text{robot}}) \wedge \|z_i - x_i\| > 3\sigma \\
\delta\mathcal{N}(z_i; x_i, \sigma^2) & s_i \in \mathcal{V}(s_{\text{robot}}) \wedge \|z_i - x_i\| \leq 3\sigma \\
1.0 - \text{FP} & s_i \notin \mathcal{V}(s_{\text{robot}}) \wedge z_i = \texttt{null} \\
\delta\text{FP}/|\mathcal{V}_E(r)| & s_i \notin \mathcal{V}(s_{\text{robot}}) \wedge z_i \neq \texttt{null}
\end{cases}
\end{aligned}
$$

This detection model is parameterized by: TP, the true positive rate; FP, the false positive rate; r, the average distance between the robot and the object for true positive detections; σ, the width of a small region around the true object location where a detection made within that region, though not exactly accurate, is still accepted as a true positive detection. We set $\sigma = 0.5$m. The notation $\mathcal{N}(\cdot; x_i, \sigma^2)$ denotes a Gaussian distribution with mean x_i and covariance $\sigma^2\mathbf{I}$. The $\mathcal{V}(s_{\text{robot}})$ denotes the line-of-sight field of view with a 90° angle. The $\mathcal{V}_E(r)$ denotes the region inside the field of view that is within distance r from the robot. The weight $\delta = 1$ if the detection is within $\mathcal{V}_E(r)$, and otherwise $\delta = \exp(-\|z_i - s_{\text{robot}}\| - r)^2$.

Target Objects

The list of target object classes and other correlated classes for each room type is listed below (with no particular order). For detection statistics, please refer to Table 6.1 and Table 6.3 (Appendix 6.8.3).

- *Bathroom* - Targets: `Fauct`, `Candle`, `ScrubBrush`; Correlated objects: `ToiletPaperHanger`, `Towel`, `Mirror`, `Toilet`, `SoapBar`.

- *Bedroom* - Targets: `AlarmClock`, `Pillow`, `CD`; Correlated objects: `Laptop`, `DeskLamp`, `Mirror`, `LightSwitch`, `Bed`.
- *Kitchen* - Targets are `Bowl`, `Knife`, `PepperShaker`; Correlated classes are: `Lettuce`, `LightSwitch`, `Microwave`, `Plate`, `StoveKnob`
- *Living room* - Targets are `CreditCard`, `RemoteControl`, `Television`; Correlated classes are: `LightSwitch`, `Pillow`, `HousePlant`, `Laptop`, `FloorLamp`, `Painting`.

Correlation Model

We consider a binary correlation model that takes into account whether the correlated object and the target are close or far. Specifically, we define:

$$C(x_{\text{target}}, x_i) = \Pr(x_i | x_{\text{target}}) \tag{6.7}$$

$$= \begin{cases} 1 & \text{Close}(i, \text{target}) \wedge \|x_i - x_{\text{target}}\| < d(i, \text{target}) \\ 0 & \text{Close}(i, \text{target}) \wedge \|x_i - x_{\text{target}}\| \geq d(i, \text{target}) \\ 1 & \text{Far}(i, \text{target}) \wedge \|x_i - x_{\text{target}}\| > d(i, \text{target}) \\ 0 & \text{Far}(i, \text{target}) \wedge \|x_i - x_{\text{target}}\| \leq d(i, \text{target}) \end{cases} \tag{6.8}$$

where $\text{Close}(\cdot, \cdot)$ and $\text{Far}(\cdot, \cdot)$ are class-level predicates, $\|\cdot\|$ denotes the Euclidean distance, and $d(\cdot, \cdot)$ is the expected distance between the two objects. This model is applicable between arbitrary object classes and can be estimated based on instances of object classes. In Sec. 6.6.2, we conduct an ablation study where $d(\cdot, \text{target})$ is estimated under different scenarios: **accurate**: based on object ground truth locations in the deployed scene; **learned** (lrn): based on instances in training scenes; **wrong** (wrg): same as accurate except we flip the close/far relationship between the objects so that they do not match the scene.

Implementation Detail of COS-POMDP

Objects exist in 3D space in AI2-THOR scenes, and the robot can rotate its camera both horizontally and vertically. Our implementation of COS-POMDP allows for search in such a setting by estimating, in the belief over target locations, both the 2D position of the target as well as the height of the target. Since the robot can tilt only its camera within a small range of angles, we consider a discrete set of possible height values, `Above`, `Below`, and `Same`, which indicates the object is above, below, or at the same level with respect to the camera's current tilt angle. Our implementation is based on the `pomdp_py` (Zheng & Tellex, 2020) library.

Evaluation Metric

We use three metrics: (1) success weighted by inverse path length (SPL) (Anderson et al., 2018); (2) success rate (SR) and (3) dicounted cumulative rewards (DR). The SPL of each

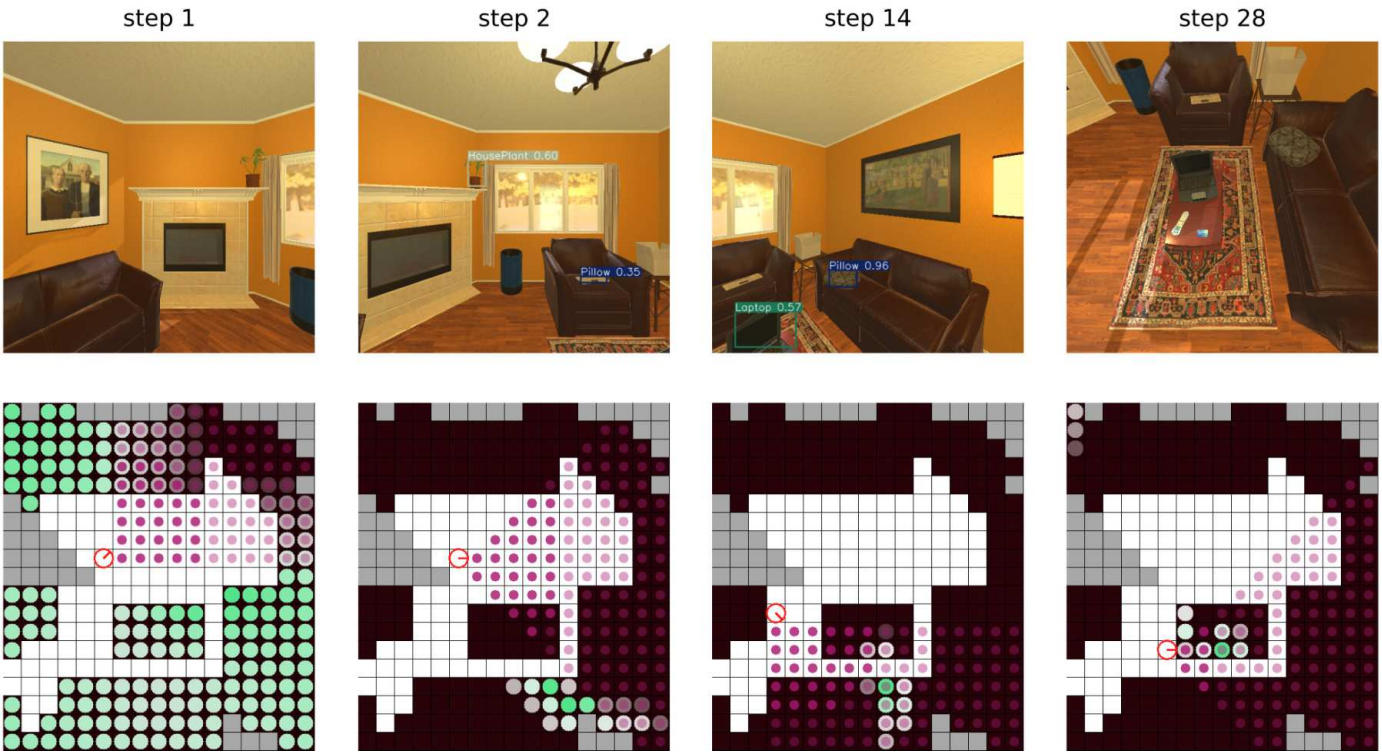


Figure 6.3: **Example Sequence.** Top: first-person view with object detection bounding boxes. Bottom: Visualization of belief state corresponding to each view. See Fig 7.2 for the legend of the belief state visualization. Our method (COS-POMDP) successfully finds a `CreditCard` in a living room scene, leveraging the detection of other objects such as `FloorLamp` and `Laptop`. For more examples, please refer to the video at https://youtu.be/wd1tmD0mckY.

search trial is defined as $SPL = S \cdot \ell / \max(p, \ell)$ where S is the binary success outcome of the search, ℓ is the shortest path between the robot and the target, and p is the actual search path. The SPL measures the search performance by taking into account both the success and efficiency of the search. It is a difficult metric because ℓ uses information about the true object location. However, it does not penalize excessive rotations (Batra et al., 2020). Therefore, we also include discounted cumulative rewards ($\gamma = 0.95$) which takes such actions into account.

Baselines

Baselines are defined in the caption of Table 6.1. Note that for **Greedy-NBV**, based on (Zeng et al., 2020), a weighted particle belief is used to maintain the belief over the joint state over all object locations. During planning, the agent selects the next best viewpoint to navigate towards based on a cost function that considers both navigation distance and

Figure 6.4: Visualization of robot trajectory produced by different methods for the example shown in Fig. 6.3. Each gray circle represents the position of a viewpoint, and each black line segment indicates the orientation of the robot's camera at a viewpoint. The path traversed during the search is shown in blue.

the probability of detecting any object. This provides a baseline that is in contrast to the sequential decision-making paradigm considered by COS-POMDPs and the modeling of only robot and target states. [1]

6.6.2 Results and Discussions

Our main results are shown in Table 6.1. The performance of **COS-POMDP** is the most consistent compared to other baselines, with **COS-POMDP** performing either the best or the second best in the four room types. The performance is broken down by target classes in Table 6.2. **Greedy-NBV** performs well in *Bedroom*; it appears to experience less inaccuracy in the particle-based belief over all objects as a result of particle reinvigoration in bedroom compared to the other room types. **COS-POMDP** appears to be the most robust when the target object has significant noise of being correctly detected, including `ScrubBrush`, `CreditCard`, `Candle` `RemoteControl`, `Knife`, and `CD`. An example search trial of **COS-POMDP** for `CreditCard` is shown in Fig 6.3 and the search paths of the methods under comparison are visualized in Fig 6.4. For target objects with a true positive detection rate below 40%, **COS-POMDP** improves the POMDP baseline that ignores correlational information by 70% in terms of the SPL metric, and is more than 1.7 times better than the greedy baseline. Indeed, when the target object is reliably detectable, such as `Television`, the ability to detect multiple other objects may actually hurt performance, compared to **Target-POMDP**, due to the noise from detecting those other objects and the

1. I attempted a comparison with deep reinforcement learning methods (Wortsman et al., 2019; Qiu et al., 2020), yet I was blocked by the fact that their codebases were developed for earlier versions of AI2-THOR (v1.0) and use different configurations (*e.g.* rotating at 90 degrees instead of 45 degrees). The trained models performed poorly on the newer version I was using (v3.3.4) due to backwards incompatibility.

influence on search behavior. These results demonstrate that COS-POMDPs can be applied to search for hard-to-detect objects leveraging the more reliable detection of correlated objects.

Ablation Studies

We also conduct two ablation studies. First, we equip **COS-POMDP** with a groundtruth object detector, as done in (Qiu et al., 2020), henceforth called **COS-POMDP (gt)**. This shows the performance when the detections of both the target and correlated objects involve no noise at all. We observe better or competitive performance from using groundtruth detectors across all metrics in all room types.

Additionally, we use correlations obtained by learning from data (**COS-POMDP (lrn)**) as well as incorrect correlation information that is the reverse of the correct one (**COS-POMDP (wrg)**). Indeed, using accurate correlations provides the most benefit, while correlations learned through this simple method could offer benefit compared to using incorrect correlations in some cases (*Bathroom* and *Bedroom*), but can also backfire and hurt performance in other others. Therefore, properly learning correlation is important, while leveraging a reliable source of information, for example, from a human at the scene, may offer the most benefit.

6.7 Summary

In this chapter, we formulated the problem of correlational object search and proposed COS-POMDP, a POMDP-based approach to model this problem. Our quantitative evaluation, conducted in AI2-THOR (Kolve et al., 2017) using the YOLOv5 (Redmon et al., 2016; Jocher et al., 2020) detector, demonstrates the benefit of our approach in exploiting the correlational information with easier-to-detect objects to find hard-to-detect objects. Future work directions include studying the search behavior given different kinds of learned correlation models as well as in more complex settings that involve *e.g.*, container opening and dynamic objects.

Method	Bathroom			Bedroom			Kitchen			Living room		
	SPL (%)	DR	SR (%)	SPL (%)	DR	SR (%)	SPL (%)	DR	SR (%)	SPL (%)	DR	SR (%)
Random	0.00 (0.00)	-82.75 (3.43)	0.00	0.00 (0.00)	-85.27 (3.82)	0.00	6.90 (9.81)	-68.51 (15.61)	6.90	0.00 (0.00)	-82.37 (3.62)	0.00
Greedy-NBV	14.85 (9.40)	-18.86 (12.14)	35.71	**31.10 (17.86)**	**-6.97 (14.20)**	**40.91**	12.03 (9.01)	-17.16 (12.85)	32.14	7.13 (7.11)	-21.41 (8.21)	20.00
Target-POMDP	24.17 (12.03)	**-2.60 (17.02)**	**66.67**	14.70 (12.86)	-26.74 (13.27)	31.58	14.82 (9.22)	-20.06 (13.85)	37.04	**29.23 (15.34)**	-30.65 (13.60)	48.00
COS-POMDP	**30.03 (13.59)**	-14.92 (12.76)	56.00	28.54 (17.63)	-16.02 (14.03)	40.00	**20.95 (13.10)**	**-4.67 (14.71)**	**44.00**	27.76 (15.21)	**-12.32 (15.71)**	**48.15**
COS-POMDP (gt)	**33.38 (13.92)**	-11.69 (13.24)	62.96	**39.22 (19.56)**	-13.50 (17.28)	**56.25**	**36.92 (14.33)**	**-2.92 (16.46)**	**64.00**	**35.71 (16.05)**	**-9.31 (13.09)**	**62.50**
COS-POMDP (lrn)	19.77 (12.07)	-21.53 (13.13)	45.83	16.43 (14.53)	-33.73 (11.05)	23.81	6.29 (6.93)	-32.72 (13.62)	17.86	14.76 (11.41)	-43.09 (13.57)	25.00
COS-POMDP (wrg)	10.83 (7.79)	-19.00 (10.21)	28.00	14.54 (14.29)	-32.54 (14.87)	27.78	8.80 (7.38)	-20.49 (10.63)	25.93	**29.34 (16.10)**	-16.15 (11.96)	**54.17**

Table 6.1: **Main and Ablation Study Results.** Unless otherwise specified, all methods use the YOLOv5 (Jocher et al., 2020) vision detector and are given accurate correlational information. **Target-POMDP** uses the hierarchical planning except only the target object is detectable. **Greedy-NBV** is a next-best view approach based on (Zeng et al., 2020). **Random** chooses actions uniformly at random. The highest value of each metric per room type is bolded. Parentheses contain 95% confidence interval. Ablation study results are bolded if it outperforms the best result from the main evaluation. Metrics are success weighted by inverse path length (SPL) (Anderson et al., 2018), discounted cumulative reward (DR), and success rate (SR). **COS-POMDP** is more consistent, performing either the best or the second best across all room types and metrics.

					Greedy-NBV			Target-POMDP			COS-POMDP		
Room Type	Target Class	TP	FP	r (m)	SPL (%)	DR	SR (%)	SPL (%)	DR	SR (%)	SPL (%)	DR	SR (%)
Bathroom	Faucet	56.1	8.0	2.16	31.45 (20.79)	6.11 (21.02)	**77.78**	**40.21 (27.65)**	**13.97 (30.40)**	75.00	24.59 (29.84)	-28.23 (26.54)	50.00
	Candle	29.4	2.4	1.81	12.52 (20.12)	-22.81 (20.80)	22.22	18.63 (14.51)	-6.61 (33.50)	**75.00**	**33.89 (21.83)**	**-2.94 (19.08)**	66.67
	ScrubBrush	64.3	9.9	1.71	2.00 (4.52)	-37.79 (17.36)	10.00	7.38 (13.80)	-22.68 (34.63)	40.00	**31.12 (32.00)**	**-15.08 (28.68)**	**50.00**
Bedroom	AlarmClock	79.6	7.4	2.77	**48.10 (43.23)**	**-0.80 (26.52)**	**57.14**	14.64 (46.61)	-17.37 (44.62)	25.00	35.07 (33.97)	-15.52 (23.95)	44.44
	Pillow	88.3	5.2	2.43	**30.04 (49.28)**	**-13.59 (38.13)**	33.33	5.16 (14.32)	-29.49 (30.07)	**40.00**	21.02 (90.44)	-14.85 (98.69)	33.33
	CD	48.6	4.5	1.70	18.59 (21.89)	**-7.36 (26.46)**	33.33	19.49 (22.66)	-29.12 (21.97)	30.00	**24.01 (28.21)**	-17.03 (24.75)	**37.50**
Kitchen	Bowl	60.6	11.5	1.75	**19.88 (26.57)**	-15.76 (32.76)	33.33	16.33 (16.00)	-10.06 (27.39)	**55.56**	16.27 (20.22)	**-0.19 (37.01)**	42.86
	Knife	37.7	8.7	1.68	8.22 (12.85)	-17.74 (26.85)	33.33	5.13 (11.84)	-37.99 (17.17)	11.11	**23.97 (25.58)**	**-2.59 (25.33)**	**50.00**
	PepperShaker	38.1	9.4	1.43	8.39 (10.53)	-17.90 (17.39)	30.00	**22.99 (23.22)**	-12.14 (31.40)	**44.44**	21.26 (30.90)	**-11.17 (30.04)**	37.50
Living room	Television	85.3	5.2	2.59	8.98 (18.36)	-22.86 (13.31)	20.00	**59.56 (25.42)**	**-6.50 (19.73)**	**88.89**	44.53 (34.78)	-10.79 (31.79)	55.56
	RemoteControl	69.6	4.5	1.93	9.24 (13.99)	-13.21 (20.44)	30.00	26.67 (35.38)	-29.18 (20.47)	42.86	**33.49 (31.89)**	**8.94 (27.67)**	**66.67**
	CreditCard	42.9	4.3	1.48	3.18 (7.19)	**-28.15 (11.70)**	10.00	0.91 (2.09)	-55.95 (22.44)	11.11	**5.26 (8.08)**	-35.12 (24.57)	**22.22**

Table 6.2: **Detection Statistics and Object Search Results Grouped by Target Classes.** TP: true positive rate (%); FP: false positive rate (%); r: average distance to the true positive detections (m). We estimated these values by running the vision detector at 30 random camera poses per validation scene. Target objects are sorted by average detection range. Parentheses contain 95% confidence interval. Metrics are success weighted by inverse path length (SPL) (Anderson et al., 2018), discounted cumulative reward (DR), and success rate (SR). **COS-POMDP** performs more robustly for hard-to-detect objects, such as `ScrubBrush`, `CD`, `Candle`, `Knife`, and `CreditCard`.

6.8 Appendix

6.8.1 Proof of Theorem 1

Theorem 1. Given any policy $\pi : h_t \to a$, the distribution of histories is identical between the COS-POMDP and the F-POMDP.

Proof. We prove this by induction. When $t = 1$, the statement is true because both histories are empty. The inductive hypothesis assumes that the distributions $\Pr(h_t)$ is the same for the two POMDPs at $t \geq 1$. Then, by definition, $\Pr(h_{t+1}) = \Pr(h_t, a_t, z_t) = \Pr(z_t|h_t, a_t)\Pr(a_t|h_t)\Pr(h_t)$. Since $\Pr(a_t|h_t)$ is the same under the given π, we can conclude $\Pr(h_{t+1})$ is identical if the two POMDPs have the same $\Pr(z_t|h_t, a_t)$. We show that this is true as follows.

First, we sum out the state s_t at time t:

$$\Pr(z_t|h_t, a_t) = \sum_{s_t} \Pr(s_t, z_t|h_t, a_t) \tag{6.9}$$

By definition of conditional probability,

$$= \sum_{s_t} \Pr(z_t|s_t, h_t, a_t)\Pr(s_t|h_t, a_t) \tag{6.10}$$

Since s_t is does not depend on a_t (which affects s_{t+1}),

$$= \sum_{s_t} \Pr(z_t|s_t, h_t, a_t)\Pr(s_t|h_t) \tag{6.11}$$

Suppose we are deriving this distribution for COS-POMDP, denoted as $\Pr_{\text{COS-POMDP}}(z_t|h_t, a_t)$. Then, by definition, the state $s_t = (x_{\text{target}}, s_{\text{robot}})$. Therefore, we can write:

$$\begin{aligned} &\Pr_{\text{COS-POMDP}}(z_t|h_t, a_t) \\ &= \sum_{x_{\text{target}}, s_{\text{robot}}} \Pr(z_t|x_{\text{target}}, s_{\text{robot}}, h_t, a_t) \\ &\qquad \times \Pr(x_{\text{target}}, s_{\text{robot}}|h_t) \end{aligned} \tag{6.12}$$

Summing out $x_1, \ldots, x_n$,

$$\begin{aligned} &= \sum_{x_{\text{target}}, s_{\text{robot}}} \sum_{x_1, \cdots, x_n} \Pr(x_1, \ldots, x_n, z_t|x_{\text{target}}, s_{\text{robot}}, h_t, a_t) \\ &\qquad \times \Pr(x_{\text{target}}, s_{\text{robot}}|h_t) \end{aligned} \tag{6.13}$$

Merging sum,

$$= \sum_{\substack{x_1,\cdots,x_n,\\ x_{\text{target}},s_{\text{robot}}}} \Pr(x_1,\ldots,x_n,z_t|x_{\text{target}},s_{\text{robot}},h_t,a_t) \times \Pr(x_{\text{target}},s_{\text{robot}}|h_t) \tag{6.14}$$

By the definition of conditional probability,

$$= \sum_{\substack{x_1,\cdots,x_n,\\ x_{\text{target}},s_{\text{robot}}}} \Pr(z_t|x_1,\ldots,x_n,x_{\text{target}},s_{\text{robot}},h_t,a_t) \times \Pr(x_1,\ldots,x_n|x_{\text{target}},s_{\text{robot}},h_t,a_t) \times \Pr(x_{\text{target}},s_{\text{robot}}|h_t) \tag{6.15}$$

Again, because the object locations are independent of a_t,

$$= \sum_{\substack{x_1,\cdots,x_n,\\ x_{\text{target}},s_{\text{robot}}}} \Pr(z_t|x_1,\ldots,x_n,x_{\text{target}},s_{\text{robot}},h_t,a_t) \times \Pr(x_1,\ldots,x_n|x_{\text{target}},s_{\text{robot}},h_t) \times \Pr(x_{\text{target}},s_{\text{robot}}|h_t) \tag{6.16}$$

By the definition of conditional probability again,

$$= \sum_{\substack{x_1,\cdots,x_n,\\ x_{\text{target}},s_{\text{robot}}}} \Pr(z_t|x_1,\ldots,x_n,x_{\text{target}},s_{\text{robot}},h_t,a_t) \times \Pr(x_1,\ldots,x_n,x_{\text{target}},s_{\text{robot}}|h_t) \tag{6.17}$$

Note that $(x_{\text{target}}, s_{\text{robot}}, x_1, \ldots, x_n)$ is a state in F-POMDP. Denote the state space of F-POMDP as $\mathcal{S}_{\text{F}}$. According to Eq (6.11), we can write the above Eq (6.17) as

$$= \sum_{s_t \in \mathcal{S}_{\text{F}}} \Pr(z_t|s_t,h_t,a_t)\Pr(s_t|h_t) \tag{6.18}$$

$$= \Pr_{\text{F-POMDP}}(z_t|h_t,a_t) \tag{6.19}$$

□

6.8.2 Auxiliary Procedures

Algorithm 6 is the pseudocode of the SampleTopoGraph algorithm, implemented for our experiments in AI2-THOR. We set $M = 10$, $d_{\text{sep}} = 1.0\text{m}$, $\zeta_{\text{min}} = 3$, $\zeta_{\text{max}} = 5$. In our implementation, the topological graph is resampled only if the cumulative belief captured by the nodes in the current topological graph, $\sum_{s_{\text{robot}} \in \mathcal{V}} p(s_{\text{robot}})$, is below 50%. Otherwise, the same topological graph will be returned. Note that depending on the application domain, a different algorithm for forming the topological graph may be used in the place of SampleTopoGraph in the OnlineHierarchicalPlanning algorithm (Algorithm 5).

Algorithm 6: SampleTopoGraph

Input: $\mathcal{X}, \mathcal{S}_{\text{robot}}, b_{\text{target}}$

Parameter: maximum number of nodes M, minimum separation between nodes d_{sep}, minimum and maximum out-degrees $\zeta_{\text{min}}, \zeta_{\text{max}}$

Output: A topological graph $(\mathcal{V}, \mathcal{E})$

// Obtain mapping from s_{robot} to a set of closest locations

foreach $x \in \mathcal{X}$ **do**

  $s_{\text{robot}}^{\text{closest}} \leftarrow \text{argmin}_{s_r \in \mathcal{S}_{\text{robot}}} \|s_r.\text{pos} - x\|$;

  $U(s_{\text{robot}}^{\text{closest}}) \leftarrow U(s_{\text{robot}}^{\text{closest}}) \cup \{x\}$;

end

// Construct probability distribution over $\mathcal{S}_{\text{robot}}$ using b_{target}

foreach $s_{\text{robot}} \in \mathcal{S}_{\text{robot}}$ **do**

  $p(s_{\text{robot}}) \leftarrow \sum_{x \in U(s_{\text{robot}})} b_{\text{target}}(x)$

end

// Construct nodes and edges

$\mathcal{V} \leftarrow$ sample $\leq M$ nodes from $\mathcal{S}_{\text{robot}}$ according to p such that any pair of nodes has a minimum distance of d_{sep};

$\mathcal{E} \leftarrow$ add edges between nodes in $\mathcal{V}$ so that the graph is connected and each node has an out-degree between ζ_{min} and ζ_{max};

return $(\mathcal{V}, \mathcal{E})$

6.8.3 Detection Statistics for Correlated Object Classes

Table 6.3 shows the detection statistics for correlated object classes. The detection statistics of target object classes can be found in Table 6.2.

Room Type	Correlated Object Class	TP	FP	r (m)
Bathroom	Mirror	76.9	3.7	2.10
	ToiletPaperHanger	84.4	1.5	1.96
	Towel	79.4	2.7	1.88
	Toilet	86.3	3.5	1.81
	SoapBar	73.2	1.8	1.53
Bedroom	DeskLamp	89.5	2.6	2.41
	Bed	63.5	0.6	2.39
	Mirror	86.0	0.6	2.27
	LightSwitch	76.3	2.8	2.26
	Laptop	75.9	1.2	2.19
Kitchen	LightSwitch	90.0	3.9	2.57
	Microwave	75.3	5.6	2.31
	StoveKnob	82.8	5.6	2.00
	Lettuce	98.6	0.3	1.98
	Plate	60.6	3.2	1.90
Living room	FloorLamp	71.7	5.1	3.44
	Painting	85.2	4.0	3.18
	LightSwitch	80.6	1.5	3.10
	HousePlant	82.9	3.9	3.00
	Pillow	67.4	2.8	2.84
	Laptop	66.3	2.6	2.24

Table 6.3: **Detection Statistics for Correlated Object Classes.** TP: true positive rate (%); FP: false positive rate (%); r: average distance to the true positive detections (m). We estimated these values by running the vision detector at 30 random camera poses per validation scene. The correlated object classes for each room type are sorted by average detection range.

6.8.4 Evaluation on a Toy Domain: HallwaySearch

HallwaySearch

This domain is designed to be minimal and allows the use of an offline POMDP solver until convergence. We evaluate COS-POMDPs on this domain to investigate the influence that detecting spatially correlated objects has on the expected return. In HallwaySearch, there are two objects in a hallway: a target object and a spatially correlated object, both at unknown locations sampled from a joint distribution. The robot has two detectors, one for each object, that return a binary observation indicating successful or `null` detection of the object. The detector for the target object has a small range that only returns a successful detection if the robot is directly on top of the object. The detector for the spatially correlated object has a larger range that can also return a successful detection from the two adjacent locations. Both detectors are noisy, and false negatives and false positives may occur.

Baselines

- *Corr*. Solve the COS-POMDP.
- *Target*. Rather than solving the COS-POMDP, we solve a minimal POMDP for the object search task that ignores the correlational information and assumes the object locations are independent. As a result, the robot uses the target object detector, but never needs its other detectors.

Experimental Procedure and Results

For HallwaySearch, we conduct two sets of experiments. In the first experiment, the hallway length varies from 4 to 8 while the robot has a perfect detector for both objects. In the second experiment, the hallway length is fixed at 4 and the noise of the target detector varies, specified by pairs of (false positive, false negative) rates that range from zero to 10%. We report both the optimal POMDP value as given by SARSOP and the approximate discounted cumulative reward calculated over 100 trials.

The results for the HallwaySearch domain are shown in Figure 6.5. We observe that considering the correlational information (green curves) leads to greater or equal optimal POMDP value than not considering it (red curves) for all experimental settings. This suggests that the optimal COS-POMDP policy makes use of the detector for the correlated object, improving the expected returns. The actual return follow a similar pattern. In this domain, the impact due to detector noise is significant, and using the correlational information leads to more robust performance. The variance in the estimates of the actual returns (left plots) is due to the stochasticity of the observation model and object locations. Overall, this experiment supports our first hypothesis: that using correlational information can improve the performance of object search, both in expectation and in empirical returns.

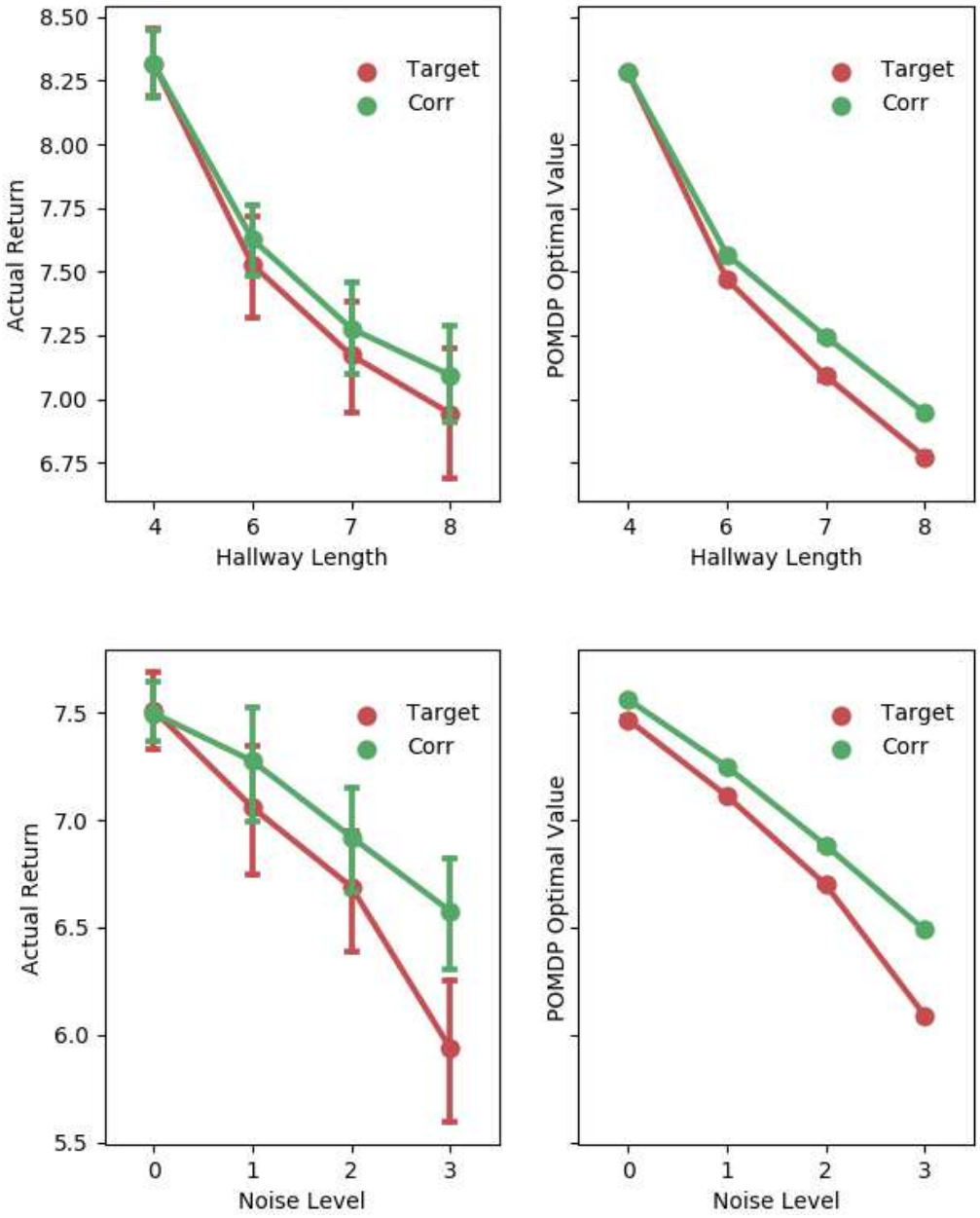


Figure 6.5: Results in the HallwaySearch domain. Top row shows estimated returns (left) and POMDP optimal value (right) as a function of hallway length; bottom row shows estimated returns (left) and POMDP optimal value (right) as a function of detector noise levels. The *Target* baseline (red) does not consider correlational information, and we can see that it always performs worse than *Corr* (green), which does. Thus, this experiment supports our first hypothesis.

CHAPTER 7

Spatial Language Understanding for Object Search

7.1 Motivation - Why Spatial Language?

CONSIDER the scenario in which a tourist is looking for an ice cream truck in an amusement park. She asks a passer-by and gets the reply *the ice cream truck is behind the ticket booth*. The tourist looks at the amusement park map and locates the ticket booth. Then, she is able to infer a region corresponding to that statement and find the ice cream truck, even though the spatial preposition *behind* is inherently ambiguous and subjective to the passer-by. Robots capable of understanding spatial language can leverage prior knowledge possessed by humans to search for objects more efficiently, and interface with humans more naturally. Such capabilities can be useful for applications such as autonomous delivery and search-and-rescue, where the customer or people at the scene communicate with the robot via natural language.

7.1.1 Challenges

This problem is challenging because humans produce diverse spatial language phrases based on their observation of the environment and knowledge of target locations, yet none of these factors are available to the robot. In addition, the robot may operate in a different area than where it was trained. The robot must generalize its ability to understand spatial language across environments.

. Project website: https://h2r.github.io/sloop/

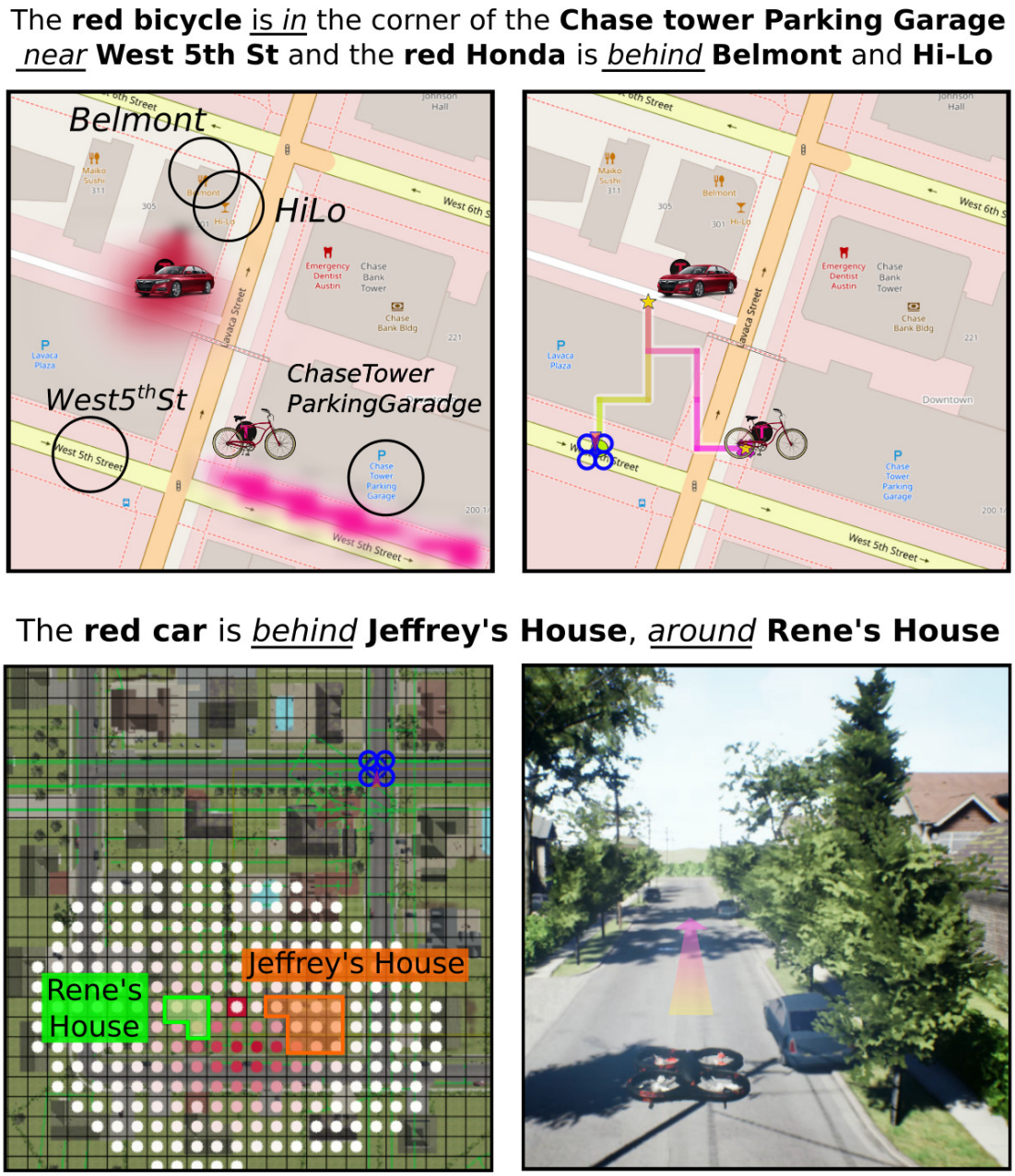


Figure 7.1: Given a spatial language description, a drone with limited field of view must find target objects in a city-scale environment. Top row: example trial from OpenStreetMap (OpenStreetMap contributors, 2017). Bottom row: example trial from AirSim (S. Shah et al., 2017). Left side: belief over target location after incorporating spatial language using the proposed approach. Right side: Screenshot of simulation with search path.

7.1.2 Remark on Previous Work

Prior works on spatial language understanding assume referenced objects already exist in the robot's world model (Tellex et al., 2011; Fasola & Matarić, 2013; Janner et al., 2018) or within the robot's field of view (Blukis, Brukhim, et al., 2018). Works that consider partial observability do not handle ambiguous spatial prepositions (Hemachandra et al., 2015; Wandzel et al., 2019) or assume a robot-centric frame of reference (Bisk et al., 2018; Patki et al., 2020), limiting the ability to understand diverse spatial relations that provide critical disambiguating information, such as *behind the ticket booth*. For downstream tasks, existing works primarily consider spatial language as goal or trajectory specification (A. Vogel & Jurafsky, 2010; Kollar et al., 2010). They require large datasets of spatial language paired with reference paths to learn a policy by end-to-end reward-based learning (Jain et al., 2019; X. Wang et al., 2019) or imitation learning (Blukis, Misra, et al., 2018), which is expensive to acquire for realistic environments (such as urban areas), and generalization to such environments is an ongoing challenge (Blukis, Brukhim, et al., 2018; Bisk et al., 2016; Blukis et al., 2019).

Partially Observable Markov Decision Process (POMDP) (Kaelbling et al., 1998) is a principled decision making framework widely used in the object search literature (Aydemir et al., 2013; Xiao et al., 2019; Zheng et al., 2021a), due to its ability to capture uncertainty in object locations and the robot's perception. Wandzel et al. (2019) proposed Object-Oriented POMDP (OO-POMDP), which factors the state and observation spaces by objects and is designed to model tasks in human environments.

Spatial language is a way of communicating spatial information of objects and their relations using spatial prepositions (e.g. *on*, *between*, *front*) (Hayward & Tarr, 1995). Understanding spatial language for decision making requires the robot to map symbols of the given language description to concepts and structures in the world, a problem referred to as language grounding. Most prior works on spatial language grounding assume a fully observable domain (Tellex et al., 2011; Fasola & Matarić, 2013; Blukis, Brukhim, et al., 2018; A. Vogel & Jurafsky, 2010; Kollar et al., 2010), where the referenced objects have known locations or are within the field of view, and the concern is synthesizing navigation behavior faithful to a given instruction (e.g. *Go to the right side of the rock* (Blukis, Brukhim, et al., 2018)). Recent works aim to map such instructions directly to low-level controls or navigation trajectories leveraging deep reinforcement learning (A. Vogel & Jurafsky, 2010; Jain et al., 2019; Blukis et al., 2020) or imitation learning (Blukis, Brukhim, et al., 2018; X. Wang et al., 2019; Blukis, Misra, et al., 2018), requiring large datasets of instructions paired with demonstrations. In this work, the referenced target objects have unknown locations, and the robot has a limited field of view. We regard the spatial language description as observation and obtain policy through online planning.

Spatial language understanding in partially observable environments is an emerging area of study (Hemachandra et al., 2015; Wandzel et al., 2019; Patki et al., 2020; Thomason et al., 2019). Thomason et al. (2019) propose a domain where the robot, tasked to reach a goal room, has access to a dialogue with an oracle discussing the location of the

goal during execution. Hemachandra et al. (2015) and Patki et al. (2020) infer a distribution over semantic maps for instruction following then plan actions through behavior inference. These instructions are typically FoR-independent or involve only the robot's own FoR. In contrast, we consider language descriptions with FoRs relative to referenced landmarks. Wandzel et al. (2019) propose the Object-Oriented POMDP framework for object search and a proof-of-concept keyword-based model for language understanding in indoor space. Our work handles diverse spatial language using a novel spatial language observation model and focuses on search in cityscale environments. We evaluate our system against a keyword-based baseline similar to the one in (Wandzel et al., 2019).

Cognitive scientists have grouped FoRs into three categories: absolute, intrinsic, and relative (Majid et al., 2004; Shusterman & Li, 2016). Absolute FoRs (e.g. for *north*) are fixed and depend on the agreement between speakers of the same language. Intrinsic FoRs (e.g. for *at the door of the house*) depend only on properties of the referenced object. Relative FoRs (e.g. for *behind the ticket booth*) depend on both properties of the referenced object and the perspective of the observer. In this chapter, spatial descriptions are provided by human observers who may impose relative FoRs or absolute FoRs.

7.2 Contributions

In this work, we make the following contributions:

- We introduce SLOOP (Spatial Language Object-Oriented POMDP), which extends OO-POMDP by considering spatial language as an additional perceptual modality. We derive a probabilistic model to capture the uncertainty of the language through referenced objects and landmarks. This enables the robot to incorporate into its belief state spatial information about the referenced object via belief update.
- We apply SLOOP to object search in city-scale environments given a spatial language description of target locations. Search begins after the initial belief update upon receiving the spatial language. Note that in general, because SLOOP regards spatial language as an observation, the language can be received during task execution.
- We collected a dataset of five city maps from OpenStreetMap (OpenStreetMap contributors, 2017) as well as spatial language descriptions through Amazon Mechanical Turk (AMT).
- To understand ambiguous, context-dependent prepositions (e.g. *behind*), we develop a simple convolutional neural network that infers the latent frame of reference (FoR) given an egocentric synthetic image of the referenced landmark and surrounding context. This FoR prediction model is integrated into the spatial language observation model in SLOOP.
- We evaluate both the FoR prediction model and the object search performance under SLOOP using the collected dataset. Results show that our method leads to search

strategies that find objects faster with higher success rate by exploiting spatial information from language compared to a keyword-based baseline used in prior work (Wandzel et al., 2019). We also report results for varying language complexity and spatial prepositions to discuss advantages and limitations of our approach.

- We demonstrate SLOOP for object search in AirSim (S. Shah et al., 2017), a realistic drone simulator shown in Fig. 7.1, where the drone is tasked to find cars in a neighborhood environment. We also demonstrate SLOOP on a real-robot by integrating it with the Boston Dynamics Spot robot searching for objects in a lab environment.

7.3 Problem Formulation

We are interested in the problem setting similar to the opening scenario in the Motivation - Why Spatial Language?. A robot is tasked to search for N targets in an urban or suburban area represented as a discrete 2D map of landmarks $\mathcal{M}$. The robot is equipped with the map and can detect the targets within the field of view. However, the robot has no knowledge of object locations a priori, and the map size is substantially larger than the sensor's field of view, making brute-force search infeasible. A human with access to the same map and prior knowledge of object locations provides the robot with a natural language description. For example, given the map in Fig. 7.1, one could say *the red car is behind Jeffrey's House, around Rene's House.* The language is assumed to mention some of the target objects and their spatial relationships to some of the known landmarks, yet there is no assumption about how such information is described. To be successful, the robot must incorporate information from spatial language to efficiently search for the target object.

This problem can be formulated as the multi-object search (MOS) task (Wandzel et al., 2019), modeled as an OO-POMDP. The state $s_i = (x_i, y_i)$ is the location for target i, $1 \leq i \leq N$. The robot state $s_r = (x, y, \theta, \mathcal{F})$ consists of the robot's pose (x, y, θ) and a set of found targets $\mathcal{F} \subseteq \{1, \cdots, N\}$. There are three types of actions: MOVE changes the robot pose (possibly stochastically); LOOK processes sensory information within the current field-of-view; FIND(i) marks object i as found. In our implementation of MOS, a LOOK action is automatically taken following every MOVE. Upon taking FIND, the robot receives reward $R_{\text{max}} \gg 0$ if an unfound object is within the field of view, and $R_{\text{min}} \ll 0$ otherwise. Other actions receive $R_{\text{step}} < 0$. The desired policy accounts for the belief over target locations while efficiently exploring the map.

Note that in our evaluation, we use a synthetic detector that returns observations conditioned on the ground truth object locations for belief update during execution. Our POMDP-based framework can easily make use of a realistic observation model instead, for example, based on processing visual data (Xiao et al., 2019; Monsó et al., 2012). Training vision-based object detectors is outside the scope of this chapter. Our focus is on spatial language understanding for planning object search strategies.

In the next section, we introduce SLOOP, with a particular focus on the observation space and observation model. Then, to apply SLOOP to our problem setting, we describe

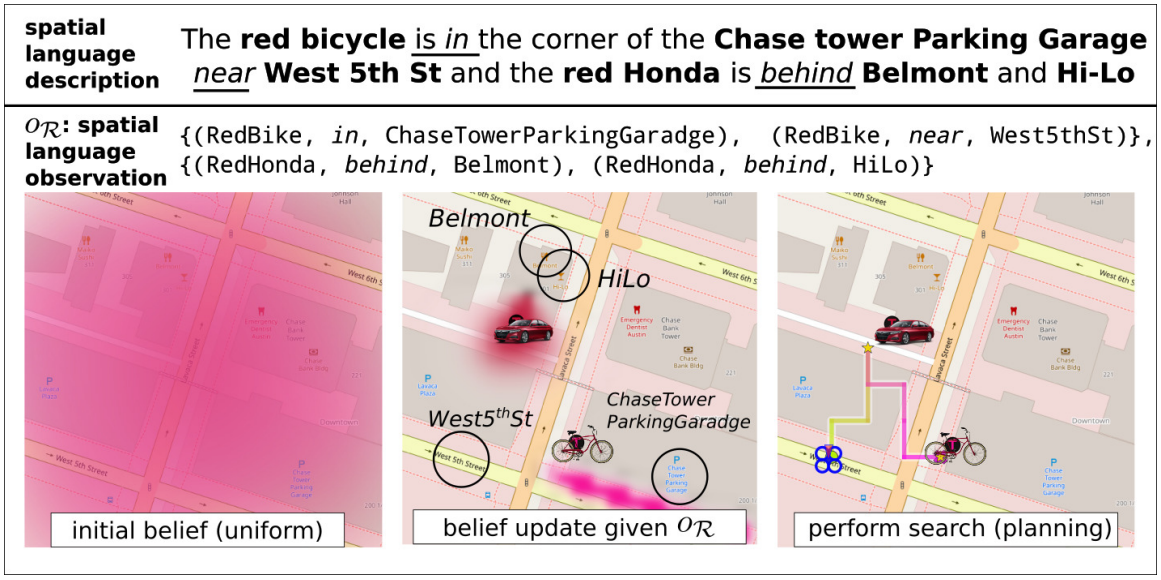


Figure 7.2: We consider a spatial language description as an observation $o_{\mathcal{R}}$, which is a set of (f, r, γ) tuples, obtained through parsing the input language. We propose an observation model that incorporates the spatial information in $o_{\mathcal{R}}$ into the robot's belief about target locations, which benefits subsequent object search performance.

our implementation of the spatial language observation model on 2D city maps, which includes a convolutional network model for FoR prediction.

7.4 Spatial Language Object-Oriented POMDP (SLOOP)

SLOOP augments an OO-POMDP defined over a given map $\mathcal{M}$ with a spatial language observation space and a spatial language observation model. The map $\mathcal{M}$ consists of a discrete set of locations and contains landmark information (e.g. landmark's name, location and geometry), such that N objects exist on the map at possibly unknown locations. Thus, the state space can be factored into a set of N objects plus the given map $\mathcal{M}$ and robot state $s = (s_1, \cdots, s_N, s_r, \mathcal{M})$. SLOOP does not augment the action space, thus the action space of the underlying OO-POMDP is left unchanged. Because the transition and reward functions are, by definition, independent of observations, they are also kept unchanged in SLOOP. Next, we introduce spatial language observations.

7.4.1 Spatial Language Observation

According to Landau & Jackendoff (1993), the standard linguistic representation of an object's place requires three elements: the object to be located (*figure*), the reference object (*ground*), and their relationship (*spatial relation*). We follow this convention and represent

spatial information from a given natural language description in terms of atomic propositions, each represented as a tuple of the form (f, r, γ), where f is the *figure*, r is the *spatial relation* and γ is the *ground*. In our case, f refers to a target object, γ refers to a landmark on the map, and r is a predicate that is true if the locations of f and γ satisfy the semantics of the spatial relation. As an example, given spatial language *the red Honda is behind Belmont, near Hi-Lo*, two tuples of this form can be extracted (parentheses indicate role): {(`RedCar`(f), `behind`(r), `Belmont`(γ)), (`RedCar`(f), `near`(r), `HiLo`(γ))}.

We define a *spatial language observation* as a set of (f, r, γ) tuples extracted from a given spatial language. We define the spatial language observation space to be the space of all possible tuples of such form for a given map, objects, and a set of spatial relations.

7.4.2 Spatial Language Observation Model

We denote a spatial language observation as $o_{\mathcal{R}}$. Our goal now is to derive $\Pr(o_{\mathcal{R}}|s', a)$, the observation model for spatial language. We can split $o_{\mathcal{R}}$ into subsets, $o_{\mathcal{R}} = \cup_{i=1}^{N} o_{\mathcal{R}_i}$, where each $o_{\mathcal{R}_i} = \cup_{k=1}^{L}(f_i, r_k, \gamma_k)$, $L = |o_{\mathcal{R}_i}|$ is the set of tuples where the figure is target object i. Since the human describes the target objects with respect to landmarks on the map, the spatial language is conditionally independent from the robot state and action given map $\mathcal{M}$ and the target locations $s_1, \cdots, s_N$. Therefore,

$$\Pr(o_{\mathcal{R}}|s', a) = \Pr(\cup_{i=1}^{N} o_{\mathcal{R}_i}|s'_1, \cdots, s'_N, \mathcal{M}) \tag{7.1}$$

We make a simplifying assumption that $o_{\mathcal{R}_i}$ is conditionally independent of all other $o_{\mathcal{R}_j}$ and s'_j ($j \neq i$) given s'_i and map $\mathcal{M}$. We make this assumption because the human observer is capable of producing a language $o_{\mathcal{R}_i}$ to describe target i given just the target location s_i and the map $\mathcal{M}$. Thus,

$$\Pr(\cup_{i=1}^{N} o_{\mathcal{R}_i}|s'_1, \cdots, s'_N, \mathcal{M}) = \prod_{i=1}^{N} \Pr(o_{\mathcal{R}_i}|s'_i, \mathcal{M}) \tag{7.2}$$

where $\Pr(o_{\mathcal{R}_i}|s'_i, \mathcal{M})$ models the spatial information in the language description for object i. For each spatial relation r_j in $o_{\mathcal{R}_i}$ whose interpretation depends on the FoR imposed by the human observer (e.g. *behind*), we introduce a corresponding random variable Ψ_j denoting the FoR vector that distributes according to the indicator function $\Pr(\Psi_j = \psi_j) = \mathbb{1}(\psi_j = \psi_j^*)$, where ψ_j^* is the one imposed by the human, unknown to the robot. Then, our model for $\Pr(o_{\mathcal{R}_i}|s'_i, \mathcal{M})$ becomes:

$$\Pr(o_{\mathcal{R}_i}|s'_i, \mathcal{M}) = \prod_{j=1}^{L} \Pr(r_j|\gamma_j, \psi_j^*, f_i, s'_i, \mathcal{M}) \tag{7.3}$$

The step-by-step derivation can be found in the supplementary material. It is straightforward to extend this model as a mixture model $\Pr(o_{\mathcal{R}_i}|s'_i, \mathcal{M}) = \sum_{k=1}^{m} w_k \Pr_k(o_{\mathcal{R}_i}|s'_i, \mathcal{M})$, $\sum_{k=1}^{m} w_k = 1$, where multiple interpretations of the spatial language are used to form

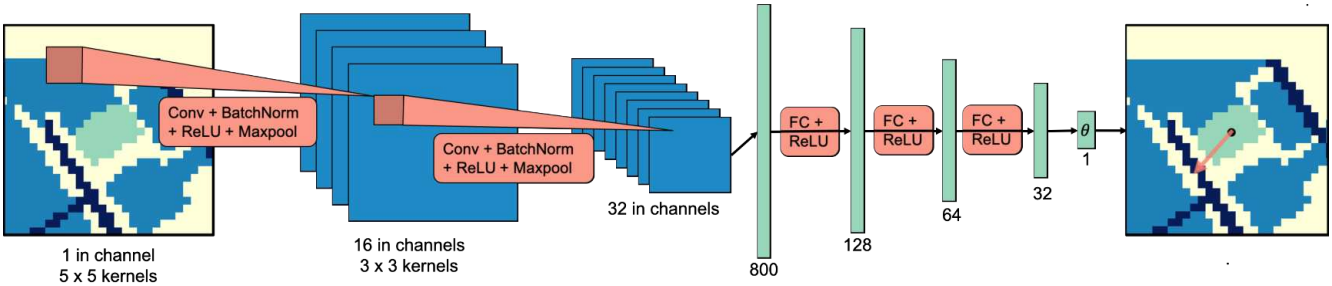


Figure 7.3: **Frame of Reference Prediction Model Design.** In this example taken from our dataset, the model is predicting the frame of reference for the preposition *front*. The grayscale image is rendered with color ranging from blue (black), green (gray) to yellow (white). Green highlights the referenced landmark, dark blue the streets, blue the surrounding buildings, and yellow the background.

separate distributions then combined into a weighted-sum. This effectively smooths the distribution under individual interpretations, which improves object search performance in our evaluation (Figure 7.7),

However, to proceed modeling $\Pr(r_j|\gamma_j, \psi_j^*, f_i, s_i', \mathcal{M})$ in Eq. (7.3), we notice that it depends on the unknown FoR ψ_j^*. Therefore, we consider two subproblems instead: The approximation of ψ_j^* by a predicted value $\hat{\psi}$ and the modeling of $\Pr(r_j|\gamma_j, \hat{\psi}_j, f_i, s_i', \mathcal{M})$. Next, we describe our approach to these two subproblems for object search in city-scale environments.

7.4.3 Learning to Predict Latent Frame of Reference

Here we describe our approach to predict ψ_j^* corresponding to a given (f_i, r_j, γ_j) tuple, which is critical for correct resolution of spatial relations. Taking inspiration from the ice cream truck example where the tourist can infer a potential FoR by looking at the 2D map of the park, we train a model that predicts the human observer's imposed FoR based on the environment context embedded in the map.

We define an FoR in a 2D map as a single vector $\psi_j = (x, y, \theta)$ located at (x, y) at an angle θ with respect to the $+x$ axis of the map. We use the center-of-mass of the ground as the origin (x, y). We make this approximation since our data collection shows that when looking at a 2D map, human observers tend to consider the landmark as a whole without decomposing it into parts. Therefore, the FoR prediction problem becomes a regression problem of predicting the angle θ given a representation of the environment context.

We design a convolutional neural network, which takes as input a grayscale image representation of the environment context where the ground in the spatial relation is highlighted. Surrounding landmarks are also highlighted with different brightness for streets

and buildings (Figure 7.3). The image is shifted to be egocentric with respect to the referenced landmark, and cropped into a 28×28 pixel image. The intuition is to have the model focus on immediate surroundings, as landmarks that are far away tend not to contribute to inferring the referenced landmark's properties. The model consists of two standard convolution modules followed by three fully connected layers. These convolution modules extract an 800-dimension feature vector, feeding into the fully connected layers, which eventually output a single value for the FoR angle. We name this model **EGO-CTX** for egocentric shift of the contextual representation.

Regarding the loss function, a direct comparison with the labeled FoR angle is not desirable. For example, suppose the labeled angle is 0 (in radians). Then, a predicted angle of 0.5 is qualitatively the same as another predicted angle of -0.5 or $2\pi - 0.5$. For this reason, we apply the following treatment to the difference between predicted angle θ and the labeled angle θ^*. Here, both θ and θ^* have been reduced to be between 0 to 2π:

$$\ell(\theta, \theta^*) = \begin{cases} 2\pi - |\theta - \theta^*|, & \text{if } |\theta - \theta^*| > \pi, \\ |\theta - \theta^*|, & \text{otherwise} \end{cases} \tag{7.4}$$

This ensures that the angular deviation used to compute the loss ranges between 0 to π. The learning objective is to reduce such deviation to zero. To this end, we minimize the mean-squared error loss $L(\boldsymbol{\theta}, \boldsymbol{\theta}^*) = \frac{1}{N}\sum_{i=1}^{N}\left(\ell(\theta_i, \theta_i^*)\right)^2$, where $\boldsymbol{\theta}, \boldsymbol{\theta}^*$ are predicted and annotated angles in the training set of size N. This objective gives greater penalty to angular deviations farther away from zero.

In our experiments, we combine the data by antonyms and train two models for each baseline: a **front** model used to predict FoRs for *front* and *behind*, and a **left** model used for *left* and *right*[1]. We augment the training data by random rotations for **front** but not for **left**.[2] We use the Adam optimizer (Kingma & Ba, 2015) to update the network weights with a fixed learning rate of 1×10^{-5}. Early stopping based on validation set loss is used with a patience of 20 epochs (Prechelt, 1998).

7.4.4 Modeling Spatial Relations

We model $\Pr(r_j|\gamma_j, \hat{\psi}_j, f_i, s_i', \mathcal{M})$ as a Gaussian following prior work and evidence from animal behavior (Fasola & Mataric, 2013; O'Keefe & Burgess, 1996):

$$\begin{aligned} &\Pr(r_j|\gamma_j, \hat{\psi}_j, f_i, s_i', \mathcal{M}) \\ &\quad = |u(s_i', \gamma_j, \mathcal{M}) \cdot v(f_i, r_j, \gamma_j, \hat{\psi}_j)| \\ &\qquad \times \exp\left(-dist(s_i', \gamma_j, \mathcal{M})^2/2\sigma^2\right) \end{aligned} \tag{7.5}$$

1. We do not train a single model for all four prepositions since *left* and *right* often also suggest an *absolute* FoR used by the language provider when looking at 2D maps, while *front* and *behind* typically suggest a *relative* FoR.
2. Again, because *left* and *right* may imply either absolute or relative FoR.

where σ controls the steepness of the distribution based on the spatial relation's semantics and landmark size, and $dist(s'_i, \gamma_j, \mathcal{M})$ is the distance between s'_i to the closest position within the ground γ_j in map $\mathcal{M}$, and $u(s'_i, \gamma_j, \mathcal{M}) \cdot v(f_i, r_j, \gamma_j, \hat{\psi}_j)$ is the dot product between $u(s'_i, \gamma_j, \mathcal{M})$, the unit vector from s'_i to the closest position within the ground γ_j in map $\mathcal{M}$, and $v(f_i, r_j, \gamma_j, \hat{\psi}_j)$, a unit vector in in the direction that satisfies the semantics of the proposition (f_i, r_j, γ_j) by rotating $\hat{\psi}_j$. The dot product is skipped for prepositions that do not require FoRs (e.g. *near*). We refer to Landau & Jackendoff (1993) for a list of prepositions meaningful in 2D that require FoRs: *above, below, down, top, under, north, east, south, west, northwest, northeast, southwest, southeast, front, behind, left, right*.

7.5 Data Collection

In this section, we describe our data collection process as well as a pipeline for spatial information extraction from natural language. We use maps from OpenStreetMap (OSM), a free and open-source database of the world map with voluntary landmark contributions (OpenStreetMap contributors, 2017). We scrape landmarks in 40,000m^2 grid-regions with a resolution of 5m by 5m grid cells in five different cities leading to a dimension of 41×41 per grid map[3]: Austin, TX; Cleveland, OH; Denver, CO; Honolulu, HI, and Washington, DC. Geographic coordinates of OSM landmarks are translated into grid map coordinates.

To collect a variety of spatial language descriptions from each city, we randomly generate 30 environment configurations for each city, each with two target objects. We prompt Amazon Mechanical Turk (AMT) workers to describe the location of the target objects and specify that the robot knows the map but does not know target locations. Each configuration is used to obtain language descriptions from up to eleven different workers. The descriptions are parsed using our pipeline described next in Sec. 7.5.1. Examples are shown in Fig. 7.4. Screenshots of the survey and statistics of the dataset are provided in the supplementary material.

The authors annotated FoRs for *front, behind, left* and *right* through a custom annotation tool which displays the AMT worker's language alongside the map without targets. We manually infer the FoR used by the AMT worker, similar to what the robot is tasked to do. This set of annotations are used as data to train and evaluate our FoR prediction model. Prepositions such as *north, northeast* have absolute FoRs with known direction. Others are either difficult to annotate (e.g. *across*) or have too little samples (e.g. *above*, *below*).

7.5.1 Spatial Information Extraction from Natural Language

We designed a pipeline to extract spatial relation triplets from the natural language using the spaCy library (Honnibal & Montani, 2017) for noun phrase (NP) identification and

3. Because of the curvature of the earth, the grid cells and overall region is not perfectly square, which is why the grid is not perfectly 40x40

Figure 7.4: Map screenshot shown to AMT workers paired with collected spatial language descriptions.

dependency parsing, as it achieves good performance on these tasks. Extracted NPs are matched against synonyms of target and landmark symbols using cosine similarity. All paths from targets to landmarks in the dependency parse tree are extracted to form the (f, r, γ) tuples used as spatial language observations (Sec. 7.4).

Our spatial language understanding models assume as input language that has been parsed into (f, r, γ) tuples, but is not dependent on this exact pipeline for doing so. Future work could explore alternative methods for parsing and entity linking, including approaches optimized for the task of spatial language resolution. In our end-to-end experiments, we report the task performance both when using this parsing pipeline and when using manually annotated (f, r, γ) tuples to indicate the influence of parsing on search performance.

7.6 Evaluation

7.6.1 Evaluation of Frame of Reference Prediction Model

We test the generalizability of our FoR prediction model (**EGO-CTX**) by cross-validation. The model is trained on maps from four cities and tested on the remaining held-out city for all possible splits. We evaluate the model by the angular deviation between predicted and annotated FoR angles, in comparison with three baselines and human performance: The first is a variation (**CTX**) that uses a synthetic image with the same kind of contextual representation yet without egocentric shift. The second is another variation (**EGO**) that performs egocentric shift and also crops a 28×28 window, but only highlights the referenced landmark at the center without contextual information. The random baseline (**Random**) predicts the angle at random uniformly between $[0, 2\pi]$. The **Human** performance is obtained by first computing the differences between pairs of annotated FoR angles for the same landmarks (Eq. 7.4), then averaging over all such differences for landmarks per city.

Each pair of FoRs may be annotated by the same or different annotators. Taking the average gives a sense of the disagreement among the annotators' interpretation of spatial relations.

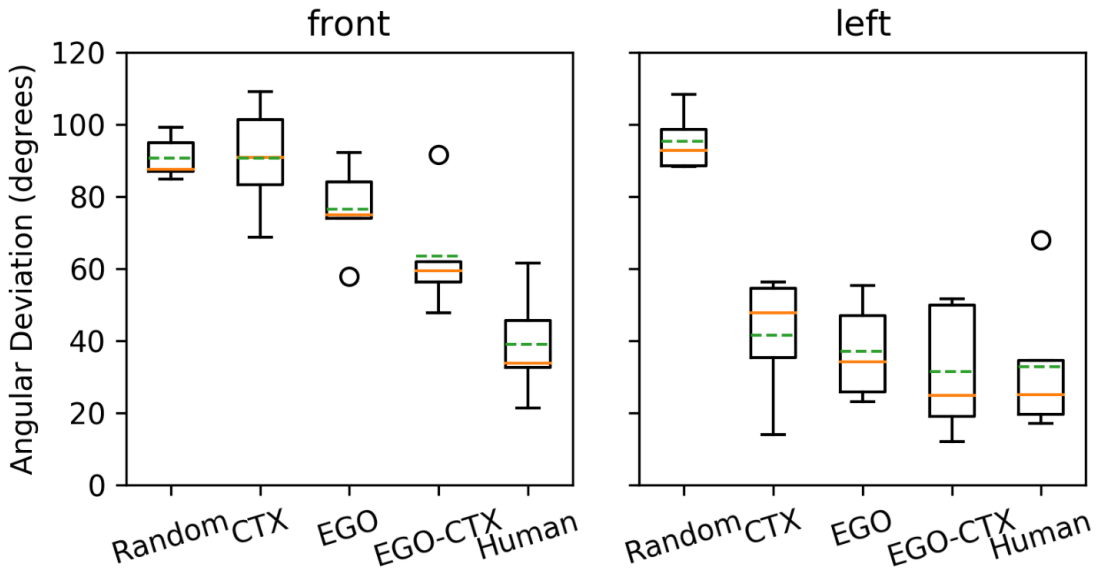


Figure 7.5: FoR prediction results. The solid orange line shows the median, and the dotted green line shows the mean. The circles are outliers. Lower is better.

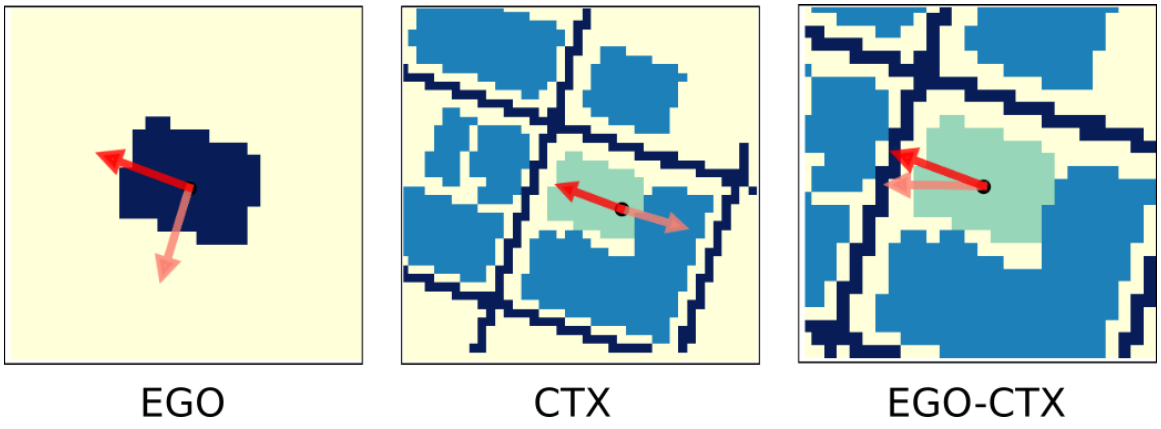


Figure 7.6: Visualization of FoR predictions for *front*. Darker arrows indicate labeled FoR, while brighter arrows are predicted FoR.

The results are shown in Figure 7.5. Each boxplot summarizes the means of baseline performance in the five cross-validation splits. The results demonstrate that **EGO-CTX** shows generalizable behavior close to the human annotators, especially for **front**. We observe that our model is able to predict *front* FoRs roughly perpendicular to streets against

other buildings, while the baselines often fail to do so (Figure 7.6). The competitive performance of the neural network baselines in **left** indicates that for *left* and *right*, the FoR annotations are often absolute, i.e. independent of the context. Our model as well as baselines are limited in determining, for example, whether the speaker refers to the left side of the map (absolute FoR), or the left side of the street relative to a perceived forward direction (relative FoR).

7.6.2 End-to-End Evaluation

We randomly select 20 spatial descriptions per city. We task the robot to search for each target object mentioned in every description separately, resulting in a total of 40 search trials per city, 200 in total. Cross-validation is employed such that for each city, the robot uses the FoR prediction model trained on the other four cities. For each step, the robot can either move or mark an object as detected. The robot can move by rotating clockwise or counterclockwise for 45 degrees, or move forward by 3 grid cells (15m). The robot receives observation through an on-board fan-shaped sensor after every move. The sensor has a field of view with an angle of 45 degrees and a varying depth of 3, 4, 5 (15m, 20m, 25m). As the field of view becomes smaller, the search task is expected to be more difficult. The robot receives $R_{\text{step}} = -10$ step cost for moving and $R_{\max} = +1000$ for correctly detecting the target, and $R_{\min} = -1000$ if the detection is incorrect. The rest of the domain setup follows (Wandzel et al., 2019).

Baselines. **SLOOP** uses the spatial language observation model without mixture, that is, for each object, it computes the observation distribution in Eq. (7.3) by multiplying the distributions for each spatial relation; With the same observation distribution, **SLOOP (m=2)** mixes in one distribution computed by treating all prepositions as *near* with weight 0.2; Also with the same observation distribution, **SLOOP (m=4)** mixes in three additional distributions: one ignores FoR-dependent prepositions, one treating all prepositions as *near*, one treating all prepositions as *at*, with weights 0.25, 0.1, 0.05, respectively. The baseline **MOS (keyword)** uses a keyword-based model based on (Wandzel et al., 2019) that assigns a uniform probability over referenced landmarks in a spatial language but does not incorporate information from spatial prepositions. Finally, **informed** and **uniform** are upper and lower bounds: for the **informed**, the agent has an initial belief that has a small Gaussian noise over the groundtruth location[4]; **uniform** uses a uniform prior. We also report the performance with annotated spatial relations and landmarks to show search performance if the languages are parsed correctly.

For all baselines, we use an online POMDP solver, POMCP (Silver & Veness, 2010) but with a histogram belief representation to avoid particle depletion. The number of simulations per planning step is 1000 for all baselines. The discount factor is set to 0.95. The robot is allowed to search for 200 steps per search task, since search beyond this point will earn very little discounted reward and is not efficient.

4. The noise is necessary for object search, otherwise the task is trivial.

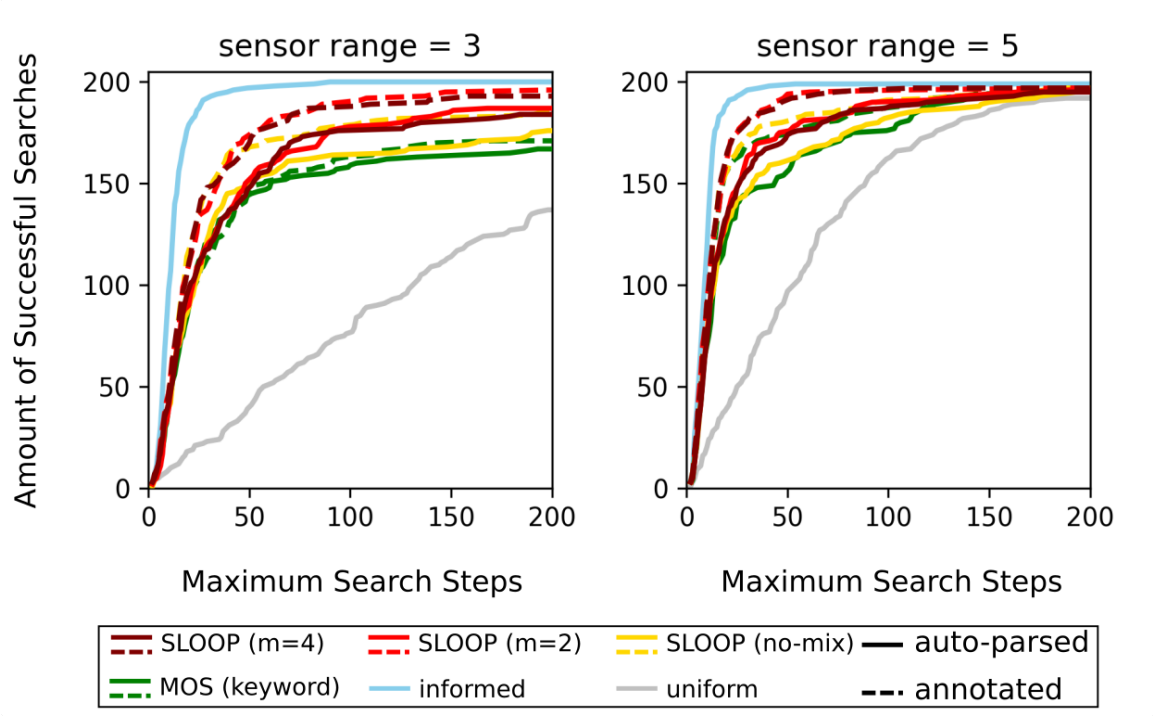


Figure 7.7: Number of completed search tasks as the maximum search step increases. Steeper slope indicates greater efficiency and success rate of search.

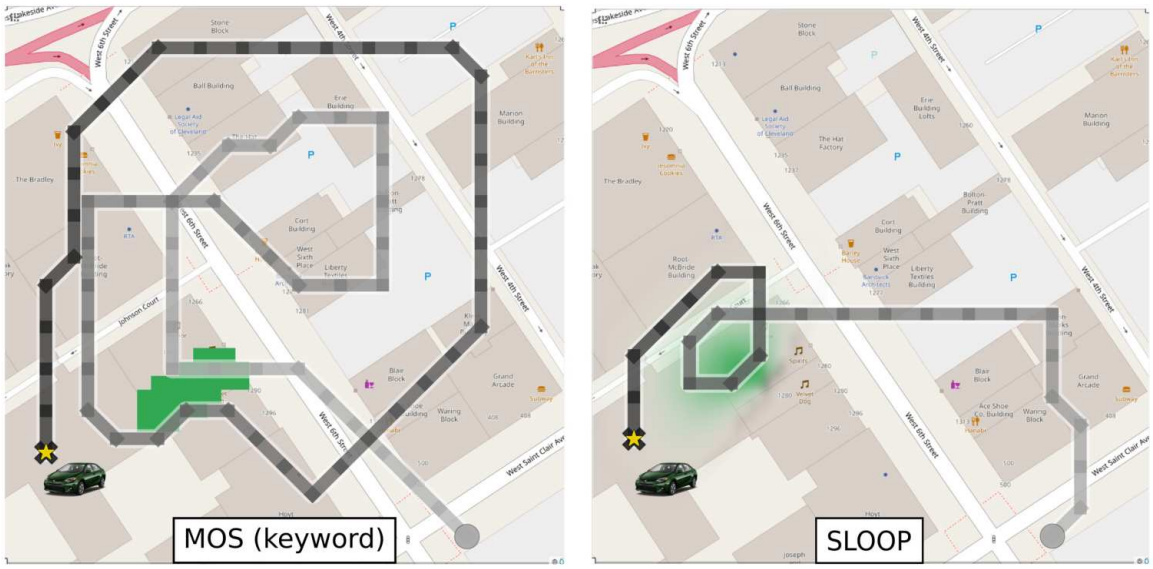


Figure 7.8: Example object search trial for description "the green toyota is *behind* velvet dog" from AMT. The green region shows the distribution over the object location after interpreting the description. Our method enables probabilistic interpretation of the spatial language leading to more efficient search strategy.

Results. We evaluate the effectiveness and efficiency of the search by the amount of search tasks the robot completed (i.e. successfully found the target) under a given limit of search steps (ranging from 1 to 200). Results are shown in Figure 7.7. The results show that using spatial language with SLOOP outperforms the keyword-based approach in MOS. The gain in the discounted reward is statistically significant over all sensor ranges comparing **SLOOP** with **MOS (keyword)**, and for sensor range 3 comparing the annotated versions. We observe that using mixture models for spatial language improves search efficiency and success rate over **SLOOP**. We observe improvement when the system is given annotated spatial relations. This suggests the important role of language parsing for spatial language understanding. Figure 7.8 shows a trial comparing **SLOOP** and **MOS (keyword)**.

spatial preposition	No. trials	MOS (keyword) annodated	SLOOP annotated	SLOOP (m=4) annotated
on	59	200.65 (78.06)	267.00 (76.45)	**290.05 (70.51)**
at	58	179.42 (81.46)	237.36 (81.03)	**238.24 (80.50)**
near	35	97.64 (135.39)	**280.69 (109.00)**	249.35 (113.60)
between	25	21.48 (116.93)	172.93 (143.77)	**175.59 (136.71)**
in	22	302.05 (151.42)	**398.88 (119.32)**	307.45 (141.67)
north	9	222.28 (291.13)	201.88 (296.49)	**365.14 (246.05)**
southeast	7	306.77 (341.43)	553.82 (174.04)	**549.43 (165.83)**
southwest	7	-75.67 (205.98)	**1.37 (271.46)**	-27.63 (281.89)
east	6	56.57 (337.58)	290.68 (303.53)	**439.99 (276.85)**
northwest	6	**385.41 (320.82)**	43.57 (282.88)	-1.82 (256.71)
south	6	79.12 (289.54)	310.29 (410.04)	**494.26 (161.60)**
west	4	-160.91 (188.02)	234.93 (587.57)	**327.13 (245.28)**
northeast	2	-167.99 (660.76)	206.42 (138.93)	**213.17 (977.62)**
front	25	**246.96 (142.45)**	168.91 (150.41)	160.55 (136.88)
behind	8	128.47 (356.25)	101.20 (333.38)	**140.92 (333.61)**
right	4	19.75 (697.88)	160.14 (601.54)	**336.00 (725.84)**
left	3	**247.35 (363.32)**	192.93 (393.75)	231.75 (330.33)
front (good)	15	255.85 (210.46)	**421.83 (143.65)**	222.67 (264.50)
behind (good)	6	145.26 (489.55)	207.58 (430.52)	**359.80 (753.15)**
front (bad)	10	**281.04 (226.47)**	-208.92 (11.39)	93.80 (176.11)
behind (bad)	2	**78.11 (3771.84)**	-217.95 (23.53)	-77.97 (223.71)

Table 7.1: Mean (95% CI) of discounted cumulative reward for different prepositions evaluated on language descriptions with annotated spatial relations. The value with highest mean per row is bolded.

We analyze the performance with respect to different spatial prepositions. We report results for annotated languages as they reflect the performance obtained if the prepositions

are correctly identified. Results for the smallest sensor range of 3 is shown in Table 7.1. **SLOOP** outperforms the baseline for the majority of prepositions. For prepositions *front*, *behind*, *left*, and *right*, our investigation shows the performance of **SLOOP** polarizes where trials with "good" FoR (i.e. ones in the correct direction towards the true target location) leads to a much greater performance than the counterpart ("bad" FoR). Yet, **MOS (keyword)** is not subject to such polarization and the target often appears close to the landmark for these prepositions. We observe that **SLOOP (m=4)** using mixture is able to consistently improve the reward for most of the prepositions, indicating the benefit of modeling multiple interpretations of the spatial language.

No. spatial prepositions	No. trials	MOS (keyword) annotated	SLOOP annotated	SLOOP (m=4) annotated
1	100	234.32 (72.64)	**320.91 (64.23)**	289.34 (66.42)
2	83	179.18 (68.60)	264.08 (62.39)	**286.19 (60.83)**
3	14	26.96 (165.98)	115.44 (202.42)	**215.30 (200.99)**

Table 7.2: Mean (95% CI) of discounted cumulative reward for completed search tasks as language complexity (number of spatial relations) increases.

Finally, we analyze the relationship between the performance and varying complexity of the language description, indicated by the number of spatial relations used to describe the target location. Again, we used annotated languages for this experiment for the smallest sensor range of 3. Results in Table 7.2 indicate that understanding spatial language can benefit search performance, with a wider gain as the number of spatial relations increases. Again, using mixture model in **SLOOP (m=4)** improves the performance even more.

7.6.3 Demonstration on AirSim

We implemented **SLOOP (m=4)** on AirSim (S. Shah et al., 2017), a realistic drone simulator built on top of Unreal Engine (Epic Games, n.d.). Similar to our evaluation in OpenStreetMap, we discretize the map into 41×41 grid cells. We use the same fan-shaped model of on-board sensor as in OpenStreetMap. As mentioned in Sec. 7.3, sensor observations are synthetic, based on the ground-truth state. Additionally, although the underlying localization and control is continuous, the drone plans discrete navigation actions (move forward, rotate left 90°, rotate right 90°). We annotated landmarks (houses and streets) in the scene on the 2D grid map. Houses with heights greater than flight height are subject to collision and results in a large penalty reward (-1000). Checking for collision in the POMDP model for this domain helped prevent such behavior during planning. We found that the FoR prediction model trained on OpenStreetMap generalizes to this domain, consistently producing reasonable FoR predictions for *front* and *behind*. This shows the benefit of using synthetic images of top-down street maps. The drone is able to plan actions to search back and forth to find the object, despite given inexact spatial language description. Please refer to the video demo on our project for the examples shown in Fig. 7.1 and 7.9.

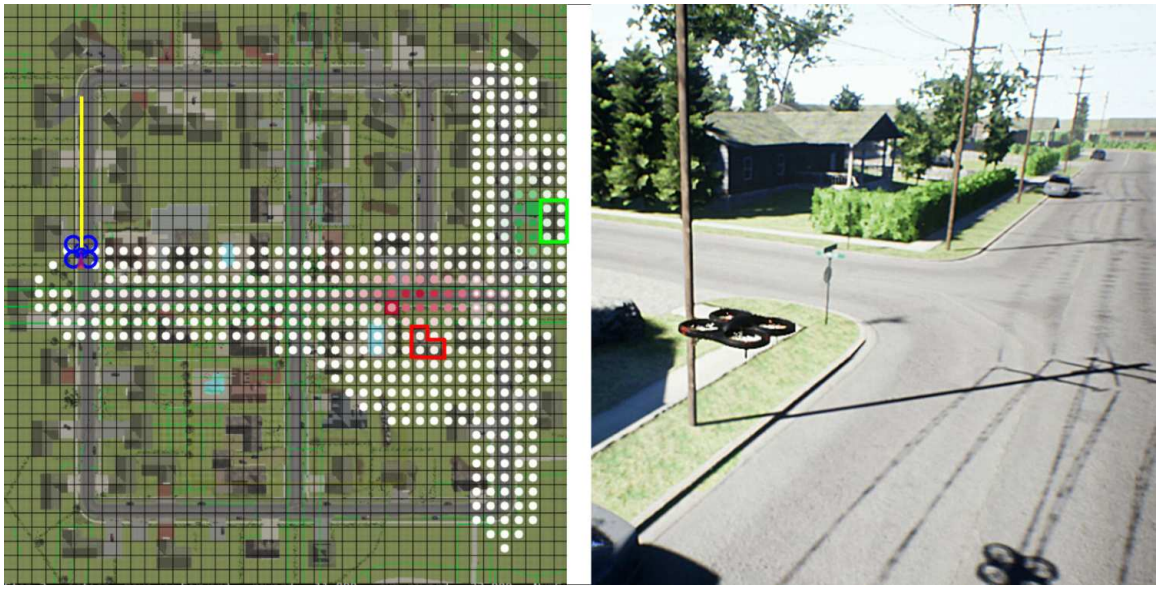


Figure 7.9: Example trial from AirSim demonstration. Given spatial language description: *The red car is by the side of Main street in front of Zoey's House (red), while the green car is within Annie's House on the right side of East street (green).* Left: belief over two target objects (red and green car). Right: screenshot from AirSim. Video: https://youtu.be/V54RY8v8VmA

7.6.4 Demonstration on Boston Dynamics Spot

We take one step further and demonstrate SLOOP on a physical robot. In particular, we task a Boston Dynamics Spot robot to find a book in a lab environment. We developed a 2D MOS-based object search system integrated with ROS that can interact with Spot through the Spot SDK. This system uses histogram-based 2D belief, and the object search POMDP model is the same 2D MOS model used for simulation experiments.[5] Object detection was done through projecting segmentation masks obtained with Mask-RCNN (He et al., 2017) to 2D grid cells using depth from Spot's gripper camera. To obtain the same kind of 2D grid map representation as in simulation experiments, we first use Spot's built-in capability to create a 3D point cloud map of the lab and then project it down to 2D, where points are filtered to separate free space from obstacles. Then, in order to apply SLOOP for 2D MOS, we build a map of landmarks similar to OpenStreetMap by driving the robot around the lab, and recording object detection results projected onto a 2D grid, and then assigning a unique name to each detected object. Since our FoR prediction model works over synthetic images, they can be directly applied to this landmark map. The end result is an interface that allows a person to type in a free-form spatial language with respect to the

5. In fact, this system is the npredecessor to GenMOS for 3D-MOS in Chapter 5, GenMOS is robot-independent (does not rely on ROS).

landmarks, and the belief gets updated after the system interprets the spatial language, and the robot goes off to search under the updated belief; Planning is based on the hierarchical planning algorithm introduced for COS-POMDP in Chapter 6 that combines local, fine-grained search actions with navigation subgoals over a topological graph. In Figure 7.10, we contrast the system's behavior with and without spatial language. SLOOP enables the robot to quickly narrow down the search region given *"the book is in front of the Monitor,"* whereas the baseline without language results in searching all over the lab.

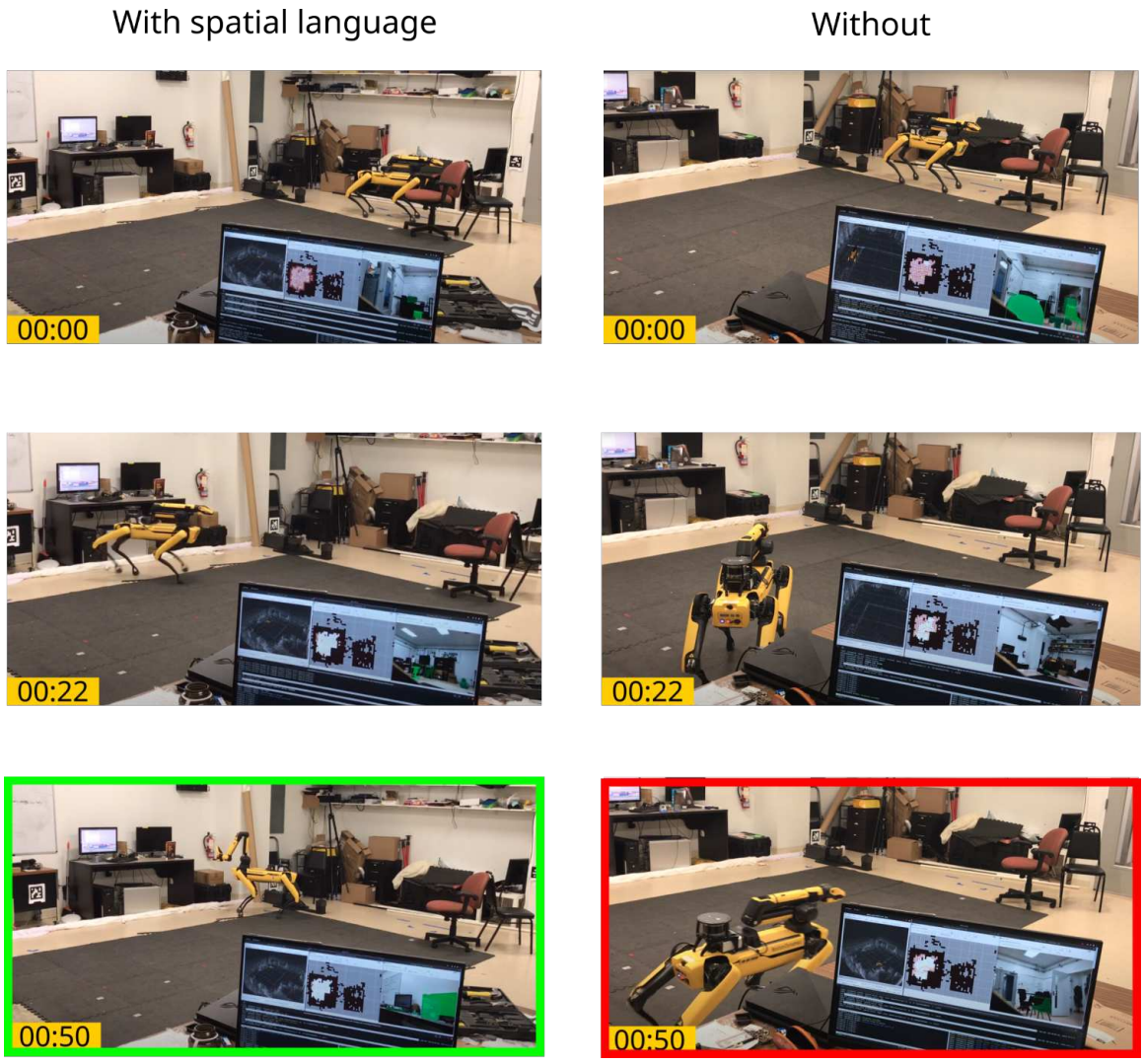


Figure 7.10: Demonstration of SLOOP on Spot, given the spatial language *"the book is in front of the Monitor."* The total time used for "with spatial language" includes the time to initially type in the language, while the baseline without language proceeds to search right off the bat. Nevertheless, SLOOP quickly narrows down the search space and the robot successfully finds the book.

7.7 Summary

This chapter first presents a formalism for integrating spatial language into the POMDP belief state as an observation, then a convolutional neural network for FoR prediction shown to generalize well to new cities. Simulation experiments show that our system significantly improves object search efficiency and effectiveness in city-scale domains through understanding spatial language. For future work, we plan to investigate compositionality in spatial language for partially observable domains.

7.8 Appendix

7.8.1 Derivation of Spatial Language Observation Model

Here we provide the derivation for Eq. (7.3). Using the definition of $o_{\mathcal{R}_i}$,

$$\Pr(o_{\mathcal{R}_i}|s'_i, \mathcal{M}) = \Pr(f_i, r_1, \gamma_1, \cdots, r_L, \gamma_L|s'_i, \mathcal{M}) \tag{7.6}$$

$$= \Pr(r_1, \cdots, r_L|\gamma_1, \cdots, \gamma_L, f_i, s'_i, \mathcal{M}) \times \Pr(\gamma_1, \cdots, \gamma_L, f_i|s'_i, \mathcal{M}) \tag{7.7}$$

$$= \frac{1}{Z} \prod_{j=1}^{L} \Pr(r_j|\gamma_j, f_i, s'_i, \mathcal{M}) \tag{7.8}$$

The first term in (7.7) is factored by individual spatial relations, because each r_j is a predicate that, by definition, involves only the figure f_i and the ground γ_j, therefore it is conditionally independent of all other relations and grounds given f_i, its location s'_i, and the landmark γ_j and its features contained in $\mathcal{M}$. Because the robot has no prior knowledge regarding the human observer's language use,[6] the second term in (7.7) is uniform with probability $1/Z$ where Z is the constant size of the support for $\gamma_1, \cdots, \gamma_L, f_i$. This constant can be canceled out during POMDP belief update upon receiving the spatial language observation, using the belief update formula in Section 2.2.2. We omit this constant in Eq. (7.3).

For predicates such as *behind*, its truth value depends on the relative FoR imposed by the human observer who knows the target location. Denote the FoR vector corresponding to r_j as a random variable Ψ_j that distributes according to the indicator function $\Pr(\Psi_j = \psi_j) = \mathbb{1}(\psi_j = \psi_j^*)$, where ψ_j^* is the one imposed by the human. Then regarding $\Pr(r_j|\gamma_j, f_i, s'_i, \mathcal{M})$, we can sum out Ψ_j:

$$\Pr(r_j|\gamma_j, f_i, s'_i, \mathcal{M}) = \frac{\sum_{\psi_j} \Pr(r_j, \gamma_j, f_i, s'_i, \mathcal{M}|\psi_j) \Pr(\psi_j)}{\Pr(\gamma_j, f_i, s'_i, \mathcal{M})} \tag{7.9}$$

6. In general, the human observer may produce spatial language that mentions arbitrary landmarks and figures whether they make sense or not.

Since the distribution for Ψ_j is an indicator function,

$$= \frac{\Pr(r_j, \gamma_j, f_i, s'_i, \mathcal{M}|\psi^*_j)}{\Pr(\gamma_j, f_i, s'_i, \mathcal{M})} \tag{7.10}$$

By the law of total probability,

$$= \frac{\Pr(r_j|\gamma_j, \psi^*_j, f_i, s'_i, \mathcal{M}) \Pr(\gamma_j, f_i|s'_i, \mathcal{M}, \psi^*_j) \Pr(s'_i, \mathcal{M}|\psi^*_j)}{\Pr(\gamma_j, f_i|s'_i, \mathcal{M}) \Pr(s'_i, \mathcal{M})} \tag{7.11}$$

Using the fact that $s'_i, \mathcal{M}$ is independent of ψ^*_j,

$$= \frac{\Pr(r_j|\gamma_j, \psi^*_j, f_i, s'_i, \mathcal{M}) \Pr(\gamma_j, f_i|s'_i, \mathcal{M}, \psi^*_j) \Pr(s'_i, \mathcal{M})}{\Pr(\gamma_j, f_i|s'_i, \mathcal{M}) \Pr(s'_i, \mathcal{M})} \tag{7.12}$$

Canceling out $\Pr(s'_i, \mathcal{M})$,

$$= \frac{\Pr(r_j|\gamma_j, \psi^*_j, f_i, s'_i, \mathcal{M}) \Pr(\gamma_j, f_i|s'_i, \mathcal{M}, \psi^*_j)}{\Pr(\gamma_j, f_i|s'_i, \mathcal{M})} \tag{7.13}$$

Similar to (7.7)-(7.8), $\Pr(\gamma_j, f_i|s'_i, \mathcal{M}, \psi^*_j)$ and $\Pr(\gamma_j, f_i|s'_i, \mathcal{M})$ are uniform with the same support. Canceling them out,

$$= \Pr(r_j|\gamma_j, \psi^*_j, f_i, s'_i, \mathcal{M}) \tag{7.14}$$

7.8.2 Data Collection Details

Amazon Mechanical Turk Questionnaire

As described in Section 7.5, we collect a variety of spatial language descriptions from five cities. We randomly generate 10 unique configurations of two object locations for every pair of object symbols from {`RedBike`, `RedCar`, `RedCar`}. Each configuration is used to obtain language descriptions from up to eleven different workers. By showing a picture of the objects placed on the map screenshot, we prompt AMT workers to describe the location of the target objects. We first show an example task as shown in Fig 7.11 (top). Then we prompt them with the actual task and a text box to submit their response, as shown in Fig 7.11 (bottom). Note that we specify that the robot does not know where the target objects are, but that it knows the buildings, streets and other landmarks available on the map. We encourage them to use the information available on the map in their description.

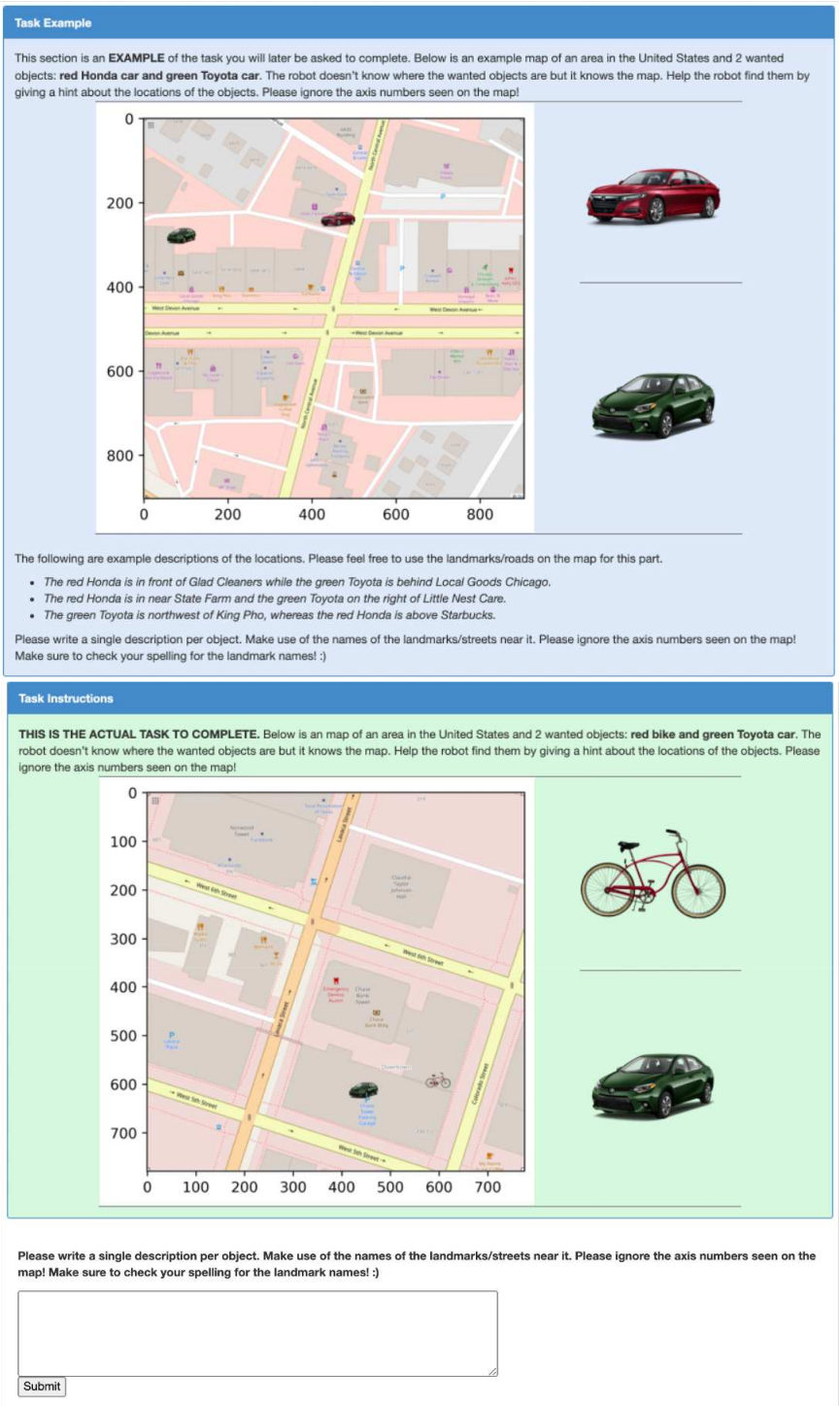


Figure 7.11: AMT Questionnaire Screenshot. Top: an example task shown to the AMT workers prior to the actual task that doesn't vary from prompt to prompt. Bottom: the actual task shown to the AMT workers. The objects and the locations change in every prompt and are all unique.

7.8.3 Distribution of Collected Predicates

Each description is parsed using our pipeline described in Section 7.5.1. 1,521 out of 1,650 gathered descriptions were successfully parsed; meaning at least one spatial relation was extracted from the sentence. The distribution of all spatial predicates are shown in Fig 7.12. Note that we include the word "is" on the list since it often appears in "is in" or "is at", yet the parser sometimes skip the word after it due to an artifact. We excluded language descriptions parsed with such artifact from the ones used in the end-to-end object search evaluation.

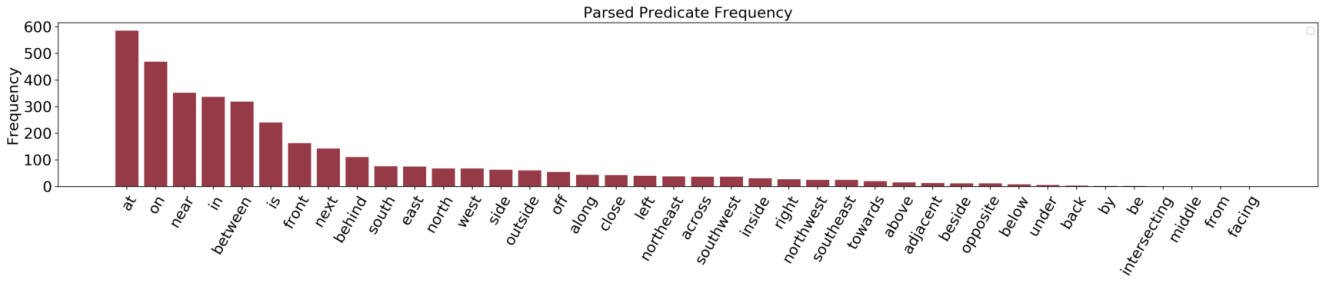


Figure 7.12: Distribution of collected parsable predicates sorted from most frequent to least.

Spatial preposition	# of FoR annotations
front	121
behind	54
left	51
right	47

Table 7.3: Number of FoR annotations per spatial relation.

Figure 7.13: The FoR annotator GUI interface. The FoR consists of a front (purple) and right (green) vector. The target objects are not shown, and the annotator only has access to the spatial language description, and the map image. This mimics the situation faced by the robot in our task.

7.8.4 FoR Annotation

To collect the FoR annotation, we create our custom annotation tool. The FoR consists of a front (purple) and right (green) vector that show how the speaker considers the direction of the ground according to their given spatial description. The interface (Fig 7.13) works as follows: (1) The annotator first clicks the "Annotate" button. (2) The interface prompts the annotator the language phrase corresponding to a spatial relation to be annotated (e.g. "`RedBike is in front of EmpireApartments`"), which is composed using the parsed (f, r, γ) tuple. (3) to annotate an FoR, the annotator clicks on the map as the origin of the FoR, and then clicks on another point on the map as the end point of the *front* vector. The vector for *right* is automatically computed to be 90 degrees clockwise with respect to the *front* vector. (4) After annotating one FoR, the annotator clicks "Next" to move on, and the process starts over again from step (2). In total we have 273 FoR annotations. Table 7.3 shows the amount of annotation per spatial relation. You can download the dataset by visiting the website linked in the footnote on the first page.

CHAPTER 8

Dialogue Object Search: Preliminary Work

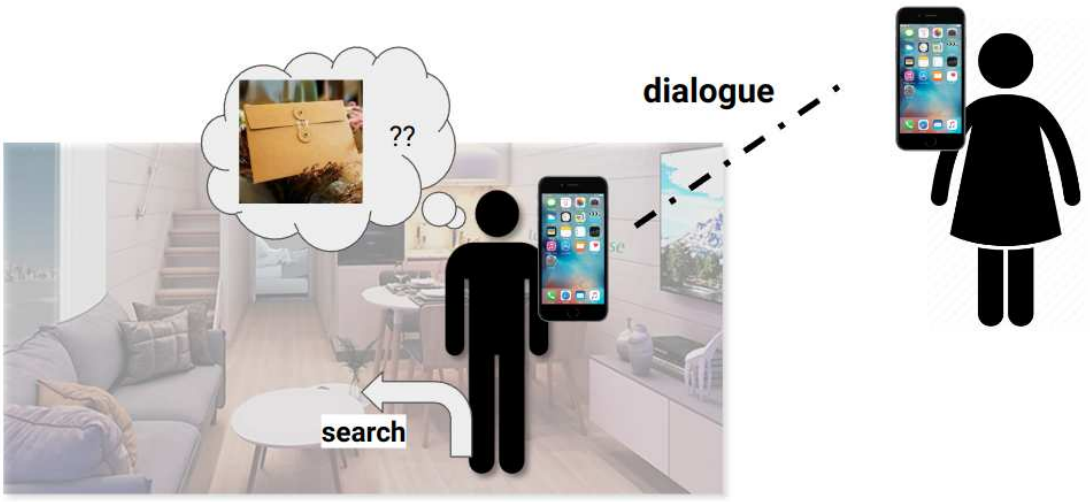


Figure 8.1: The motivating scenario for dialogue object search. Imagine one person is searching for a file at home. Not knowing where it could be, he calls a family member who knows more. Then, he is able to verbally engage in the dialogue over the phone while performing search physically. We view dialogue object search as one route towards realizing such ability to decide what to say and how to act simultaneously, fundamental for future collaborative robots.

8.1 Motivation

We envision robots that can collaborate and communicate seamlessly with humans. It is necessary for such robots to decide both what to say and how to act, while interacting with humans. To this end, we introduce a new task, *dialogue object search*: A robot is tasked to search for a target object (e.g. fork) in a human environment (e.g., kitchen), while engaging in a "video call" with a remote human who has additional but inexact knowledge about the target's location. That is, the robot conducts speech-based dialogue with the human, while sharing the image from its mounted camera. This task is challenging at multiple levels, from data collection, algorithm and system development, to evaluation. Despite these challenges, we believe such a task blocks the path towards more intelligent and collaborative robots. In this extended abstract, we motivate and introduce the dialogue object search task and analyze examples collected from a pilot study. We then discuss our next steps and conclude with several challenges on which we hope to receive feedback.

Humans can act in the physical world (such as walking, looking, or opening a cabinet) while having a conversation with others. As robots enter homes and care centers, we envision them to have such capability as well when collaborating and communicating with humans. To achieve this, robots must decide both what to say and how to act towards a goal. This involves combining task-oriented dialogue systems with decision making under uncertainty for embodied agents. Traditionally, dialogue systems have involved users interacting with a virtual agent for tasks such as technical support (Mouromtsev et al., 2015), personal assistance (e.g., Siri) and booking reservations (Wen et al., 2017; Wei et al., 2018). While recent works have proposed datasets that combine dialogue and dynamic, embodied decision making (de Vries et al., 2018; Thomason et al., 2020), the investigated problems over these datasets are limited to prediction tasks that bypass the challenges of evaluating a conversational embodied agent. For example, the Navigation from Dialog History Task (Thomason et al., 2020) asks the agent to predict the next navigation action, given a history of dialogue and past navigation actions. The tourist localization task (de Vries et al., 2018) asks the system to predict a location given a language description.

Our goal is to enable robots to naturally engage in a dialogue with a human while completing a task autonomously. We believe a task that captures the sequential nature of both the dialogue and physical decision making is necessary for in-depth study towards this goal. We choose to focus on object search, a useful and widely-studied problem (Aydemir et al., 2013; Kollar & Roy, 2009; Zheng et al., 2021b, 2021a)

8.2 Contributions

The main contribution is we introduce a new task called *dialogue object search*. From the pilot study (Sec. 8.4), we observed that participants produced language and behavior that are more natural using speech, because text-based dialogue requires users to decide

whether to type or act at every step. Additionally, we summarize the set of intent types that are observed to succinctly capture the intents behind the unstructured utterances in the dialogue object search task.

8.3 Dialogue Object Search

A robot is tasked to search for a target object in a human environment (e.g., kitchen) while engaging in an audio dialogue with a remote human assistant, who possesses inexact prior knowledge about the target object's location. In our pilot study, this is given in the form of a 2D scatter plot (Fig. 8.2). The robot has a mounted RGB-D camera, and shares its view with the human assistant. We assume the robot and the human assistant have access to two different sequences of RGB-D images of the scene, which represent their prior experiences of living in that environment. Target objects are excluded from these images. The robot must decide what to say and how to act, in order to efficiently find the target object while naturally interacting and collaborating with the human assistant.

Our inspiration for the above setting comes from the following scenario between two people living together (family or friends). One person is searching for something, such as a document or a key, but not sure where it is. They decide to video call the other home member who is currently out of the house but may have a better idea. They then engage in a dialogue while the first person conducts the search for the target object. We envision that in the future, this could happen between a home assistant robot and a human user.

8.4 Pilot Study

To investigate the above task, we first attempted to understand how a human would behave if they are in the robot's position. We designed and conducted a pilot study among three pairs of people (authors' lab members) using AI2-THOR (Kolve et al., 2017) as the simulated home environments. In this study, we designate two roles according to the above problem setting. The *Assistant* is the person assisting in the process as the robot searches for a given target object. The *Controller* is the person who is taking on the role of the robot. Due to the pandemic, we used Zoom to record the audio and create transcripts of the dialogue. We implemented a web-based data collection tool where the *Controller* controls the agent in AI2-THOR through the web interface, and the *Assistant* has access to a 2D scatter plot of a subset of objects in the scene (Fig. 8.2). Each pair of participants are assigned three object search trials in one environment. They have 90 seconds to explore the environment (with target objects removed) and 180 seconds to complete each trial. In addition to dialogue audio and transcripts, we collected data about the scene per view, the action executed, and the agent's groundtruth pose as provided by the AI2-THOR frame-

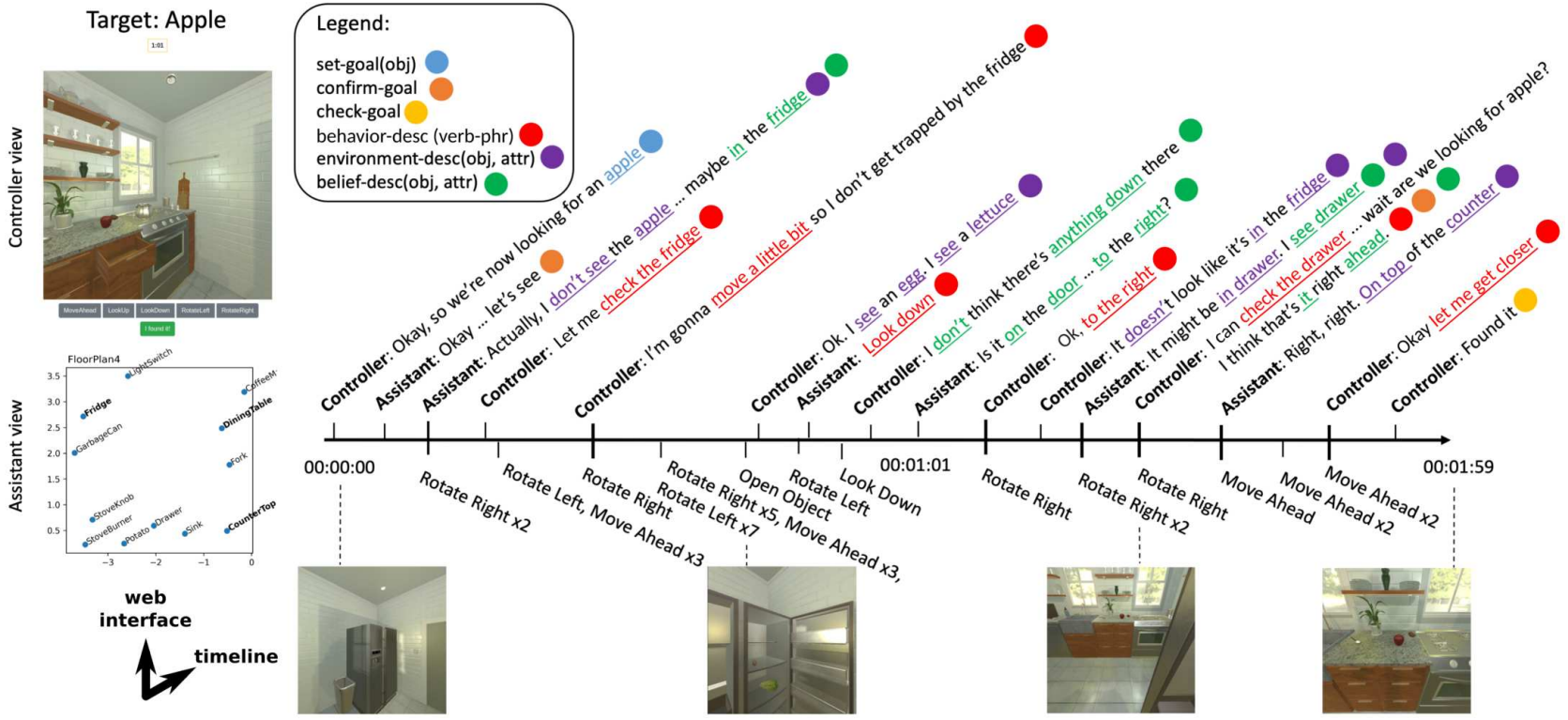


Figure 8.2: We conducted a pilot study to understand desirable behavior for the dialogue object search task. Shown here is a screenshot of the web interface (left) and the dialogue and actions organized onto a timeline (right), for an object search trial where the target object is `Apple`. We classified the dialogue utterances into a preliminary set of parameterized intents, indicated by the colors.

work. We considered a discrete action space of {*MoveAhead*(0.25m), *RotateLeft*(45°), *RotateRight*(45°), *LookUp*(30°), *LookDown*(30°), *Open*, *Close*}.[1].

Despite the small scale of our pilot dataset, we observed some interesting behaviors shared between trials. For example, at the beginning of the object search trials the *Assistant* would specify the target object and the *Controller* would confirm. Additionally, as the task progresses, both roles would describe behaviors, beliefs about the environment and location of objects, and visual observations. We codified these into a set of preliminary intent types; some examples are given in the figure above. Using this pilot dataset, we have started to explore the development of an autonomous agent (*Controller*), both modular and end-to-end that can plan actions for this task.

As mentioned in the introduction, we experimented with both speech-based dialogue and text-based dialogue, using the recording and chat features of Zoom. With speech, participants typically engage in frequent back-and-forth, as the *Controller* controls the agent. Such exchanges involve discussing, for example, the scene and possible target locations. Participants report that when using text, the *Controller* must decide between controlling the agent in AI2-THOR versus typing in the chat. Consequently, they would try to search for the object themselves without interacting with the *Assistant*, who, as a result, finds it difficult to tell if their input is being considered by the *Controller*. This suggests collecting dialogue data through text is unnatural and misaligned with our goal.

8.5 Discussion & Next Steps

Though truthful to the task, our pilot data collection procedure is currently not scalable. We plan to implement a system that can be deployed on the crowdsourcing platform Amazon Mechanical Turk (AMT), to pair up Turkers to participate in the task entirely through their web browsers for accessibility. AMT a powerful platform, yet not designed for multi-user tasks. Due to audio communication and running AI2-THOR servers, we face a more difficult situation than Das et al. (2017) who had to implemented a live chatbot on AMT. We also need solutions to scalable and accurate transcription of the collected audio as well as intent labling. We seek suggestions for strategies to collect such data at scale. In terms of evaluation, we believe both experiment with simulated assistants and real human assistants are necessary. For the simulated assistant, we are considering an oracle agent that communicates using template-based language. The goal of this simulated agent is to facilitate efficient and repeatable evaluations during algorithm development for the embodied dialogue agent, which could be a long-term effort. Ultimately, the agent should be deployed to perform the task with real human subjects. We plan to consider objective metrics for both object search performance (e.g., success rate and discounted total return and dialogue quality (Venkatesh et al., 2018), and, eventually, subjective metrics such as naturalness (Hung

1. We first experimented with a rotation angle of 90° following (Gordon et al., 2018; X. Ye & Yang, 2021), but experienced sudden jumps that are unnatural as felt by the participants. Therefore, we switch to 45°, also used by some existing works (Wortsman et al., 2019; Qiu et al., 2020)

et al., 2009). We believe finding solutions to scalable speech-based dialogue data collection for embodied tasks and plausible evaluation protocol are daunting, yet unavoidable challenges towards future collaborative robots.

THIS IS THE END OF THIS CHAPTER.

CHAPTER 9

Conclusion

THis thesis was motivated by the practical challenges for object search and the goal of making object search an off-the-shelf ability for robots. It argues for using POMDP to model object search and exploiting structure in the human world and human-robot interaction to achieve practical and effective object search systems.

We have provided a few examples in supporting this argument: octrees for 3D multi-object search, spatial correlations for searching for hard-to-detect objects, and structure in spatial language for searching with a hint. We were able to demonstrate our proposed algorithms in realistic simulators and on real robots. Notably, we built GenMOS, the first robot-independent, environment agnostic system for object search in 3D, and integrated it with three different robots in different environments. As a preliminary work, we also identified patterns in the intents behind utterances for dialogue object search.

We have also proposed taxonomies for three main aspects of object search studies: problem settings, solution methods, and the systems and applied them in the literature review of more than 125 papers related to object search.

Future Work

Chapter 3 has provided a generic POMDP model for object search, which has been the parent model for all the POMDP models we developed for specific problems. However, the problems studied in this thesis are still basic in the sense that they have involved static target objects and no environment interaction. Achieving generalized object search beyond the basic setting is the big open problem and ultimate pursuit in this field. Additionally, it is beneficial to study ways to learn models of the environment which can be used for object search planning. This is likely more useful for search involving environment interactions.

If the question is whether a generic POMDP for object search model in Chapter 3 is extensible to handle the additional challenges, there is both potential to explore and foreseeable hurdles to worry. In terms of potential, the transition function in the POMDP can be easily extended to consider motion of target objects. Besides, the high-level LOOK action can be extended to be an abstraction over not only navigation-based search actions, but also manipulation-based, which all serve the purpose of perceiving some part of the environment. However, trouble arises as the consequences of manipulation to the environment may become intractable to model. For example, suppose that a book might be underneath a bed. In this case, a good way to search underneath the bed is not to bend over and look, but to find a tool, such as a stick, to sweep underneath the bed. A robot that can reason at this level faces the challenge of considering consequences of sweeping in the environment. How to make a robot that can perform search while being aware of the consequence of its own actions to the environment? This is a valuable direction of future work.

THIS IS THE END OF THIS CHAPTER.

APPENDIX A

The pomdp_py library

In this appendix, we present pomdp_py, a general purpose Partially Observable Markov Decision Process (POMDP) library written in Python and Cython. Existing POMDP libraries often hinder accessibility and efficient prototyping due to the underlying programming language or interfaces, and require extra complexity in software toolchain to integrate with robotics systems. pomdp_py features simple and comprehensive interfaces capable of describing large discrete or continuous (PO)MDP problems. Here, we summarize the design principles and describe in detail the programming model and interfaces in pomdp_py. We also describe intuitive integration of this library with ROS (Robot Operating System), which enabled our torso-actuated robot to perform object search in 3D. Finally, we note directions to improve and extend this library for POMDP planning and beyond.

1 Introduction

Partially Observable Markov Decision Processes (POMDP) are a sequential decision-making framework suitable to model many robotics problems, from localization and mapping (Ocaña et al., 2005) to human-robot interaction (Whitney et al., 2017). Early efforts in developing tools for POMDPs attempt to separate solvers from domain description by creating specialized file formats to specify POMDPs (Cassandr, 2003; APPL, 2009), which are not designed for large and complex problems. Among libraries under active development, Approximate POMDP Planning Toolkit (APPL) (Somani et al., 2013) and AI-Toolbox (Bargiacchi, 2014) are implemented in C++ and contain numerous solvers. However, the learning curve for these libraries is steep as C++ is less accessible to current researchers in general compared to Python (Virtanen et al., 2020). POMDPs.jl (Egorov et al., 2017) is a POMDP library with a suite of solvers and domains, written in Julia. Though

promising, Julia has yet to achieve a wide recognition and creates language barrier for many researchers. POMDPy (Emami et al., 2015) is implemented purely in Python. Yet with an original focus on POMCP implementation, it assumes a blackbox world model in its POMDP interface, limiting its extensibility. Finally, a promising toolchain is to use Relational Dynamic Influence Diagram Language (RDDL) (Sanner, 2010) to describe factored POMDPs and solve them via ROSPlan (Cashmore et al., 2015), recently demonstrated for object fetching (Canal et al., 2019). Nevertheless, using this set of tools adds overhead of using a classical fluent-based planning paradigm, which is not required to describe and solve POMDPs in general.

This leads to our belief that there lacks a POMDP library with simple interfaces that brings together both accessibility and performance. We address this demand by presenting `pomdp_py`, a framework to build and solve POMDP problems written in Python and Cython (Behnel et al., 2011). It features simple and comprehensive interfaces to describe POMDP or MDP problems, and can be integrated with ROS (Quigley et al., 2009) intuitively through `rospy`. In the rest of this appendix, we first review POMDPs, then illustrate the design principles and key features of `pomdp_py`, including integration with ROS. Finally, we note directions to improve and extend this library, in hope of cultivating an open-source community for POMDP-related research and development. The documentation of `pomdp_py` is available at: `https://h2r.github.io/pomdp-py/html/`. Tutorials on example domains can be found in the documentation. This library is currently actively developed as we continue our POMDP-related research.

2 POMDPs

POMDPs (Kaelbling et al., 1998) model sequential decision making problems where the agent must act under partial observability of the environment state. Refer to Background (Section 2.2.2, page 25) for a formal introduction of POMDPs.

Solvers. Most recent POMDP solvers are *anytime algorithms* (Zilberstein, 1996; Ross et al., 2008), due to the intractable computation required to solve POMDPs exactly (Madani et al., 1999). There are currently two major camps of anytime solvers, point-based methods (Kurniawati et al., 2008; Shani et al., 2013) which approximates the belief space by a set of reachable α-vectors, and Monte-Carlo tree search-based methods (Silver & Veness, 2010; Somani et al., 2013) that explores a subset of future action-observation sequences.

Currently, `pomdp_py` contains an implementation of POMCP and PO-UCT (Silver & Veness, 2010), as well as a naive exact value iteration algorithm without pruning (Kaelbling et al., 1998). The interfaces of the library support implementation of other algorithms; We hope to cultivate a community to implement more solvers or create bridges between `pomdp_py` and other libraries.

Belief representation The partial observability of environment state implies that the agent has to maintain a posterior distribution over possible states (Thrun et al., 2005). The

agent should update this belief distribution through new actions and observations. A tabular belief representation requires nested iterations over the state space to update the belief, which is computationally intractable in large domains. Particle belief representation is a simple and scalable belief representation which is updated through matching simulated and real observations exactly (Silver & Veness, 2010). Different schemes of weighted particles have been proposed to handle large or continuous observation spaces where exact matching results in particle depletion (Sunberg & Kochenderfer, 2018a; Garg et al., 2019).

`pomdp_py` does not commit to any specific belief representation. It provides implementations for basic belief representations and update algorithms, including tabular, particles, and multi-variate Gaussians, but more importantly allows the user to create their own new or problem-specific representation, according to the interface of a generative probability distribution.

3 Design Philosphy

Our goal is to design a framework that allows simple and intuitive ways of defining POMDPs at scale for both discrete and continuous domains, as well as solving them either through planning or through reinforcement learning. In addition, we implement this framework in Python and Cython to improve accessibility and prototyping efficiency without losing orders of magnitude in performance (Behnel et al., 2011; Smith, 2015). We summarize the design principles behind `pomdp_py` below:

- Fundamentally, we view the POMDP scenario as the interaction between an *agent* and the *environment*, through a few important generative probability distributions (π, T, O, R or blackbox model G).

- The agent and the environment may carry different models to support learning, since for real-world problems especially in robotics, the agent generally does not know the true transition or reward models underlying the environment, and only acts based on a simplified or estimated model.

- The POMDP domain could be very large or continuous, thus explicit enumeration of elements in the spaces should be optional.

- The representation of belief distribution is decided by the user and can be customized, as long as it follows the interface of a generative distribution.

- Models can be reused across different POMDP problems. Extensions of the POMDP framework to, for example, decentralized POMDPs, should also be possible by building upon existing interfaces.

4 Programming Model and Features

The basis of pomdp_py is a set of simple interfaces that collectively form a framework for building and solving POMDPs.

When defining a POMDP, one first defines the *domain* by implementing the `State`, `Action`, `Observation` interfaces. The only required functions for each interface are `__eq__` and `__hash__`. For example, the interface for `State` is simply[1]:

```
class State:
    def __eq__(self, other):
        raise NotImplementedError
    def __hash__(self):
        raise NotImplementedError
```

Next, one defines the *models* by implementing the interfaces `TransitionModel`, `ObservationModel`, etc. Note that one may define a different transition and reward model for the agent than the environment (e.g. for learning). One also defines a `PolicyModel` which (1) determines the action space at a given history or state, and (2) samples an action from this space according to some probability distribution. Implementing these models involves implementing the `probability`, `sample` and `argmax` functions. For example, the interface for `ObservationModel`, modeling $O(s', a, o) = \Pr(o|s', a)$, is:

```
class ObservationModel:
    def probability(self, observation, next_state, action):
        """Returns the probability Pr(o|s',a)."""
        raise NotImplementedError
    def sample(self, next_state, action):
        """Returns a sample o ~ Pr(o|s',a)."""
        raise NotImplementedError
    def argmax(self, next_state, action):
        """Returns o* = argmax_o Pr(o|s',a)."""
        raise NotImplementedError
    def get_all_observations(self, *args, **kwargs):
        """Returns a set of all possible
        observations, if feasible."""
        raise NotImplementedError
```

It is up to the user to choose which subset of these functions to implement, depending on the domain. These interfaces aim to remind users the essence of models in POMDPs.

1. Note that the code snippets here are modified or shortened slightly for display purposes. Please refer to the code on github: https://github.com/h2r/pomdp-py/

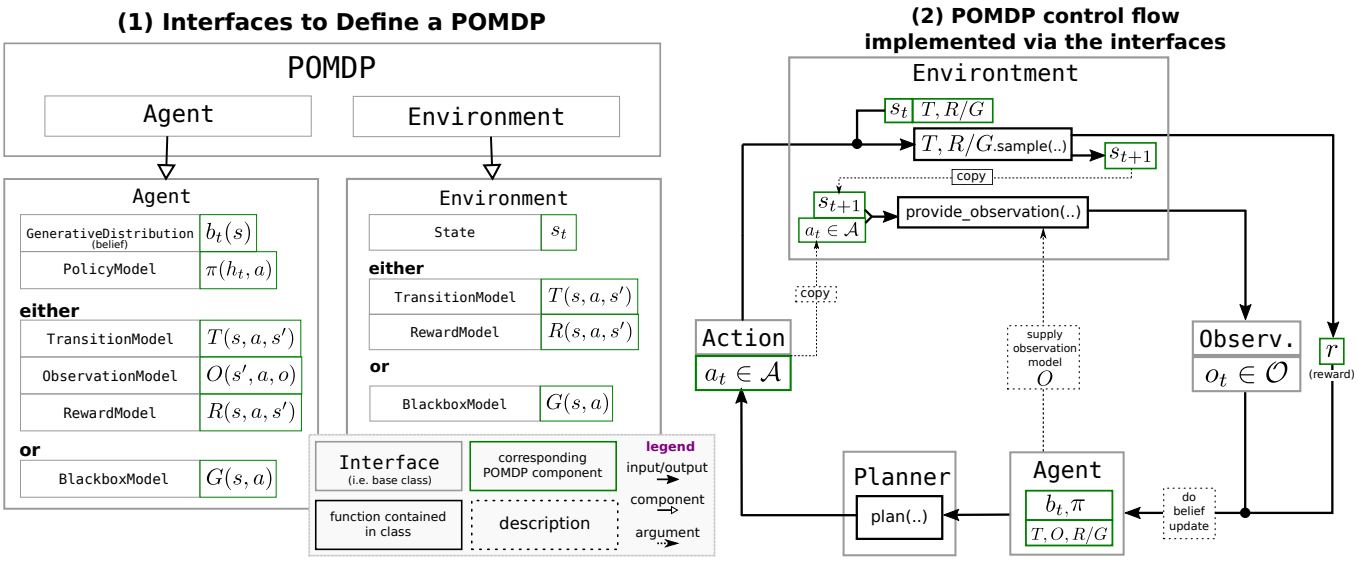


Figure A.1: (1) Core Interfaces in the pomdp_py framework; (2) POMDP control flow implemented through interaction between the core interfaces.

To instantiate a POMDP, one provides parameters for the models, the initial state of the environment, and the initial belief of the agent. For the Tiger problem[2] (Kaelbling et al., 1998), for example,

```
s0 = random.choice(list(TigerProblem.STATES))
b0 = pomdp_py.Histogram({State("tiger-left"): 0.5,
        State("tiger-right"): 0.5})
tiger_problem = TigerProblem(..., s0, b0)
```

Here, `TigerProblem` is a `POMDP` whose constructor takes care of initializing the `Agent` and `Environment` objects, and is instantiated by parameters (omitted), initial state and belief. Note that it is entirely optional to explicitly define a problem class such as `TigerProblem` in order to program the POMDP control flow, discussed below.

To solve a POMDP with pomdp_py, here is the control flow one should implement that contains the basic steps:

1. Create a planner (`Planner`), i.e. a POMDP solver.
2. Agent plans an action $a \in \mathcal{A}$ through the planner.
3. Environment state transitions $s_t \rightarrow s_{t+1}$ according to its transition model.

2. https://h2r.github.io/pomdp-py/html/examples.tiger.html

4. Agent receives an observation o_t and reward r_t from the environment.
5. Agent updates history and belief. $h_t, b_t \rightarrow h_{t+1}, b_{t+1}$, where $h_{t+1} = h_t(a_t, o_t)$.
6. Unless *termination condition* is true, repeat steps 2-5.

The `Planner` interface is as follows. The planner may be updated given a real action and a real observation, which is necessary for MCTS-based solvers.

```
class Planner:
    def plan(self, agent):
        """The agent carries the information:
        Bt, ht, O,T,R/G, pi, needed for planning"""
        raise NotImplementedError
    def update(self, agent, action, observation):
        """Updates the planner based on real action
        and observation. Updates the agent belief
        accordingly if necessary. """
        pass
```

Code Organization. In a more complicated problem such as the Light-Dark domain (Platt Jr et al., 2010) or Multi-Object Search with fan-shaped sensors (Wandzel et al., 2019), it may be tricky to organize the code base and be consistent across different problems. Below we provide a recommendation of the package structure to use `pomdp_py` to guide the development and facilitate code sharing:

```
 - domain/
    - state.py        // State
    - action.py       // Action
    - observation.py  // Observation
    - ...
 - models/
    - transition_model.py   // TransitionModel
    - observation_model.py  // ObservationModel
    - reward_model.py       // RewardModel
    - policy_model.py       // PolicyModel
    - ...
- agent/
    - agent.py  // Agent
    - ...
- env/
    - env.py  // Environment
    - ...
 - problem.py  // POMDP
```

The recommendation is to separate code for domain, models, agent and environment, and have simple generic filenames. As in the above tree, files such as `state.py` or `transition_model.py` are self-evident in their role. The `problem.py` file is where the specific implementation of the `POMDP` class is defined, and where the logic of control flow is implemented. Refer to the Multi-Object Search example in the documentation for more detail[3].

Object-Oriented POMDPs. OO-POMDP (Wandzel et al., 2019) is a particular kind of factored POMDP that factors the state and observation spaces into a set of n objects. For instance, $\Pr(s'|s,a) = \prod_i \Pr(s'_i|s,a)$, $i \in \{1, \cdots, n\}$. The belief space is also factored, which allows the belief space to grow linearly instead of exponentially as the number of objects increases. Each object is of a certain class and has a set of attributes. The values of these attributes constitute the state of an object. In `pomdp_py`, we provide interfaces to implement OO-POMDPs, which serves as an example of extending the basic POMDP framework to create another class of model. These interfaces include `OOState`, `OOBelief`, `OOTransitionModel`, etc.

Integration with ROS. ROS (Quigley et al., 2009) is an open-source system that builds a network connecting computing stations and robots, where *nodes* interact with one another through publishing messages or making service requests. It is typical to separate nodes that manage resources and controls the robot from nodes that runs sophisticated algorithms. This is the case of `pomdp_py` as well. The POMDP-related computations can be done on a node that implements the POMDP control flow (see the six steps above). Inside this node, when an action is selected by the `Planner` (step 1), the node can publish a message to the nodes for robot control so that the robot can execute the action (step 2). The environment state automatically updates in the real world as a result of that action (step 3), and the node receives the sensor measurements or other forms of observations through subscribed topics (step 4), and performs belief update (step 5). This process is repeated until termination condition is met (step 6). ROS provides a package `rospy` which eases the integration of the POMDP control flow with the robot system.

5 Summary

We present a POMDP library, named `pomdp_py`, that brings together accessibility to programmers through Python as well as performance through Cython, with an intuitive design and straightforward integration with ROS. The programming model is designed to encourage organized development and code sharing within a community, and it has potential to support research besides POMDP planning, including reinforcement learning, transfer learning, and multi-agent systems.

3. https://h2r.github.io/pomdp-py/html/examples.mos.html

REFERENCES

Abel, D., Dabney, W., Harutyunyan, A., Ho, M. K., Littman, M., Precup, D., & Singh, S. (2021). On the expressivity of markov reward. *Advances in Neural Information Processing Systems*, *34*, 7799–7812.

Ahn, J., Kim, C., & Nam, C. (2022). Coordination of two robotic manipulators for object retrieval in clutter. In *2022 international conference on robotics and automation (icra)* (pp. 1039–1045).

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., ... others (2022). Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Anderson, P., Chang, A., Chaplot, D. S., Dosovitskiy, A., Gupta, S., Koltun, V., ... others (2018). On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*.

Andreopoulos, A., Hasler, S., Wersing, H., Janssen, H., Tsotsos, J. K., & Korner, E. (2010). Active 3d object localization using a humanoid robot. *IEEE Transactions on Robotics*, *27*(1), 47–64.

APPL, P. (2009). *Pomdpx file format (version 1.0)*. https://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/index.php?n=Main.PomdpXDocumentation.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, *34*(6), 26–38.

Åström, K. J. (1965). Optimal control of markov processes with incomplete state information. *Journal of mathematical analysis and applications*, *10*(1), 174–205.

Atanasov, N., Sankaran, B., Le Ny, J., Pappas, G. J., & Daniilidis, K. (2014). Nonmyopic view planning for active object classification and pose estimation. *IEEE Transactions on Robotics*, *30*(5), 1078–1090.

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, *47*(2-3), 235–256.

Aydemir, A., Pronobis, A., Göbelbecker, M., & Jensfelt, P. (2013). Active visual object search in unknown environments using uncertain semantics. *IEEE Transactions on Robotics*, *29*(4), 986–1002.

Aydemir, A., Sjöö, K., Folkesson, J., Pronobis, A., & Jensfelt, P. (2011). Search in the real world: Active visual object search based on spatial relations. In *2011 ieee international conference on robotics and automation* (pp. 2818–2824).

Bai, A., Srivastava, S., & Russell, S. J. (2016). Markovian state and action abstractions for MDPs via hierarchical MCTS. In *Ijcai*.

Bargiacchi, E. (2014). *Ai-toolbox: A c++ framework for mdps and pomdps with python bindings*. https://github.com/Svalorzen/AI-Toolbox.

Batra, D., Gokaslan, A., Kembhavi, A., Maksymets, O., Mottaghi, R., Savva, M., ... Wijmans, E. (2020). Objectnav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*.

Battistuzzi, L., Recchiuto, C. T., & Sgorbissa, A. (2021). Ethical concerns in rescue robotics: a scoping review. *Ethics and Information Technology*, *23*(4), 863–875.

Baxter, J. L., Burke, E., Garibaldi, J. M., & Norman, M. (2007). Multi-robot search and rescue: A potential field based approach. In *Autonomous robots and agents* (pp. 9–16). Springer.

Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., & Smith, K. (2011). Cython: The best of both worlds. *Computing in Science & Engineering*, *13*(2), 31–39.

Bejjani, W., Agboh, W. C., Dogar, M. R., & Leonetti, M. (2021). Occlusion-aware search for object retrieval in clutter. In *2021 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 4678–4685).

Bekkerman, I., & Tabrikian, J. (2006). Target detection and localization using MIMO radars and sonars. *IEEE Transactions on Signal Processing*, *54*(10), 3873–3883.

Bisk, Y., Shih, K., Choi, Y., & Marcu, D. (2018). Learning interpretable spatial operations in a rich 3D blocks world. In *Proceedings of the aaai conference on artificial intelligence*.

Bisk, Y., Yuret, D., & Marcu, D. (2016). Natural language communication with robots. In *Proc. of naacl*.

Blukis, V., Brukhim, N., Bennett, A., Knepper, R. A., & Artzi, Y. (2018). Following high-level navigation instructions on a simulated quadcopter with imitation learning. In *Proceedings of the robotics: Science and systems conference*.

Blukis, V., Knepper, R. A., & Artzi, Y. (2020). Few-shot object grounding and mapping for natural language robot instruction following. In *Proceedings of the conference on robot learning*.

Blukis, V., Misra, D., Knepper, R. A., & Artzi, Y. (2018). Mapping navigation instructions to continuous control actions with position-visitation prediction. In *Conference on robot learning*.

Blukis, V., Terme, Y., Niklasson, E., Knepper, R. A., & Artzi, Y. (2019). Learning to map natural language instructions to physical quadcopter control using simulated flight. In *Conference on robot learning (corl)*.

Bono, A., D'Alfonso, L., Fedele, G., & Gazi, V. (2022). A swarm model for target capturing in a polygonal strip. In *2022 8th international conference on control, decision and information technologies (codit)* (Vol. 1, pp. 297–302).

Boston Dynamics Spot. (2019). https://www.bostondynamics.com/products/spot. (Accessed: 2019)

Bourque, F.-A. (2019). Solving the moving target search problem using indistinguishable searchers. *European Journal of Operational Research*, *275*(1), 45-52. Retrieved from https://www.sciencedirect.com/science/article/pii/S0377221718309317 doi: https://doi.org/10.1016/j.ejor.2018.11.006

Bowling, M., Martin, J. D., Abel, D., & Dabney, W. (2022). Settling the reward hypothesis. *arXiv preprint arXiv:2212.10420*.

Briney, C. (2004). State of the industry. *Global Cosmetic Industry*, *172*, 26.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., ... Colton, S. (2012). A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, *4*(1), 1–43.

Caicedo, J. C., & Lazebnik, S. (2015). Active object localization with deep reinforcement learning. In *Proceedings of the ieee international conference on computer vision* (pp. 2488–2496).

Canal, G., Cashmore, M., Krivić, S., Alenyà, G., Magazzeni, D., & Torras, C. (2019). Probabilistic planning for robotics with rosplan. In *Annual conference towards autonomous robotic systems* (pp. 236–250).

Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carrera, A., ... Carreras, M. (2015). Rosplan: Planning in the robot operating system. In *Twenty-fifth international conference on automated planning and scheduling*.

Cassandr, A. R. (2003). *Pomdp file format*. http://www.pomdp.org/code/pomdp-file-spec.html.

Cassandra, A. R., Kaelbling, L. P., & Littman, M. L. (1994). Acting optimally in partially observable stochastic domains. In *Aaai* (Vol. 94, pp. 1023–1028).

Chang, M., Gupta, A., & Gupta, S. (2020). Semantic visual navigation by watching youtube videos. *Advances in Neural Information Processing Systems*, *33*, 4283–4294.

Chaplot, D. S., Gandhi, D., Gupta, A., & Salakhutdinov, R. (2020). Object goal navigation using goal-oriented semantic exploration. In *Neural information processing systems.*

Chen, C. P., Li, H., Wei, Y., Xia, T., & Tang, Y. Y. (2013). A local contrast method for small infrared target detection. *IEEE transactions on geoscience and remote sensing*, *52*(1), 574–581.

Chen, K., de Vicente, J. P., Sepulveda, G., Xia, F., Soto, A., Vazquez, M., & Savarese, S. (2019, June). A behavioral approach to visual navigation with graph localization networks. In *Proceedings of robotics: Science and systems.* FreiburgimBreisgau, Germany. doi: 10.15607/RSS.2019.XV.010

Chen, X., & Lee, S. (2013). Visual search of an object in cluttered environments for robotic errand service. In *2013 ieee international conference on systems, man, and cybernetics* (pp. 4060–4065).

Chitta, S. (2016). Moveit!: an introduction. In *Robot operating system (ros)* (pp. 3–27). Springer.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., ... others (2022). Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Chung, T. H., Hollinger, G. A., & Isler, V. (2011). Search and pursuit-evasion in mobile robotics. *Autonomous robots*, *31*(4), 299–316.

Czyzowicz, J., Georgiou, K., Kranakis, E., Krizanc, D., Narayanan, L., Opatrny, J., & Shende, S. (2016). Search on a line by byzantine robots. *arXiv preprint arXiv:1611.08209*.

Dadgar, M., Jafari, S., & Hamzeh, A. (2016). A pso-based multi-robot cooperation method for target searching in unknown environments. *Neurocomputing*, *177*, 62–74.

Dames, P. M. (2020). Distributed multi-target search and tracking using the phd filter. *Autonomous robots*, *44*(3), 673–689.

Danielczuk, M., Kurenkov, A., Balakrishna, A., Matl, M., Wang, D., Martín-Martín, R., ... Goldberg, K. (2019). Mechanical search: Multi-step retrieval of a target object occluded by clutter. In *2019 International Conference on Robotics and Automation (ICRA)* (pp. 1614–1621).

Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J. M., ... Batra, D. (2017). Visual dialog. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 326–335).

Deitke, M., Han, W., Herrasti, A., Kembhavi, A., Kolve, E., Mottaghi, R., ... others (2020). Robothor: An open simulation-to-real embodied ai platform. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 3164–3174).

Deitke, M., VanderBilt, E., Herrasti, A., Weihs, L., Salvador, J., Ehsani, K., ... others (2022). ProcTHOR: Large-Scale Embodied AI Using Procedural Generation. In *Conference on neural information processing systems (NeurIPS).*

Delmerico, J., Isler, S., Sabzevari, R., & Scaramuzza, D. (2018). A comparison of volumetric information gain metrics for active 3d object reconstruction. *Autonomous Robots*, *42*(2), 197–208.

de Vries, H., Shuster, K., Batra, D., Parikh, D., Weston, J., & Kiela, D. (2018). Talk the walk: Navigating new york city through grounded dialogue. *arXiv preprint arXiv:1807.03367*.

Ding, H., & Castañón, D. (2018). Moving object search with multiple agents. In *2018 IEEE Conference on Decision and Control (CDC)* (pp. 5746–5752).

Diuk, C., Cohen, A., & Littman, M. L. (2008). An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on machine learning* (pp. 240–247).

Dobbie, J. M. (1974). A two-cell model of search for a moving target. *Operations Research*, *22*(1), 79–92.

Dogan, A., & Zengin, U. (2006). Unmanned aerial vehicle dynamic-target pursuit by using probabilistic threat exposure map. *Journal of guidance, control, and dynamics*, *29*(4), 944–954.

Dogar, M. R., Koval, M. C., Tallavajhula, A., & Srinivasa, S. S. (2014). Object search by manipulation. *Autonomous Robots*.

Doumanoglou, A., Kouskouridas, R., Malassiotis, S., & Kim, T.-K. (2016). Recovering 6d object pose and predicting next-best-view in the crowd. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3583–3592).

Druon, R., Yoshiyasu, Y., Kanezaki, A., & Watt, A. (2020). Visual object search by learning spatial context. *IEEE Robotics and Automation Letters*, *5*(2), 1279–1286.

Du, G., Wang, K., Lian, S., & Zhao, K. (2021). Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review. *Artificial Intelligence Review*, *54*(3), 1677–1734.

Duncan, J., & Humphreys, G. W. (1989). Visual search and stimulus similarity. *Psychological review*, *96*(3), 433.

Eaton, J. H., & Zadeh, L. (1962). Optimal pursuit strategies in discrete-state probabilistic systems.

Ebert, J. T., Berlinger, F., Haghighat, B., & Nagpal, R. (2022). A hybrid PSO algorithm for multi-robot target search and decision awareness. In *2022 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 11520–11527).

Eckstein, M. P. (2011). Visual search: A retrospective. *Journal of vision*, *11*(5), 14–14.

Egorov, M., Sunberg, Z. N., Balaban, E., Wheeler, T. A., Gupta, J. K., & Kochenderfer, M. J. (2017). Pomdps. jl: A framework for sequential decision making under uncertainty. *The Journal of Machine Learning Research*, *18*(1), 831–835.

Eismann, M. T., Stocker, A. D., & Nasrabadi, N. M. (2009). Automated hyperspectral cueing for civilian search and rescue. *Proceedings of the IEEE*, *97*(6), 1031–1055.

Elfring, J., Jansen, S., Molengraft, R. v. d., & Steinbuch, M. (2013). Active object search exploiting probabilistic object–object relations. In *Robot soccer world cup* (pp. 13–24).

Emami, P., Hamlet, A. J., & Crane, C. (2015). Pomdpy: An extensible framework for implementing pomdps in python.

Epic Games. (n.d.). *Unreal Engine.* Retrieved from https://www.unrealengine.com

Erickson, R. A. (1964). *Visual search for targets: Laboratory experiments* (Tech. Rep.). NAVAL ORDNANCE TEST STATION CHINA LAKE CA.

Fasola, J., & Mataric, M. J. (2013). Using semantic fields to model dynamic spatial relations in a robot architecture for natural language instruction of service robots. In *Ieee/rsj international conference on intelligent robots and systems* (pp. 143–150).

Fasola, J., & Matarić, M. J. (2013). Using spatial semantic and pragmatic fields to interpret natural language pick-and-place instructions for a mobile service robot. In *International conference on social robotics* (pp. 501–510).

Fidan, B., Dasgupta, S., & Anderson, B. D. (2013). Adaptive range-measurement-based target pursuit. *International Journal of Adaptive Control and Signal Processing*, *27*(1-2), 66–81.

Gan, C., Schwartz, J., Alter, S., Schrimpf, M., Traer, J., De Freitas, J., ... others (2020). Threedworld: A platform for interactive multi-modal physical simulation. *arXiv preprint arXiv:2007.04954*.

Garg, N. P., Hsu, D., & Lee, W. S. (2019). DESPOT-α: Online POMDP planning with large state and observation spaces. In *Robotics: Science and systems.*

Garvey, T. D. (1976). Perceptual strategies for purposive vision. *Thesis Ph.D. Stanford University*.

Gelenbe, E., & Cao, Y. (1998). Autonomous search for mines. *European Journal of Operational Research*, *108*(2), 319–333.

Gervet, T., Chintala, S., Batra, D., Malik, J., & Chaplot, D. S. (2022). Navigating to objects in the real world. *arXiv preprint arXiv:2212.00922*.

Giuliari, F., Castellini, A., Berra, R., Del Bue, A., Farinelli, A., Cristani, M., ... Wang, Y. (2021). Pomp++: Pomcp-based active visual search in unknown indoor environments. In *2021 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 1523–1530).

Goldsmith, S. Y., & Robinett, R. (1998). Collective search by mobile robots using alpha-beta coordination. In *International workshop on collective robotics* (pp. 135–146).

Gordon, D., Kembhavi, A., Rastegari, M., Redmon, J., Fox, D., & Farhadi, A. (2018). Iqa: Visual question answering in interactive environments. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4089–4098).

gPRC documentation. (n.d.). https://grpc.io/docs/. (Accessed: 2022)

Grisetti, G., Stachniss, C., & Burgard, W. (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, *23*(1), 34–46.

Gupta, S., Davidson, J., Levine, S., Sukthankar, R., & Malik, J. (2017). Cognitive mapping and planning for visual navigation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2616–2625).

Harbers, M., de Greeff, J., Kruijff-Korbayová, I., Neerincx, M. A., & Hindriks, K. V. (2017). Exploring the ethical landscape of robot-assisted search and rescue. In *A world with robots* (pp. 93–107). Springer.

Hayward, W. G., & Tarr, M. J. (1995). Spatial language and spatial representation. *Cognition*, *55*(1), 39–84.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the ieee international conference on computer vision* (pp. 2961–2969).

Hemachandra, S., Duvallet, F., Howard, T. M., Roy, N., Stentz, A., & Walter, M. R. (2015). Learning models for following natural language directions in unknown environments. In *Ieee international conference on robotics and automation (icra)* (pp. 5608–5615).

Hereford, J. M., & Siebold, M. A. (2010). Bio-inspired search strategies for robot swarms. *Swarm robotics, from biology to robotics*, 1–27.

Hernandez, A. C., Durner, M., Gomez, C., Grixa, I., Teikmanis, O., Marton, Z.-C., & Barber, R. (2021). Searching for objects in human living environments based on relevant inferred and mined priors. In *2021 european conference on mobile robots (ecmr)* (pp. 1–7).

Hess, W., Kohler, D., Rapp, H., & Andor, D. (2016). Real-time loop closure in 2d lidar slam. In *2016 ieee international conference on robotics and automation (icra)* (pp. 1271–1278).

Ho, Y., Bryson, A., & Baron, S. (1965). Differential games and optimal pursuit-evasion strategies. *IEEE Transactions on Automatic Control*, *10*(4), 385–389.

Hollinger, G., Singh, S., Djugash, J., & Kehagias, A. (2009). Efficient multi-robot search for a moving target. *The International Journal of Robotics Research*, *28*(2), 201–219.

Holzherr, L., Förster, J., Breyer, M., Nieto, J., Siegwart, R., & Chung, J. J. (2021). Efficient multi-scale POMDPs for robotic object search and delivery. In *2021 ieee international conference on robotics and automation (icra)* (pp. 6585–6591).

Honnibal, M., & Montani, I. (2017). *spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.* (To appear)

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., & Burgard, W. (2013). Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, *34*(3), 189–206.

Huang, H., Danielczuk, M., Kim, C. M., Fu, L., Tam, Z., Ichnowski, J., ... Goldberg, K. (2022). Mechanical search on shelves using a novel "bluction" tool. In *2022 international conference on robotics and automation (icra)* (pp. 6158–6164).

Huang, H., Dominguez-Kuhne, M., Satish, V., Danielczuk, M., Sanders, K., Ichnowski, J., ... Goldberg, K. (2021). Mechanical search on shelves using lateral access x-ray. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 2045–2052).

Hung, V., Elvir, M., Gonzalez, A., & DeMara, R. (2009). Towards a method for evaluating naturalness in conversational dialog systems. In *2009 ieee international conference on systems, man and cybernetics* (pp. 1236–1241).

Idrees, I., Reiss, S. P., & Tellex, S. (2020). Robomem: Giving long term memory to robots. *arXiv preprint arXiv:2003.10553*.

Isaza, A. I. A., Lu, J., Bulitko, V., & Greiner, R. (2008). A cover-based approach to multi-agent moving target pursuit. In *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment* (Vol. 4, pp. 54–59).

Ishida, T., & Korf, R. E. (1991). Moving target search. In *Ijcai* (Vol. 91, pp. 204–210).

Izquierdo-Cordova, R., Morales, E. F., Sucar, L. E., & Murrieta-Cid, R. (2016). Searching objects in known environments: Empowering simple heuristic strategies. In *Robot world cup* (pp. 380–391).

Jain, V., Magalhaes, G., Ku, A., Vaswani, A., Ie, E., & Baldridge, J. (2019). Stay on the Path: Instruction Fidelity in Vision-and-Language Navigation. In *Proc. of acl.*

Janner, M., Narasimhan, K., & Barzilay, R. (2018). Representation learning for grounded spatial reasoning. *Transactions of the Association for Computational Linguistics*, *6*, 49–61.

Jocher, G., Stoken, A., Borovec, J., NanoCode012, ChristopherSTAN, Changyu, L., ... Rai, P. (2020, October). *ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements.* Zenodo. Retrieved from https://doi.org/10.5281/zenodo.4154370 doi: 10.5281/zenodo.4154370

Joho, D., Senk, M., & Burgard, W. (2011). Learning search heuristics for finding objects in structured environments. *Robotics and Autonomous Systems*, *59*(5), 319–328.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial intelligence*, *101*(1-2), 99–134.

Kam, H. R., Lee, S.-H., Park, T., & Kim, C.-H. (2015). Rviz: a toolkit for real domain data visualization. *Telecommunication Systems*, *60*(2), 337–345.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International conference on learning representations.*

Kinova MOVO. (2017). https://kinovarobotics.github.io/kinova-movo/Concepts/c_movo_hardware_overview.html. (Accessed: 2017)

Kochenderfer, M. J. (2015). *Decision making under uncertainty: theory and application*. MIT press.

Kocsis, L., & Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*.

Kollar, T., & Roy, N. (2009). Utilizing object-object and object-scene context when planning to find things. In *IEEE International Conference on Robotics and Automation* (pp. 2168–2173).

Kollar, T., Tellex, S., Roy, D., & Roy, N. (2010). Toward understanding natural language directions. In *2010 5th acm/ieee international conference on human-robot interaction (hri).*

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., ... Farhadi, A. (2017). AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv preprint arXiv:1712.05474*.

Koopman, B. O. (1956). The theory of search. II. Target detection. *Operations research*, *4*(5), 503–531.

Kulich, M., Juchelka, T., & Přeučil, L. (2015). Comparison of exploration strategies for multi-robot search. *Acta Polytechnica*, *55*(3), 162–168.

Kurenkov, A., Taglic, J., Kulkarni, R., Dominguez-Kuhne, M., Garg, A., Martín-Martín, R., & Savarese, S. (2020). Visuomotor mechanical search: Learning to retrieve target objects in clutter. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 8408–8414).

Kurniawati, H. (2022). Partially observable markov decision processes and robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, *5*.

Kurniawati, H., Hsu, D., & Lee, W. S. (2008). Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems* (Vol. 2008).

Landau, B., & Jackendoff, R. (1993). "what" and "where" in spatial language and spatial cognition. *Behavioral and Brain Sciences*, *16*, 217-238. doi: 10.1017/S0140525X00029733

Lauri, M., Hsu, D., & Pajarinen, J. (2022). Partially observable markov decision processes in robotics: A survey. *IEEE Transactions on Robotics*.

Li, J. K., Hsu, D., & Lee, W. S. (2016). Act to see and see to act: POMDP planning for objects search in clutter. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Li, L., Walsh, T. J., & Littman, M. L. (2006). Towards a unified theory of state abstraction for MDPs. In *Isaim.*

Li, Y., Ma, Y., Huo, X., & Wu, X. (2022). Remote object navigation for service robots using hierarchical knowledge graph in human-centered environments. *Intelligent Service Robotics*, *15*(4), 459–473.

Liang, Y., Chen, B., & Song, S. (2021). Sscnav: Confidence-aware semantic scene completion for visual semantic navigation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 13194–13200).

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *European Conference on Computer Vision* (pp. 740–755).

Lin, Y.-C., Wei, S.-T., Yang, S.-A., & Fu, L.-C. (2015). Planning on searching occluded target object with a mobile robot manipulator. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3110–3115).

Littman, M. L. (1996). *Algorithms for sequential decision-making*. Brown University.

Littman, M. L. (2009). A tutorial on partially observable markov decision processes. *Journal of Mathematical Psychology*, *53*(3), 119–125.

Liu, M.-Y., Tuzel, O., Veeraraghavan, A., Taguchi, Y., Marks, T. K., & Chellappa, R. (2012). Fast object localization and pose estimation in heavy clutter for robotic bin picking. *The International Journal of Robotics Research*, *31*(8), 951–973.

Liu, W., Bansal, D., Daruna, A. A., & Chernova, S. (2021). Learning instance-level n-ary semantic knowledge at scale for robots operating in everyday environments. In *Robotics: Science and systems.*

Loghmani, M. R., Patten, T., & Vincze, M. (2018). Towards socially assistive robots for elderly: An end-to-end object search framework. In *2018 ieee international conference on pervasive computing and communications workshops (percom workshops).*

Lorbach, M., Höfer, S., & Brock, O. (2014). Prior-assisted propagation of spatial information for object search. In *2014 ieee/rsj international conference on intelligent robots and systems.*

Lu, Y., Wang, Z., Tang, Z., & Javidi, T. (2018). Target localization with drones using mobile cnns. In *2018 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 2566–2573).

Luo, W., Sun, P., Zhong, F., Liu, W., Zhang, T., & Wang, Y. (2019). End-to-end active object tracking and its real-world deployment via reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*, *42*(6), 1317–1332.

Macenski, S., Foote, T., Gerkey, B., Lalancette, C., & Woodall, W. (2022). Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, *7*(66), eabm6074. Retrieved from https://www.science.org/doi/abs/10.1126/scirobotics.abm6074 doi: 10.1126/scirobotics.abm6074

Macenski, S., Martín, F., White, R., & Clavero, J. G. (2020). The marathon 2: A navigation system. In *2020 IEEE/RSJ international conference on intelligent robots and systems (iros)* (pp. 2718–2725).

Madani, O., Hanks, S., & Condon, A. (1999). On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems. In *Aaai/iaai* (pp. 541–548).

Majid, A., Bowerman, M., Kita, S., Haun, D. B., & Levinson, S. C. (2004). Can language restructure cognition? The case for space. *Trends in cognitive sciences*, *8*(3), 108–114.

Mangel, M. (1981). Search for a randomly moving object. *SIAM Journal on Applied Mathematics*, *40*(2), 327–338.

Marjovi, A., Nunes, J. G., Marques, L., & De Almeida, A. (2009). Multi-robot exploration and fire searching. In *2009 ieee/rsj international conference on intelligent robots and systems* (pp. 1929–1934).

Mayo, B., Hazan, T., & Tal, A. (2021). Visual navigation with spatial attention. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 16898–16907).

Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, *2014*(239), 2.

Moldovan, B., & De Raedt, L. (2014). Occluded object search by relational affordances. In *2014 ieee international conference on robotics and automation (icra)* (pp. 169–174).

Monsó, P., Alenyà, G., & Torras, C. (2012). Pomdp approach to robotized clothes separation. In *2012 ieee/rsj international conference on intelligent robots and systems* (pp. 1324–1329).

Montemerlo, M., Roy, N., & Thrun, S. (2003). Perspectives on standardization in mobile robot programming: The carnegie mellon navigation (carmen) toolkit. In *Proceedings 2003 ieee/rsj international conference on intelligent robots and systems (iros 2003)(cat. no. 03ch37453)* (Vol. 3, pp. 2436–2441).

Mouromtsev, D., Kovriguina, L., Emelyanov, Y., Pavlov, D., & Shipilo, A. (2015). From spoken language to ontology-driven dialogue management. In *International conference on text, speech, and dialogue* (pp. 542–550).

Mousavi, S. S., Schukat, M., & Howley, E. (2016). Deep reinforcement learning: an overview. In *Proceedings of sai intelligent systems conference* (pp. 426–440).

Mousavian, A., Toshev, A., Fišer, M., Košecká, J., Wahid, A., & Davidson, J. (2019). Visual representations for semantic target driven navigation. In *2019 international conference on robotics and automation (icra)* (pp. 8846–8852).

Mukherjee, K., Gupta, S., Ray, A., & Phoha, S. (2011). Symbolic analysis of sonar data for underwater target detection. *IEEE Journal of Oceanic Engineering*, *36*(2), 219–230.

Nam, C., Lee, J., Cho, Y., Lee, J., Kim, D. H., & Kim, C. (2019). Planning for target retrieval using a robotic manipulator in cluttered and occluded environments. *arXiv preprint arXiv:1907.03956*.

Nguyen, T., Gopalan, N., Patel, R., Corsaro, M., Pavlick, E., & Tellex, S. (2022). Affordance-based robot object retrieval. *Autonomous Robots*, *46*(1), 83–98.

Nie, X., Wong, L. L., & Kaelbling, L. P. (2016). Searching for physical objects in partially known environments. In *2016 ieee international conference on robotics and automation (icra)*.

Nomatsu, M., Suganuma, Y., Yui, Y., & Uchimura, Y. (2015). Development of an autonomous mobile robot with self-localization and searching target in a real environment. *Journal of Robotics and Mechatronics*, *27*(4), 356–364.

Novkovic, T., Pautrat, R., Furrer, F., Breyer, M., Siegwart, R., & Nieto, J. (2019). Object finding in cluttered scenes using interactive perception. *arXiv preprint arXiv:1911.07482*.

Ocaña, M., Bergasa, L. M., Sotelo, M., & Flores, R. (2005). Indoor robot navigation using a pomdp based on wifi and ultrasound observations. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2592–2597).

O'Keefe, J., & Burgess, N. (1996). Geometric determinants of the place fields of hippocampal neurons. *Nature*, *381*, 425–428.

OpenStreetMap contributors. (2017). *Planet dump retrieved from https://planet.osm.org* .

O'rourke, J., et al. (1987). *Art gallery theorems and algorithms* (Vol. 57). Oxford University Press Oxford.

Papadimitriou, C. H., & Tsitsiklis, J. N. (1987). The complexity of markov decision processes. *Mathematics of operations research*, *12*(3), 441–450.

Park, J. J., Florence, P., Straub, J., Newcombe, R., & Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 165–174).

Patki, S., Fahnestock, E., Howard, T. M., & Walter, M. R. (2020). Language-guided semantic mapping and mobile manipulation in partially observable environments. In *Conference on robot learning*.

Pineau, J., Gordon, G., Thrun, S., et al. (2003). Point-based value iteration: An anytime algorithm for pomdps. In *Ijcai* (Vol. 3, pp. 1025–1032).

Platt Jr, R., Tedrake, R., Kaelbling, L., & Lozano-Perez, T. (2010). Belief space planning assuming maximum likelihood observations.

Pollock, S. M. (1970). A simple model of search for a moving target. *Operations Research*, *18*(5), 883–903.

Prechelt, L. (1998). Early stopping-but when? In *Neural networks: Tricks of the trade* (pp. 55–69). Springer.

Qiu, Y., Pal, A., & Christensen, H. I. (2020). Learning hierarchical relationships for object-goal navigation. In *2020 conference on robot learning (corl)*.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... Ng, A. Y. (2009). Ros: an open-source robot operating system. In *Icra workshop on open source software* (Vol. 3, p. 5).

Rad, M., & Lepetit, V. (2017). Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the ieee international conference on computer vision* (pp. 3828–3836).

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... others (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning* (pp. 8748–8763).

Radmard, S., & Croft, E. A. (2017). Active target search for high dimensional robotic systems. *Autonomous Robots*, *41*(1), 163–180.

Ramakrishnan, S. K., Gokaslan, A., Wijmans, E., Maksymets, O., Clegg, A., Turner, J., ... others (2021). Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *arXiv preprint arXiv:2109.08238*.

Rasouli, A. (2015). Attention and sensor planning in autonomous robotic visual search.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the ieee conference on computer vision and pattern recognition.*

Rizk, Y., Awad, M., & Tunstel, E. W. (2019). Cooperative heterogeneous multi-robot systems: A survey. *ACM Computing Surveys (CSUR)*, *52*(2), 1–31.

Robin, C., & Lacroix, S. (2016). Multi-robot target detection and tracking: taxonomy and survey. *Autonomous Robots*, *40*(4), 729–760.

Ross, S., Pineau, J., Paquet, S., & Chaib-Draa, B. (2008). Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, *32*, 663–704.

Roy, M., Zheng, K., Liu, Jason, & Tellex, S. (2021). Dialogue Object Search. In *RSS Workshop on Robotics for People (R4P): Perspectives on Interaction, Learning and Safety*. Retrieved from https://arxiv.org/pdf/2107.10653.pdf (Extended Abstract)

Roy, N., Gordon, G., & Thrun, S. (2005). Finding approximate pomdp solutions through belief compression. *Journal of artificial intelligence research*, *23*, 1–40.

Rybski, P. E., Larson, A. C., Metcalf, H., Skyllingstad, D., Veeraraghavan, H., & Gini, M. L. (2002). Mindart: A multi-robot search and retrieval system. In *Aaai mobile robot competition* (pp. 83–91).

Sadeghi, M., Behnia, F., & Amiri, R. (2020). Optimal sensor placement for 2-d range-only target localization in constrained sensor geometry. *IEEE Transactions on Signal Processing*, *68*, 2316–2327.

Saidi, F., Stasse, O., Yokoi, K., & Kanehiro, F. (2007). Online object search with a humanoid robot. In *2007 ieee/rsj international conference on intelligent robots and systems* (pp. 1677–1682).

Sanner, S. (2010). Relational dynamic influence diagram language (rddl): Language description. *Unpublished ms. Australian National University*, *32*.

Sarmiento, A., Murrieta, R., & Hutchinson, S. A. (2003). An efficient strategy for rapidly finding an object in a polygonal world. In *Proceedings 2003 ieee/rsj international conference on intelligent robots and systems (iros 2003)(cat. no. 03ch37453).*

Sarmiento, A., Murrieta-Cid, R., & Hutchinson, S. (2004). A multi-robot strategy for rapidly searching a polygonal environment. In *Ibero-american conference on artificial intelligence* (pp. 484–493).

Schmalstieg, F., Honerkamp, D., Welschehold, T., & Valada, A. (2022). Learning long-horizon robot exploration strategies for multi-object search in continuous action spaces. *arXiv preprint arXiv:2205.11384*.

Schmid, J. F., Lauri, M., & Frintrop, S. (2019). Explore, approach, and terminate: Evaluating subtasks in active visual object search based on deep reinforcement learning. In *2019 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 5008–5013).

Shah, D., Osinski, B., Ichter, B., & Levine, S. (2022). Robotic Navigation with Large Pre-Trained Models of Language, Vision, and Action.. Retrieved from https://arxiv.org/abs/2207.04429

Shah, S., Dey, D., Lovett, C., & Kapoor, A. (2017). Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics.*

Shani, G., Pineau, J., & Kaplow, R. (2013). A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*.

Shapiro, A. (2003). Monte Carlo sampling methods. *Handbooks in operations research and management science*, *10*, 353–425.

Sharkey, N., & Sharkey, A. (2011). 17 the rights and wrongs of robot care. *Robot ethics: The ethical and social implications of robotics*, 267.

Sharma, P., Torralba, A., & Andreas, J. (2022). Skill induction and planning with latent language. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 1713–1726).

Sharma, R., Kothari, M., Taylor, C. N., & Postlethwaite, I. (2010). Cooperative target-capturing with inaccurate target information. In *Proceedings of the 2010 american control conference* (pp. 5520–5525).

Shirsat, A., Elamvazhuthi, K., & Berman, S. (2020). Multi-robot target search using probabilistic consensus on discrete markov chains. In *2020 ieee international symposium on safety, security, and rescue robotics (ssrr)* (pp. 108–115).

Shkurti, F., Kakodkar, N., & Dudek, G. (2018). Model-based probabilistic pursuit via inverse reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 7804–7811).

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., ... Fox, D. (2020). Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 10740–10749).

Shubina, K., & Tsotsos, J. K. (2010). Visual search for an object in a 3d environment using a mobile robot. *Computer Vision and Image Understanding*, *114*(5), 535–547.

Shusterman, A., & Li, P. (2016). Frames of reference in spatial language acquisition. *Cognitive psychology*, *88*, 115–161.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... others (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, *362*(6419), 1140–1144.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... others (2017). Mastering the game of go without human knowledge. *nature*, *550*(7676), 354–359.

Silver, D., Singh, S., Precup, D., & Sutton, R. S. (2021). Reward is enough. *Artificial Intelligence*, *299*, 103535.

Silver, D., & Veness, J. (2010). Monte-Carlo planning in large POMDPs. In *Advances in neural information processing systems.*

Sjöö, K., Aydemir, A., & Jensfelt, P. (2012). Topological spatial relations for active visual search. *Robotics and Autonomous Systems*, *60*(9), 1093–1107.

Smallwood, R. D., & Sondik, E. J. (1973). The optimal control of partially observable markov processes over a finite horizon. *Operations research*, *21*(5), 1071–1088.

Smith, K. W. (2015). *Cython: A guide for python programmers*. " O'Reilly Media, Inc.".

Somani, A., Ye, N., Hsu, D., & Lee, W. S. (2013). DESPOT: Online POMDP planning with regularization. In *Advances in neural information processing systems* (pp. 1772–1780).

Sondik, E. J. (1971). *The optimal control of partially observable markov processes*. Stanford University.

Song, T., Huo, X., & Wu, X. (2020). A two-stage method for target searching in the path planning for mobile robots. *Sensors*, *20*(23), 6919.

Spaan, M. T., & Vlassis, N. (2005). Perseus: Randomized point-based value iteration for pomdps. *Journal of artificial intelligence research*, *24*, 195–220.

Sprute, D., Pörtner, A., Rasch, R., Battermann, S., & König, M. (2017). Ambient assisted robot object search. In *International conference on smart homes and health telematics.*

Sticht, D., Vincent, T., & Schultz, D. (1975). Sufficiency theorems for target capture. *Journal of Optimization Theory and Applications*, *17*(5), 523–543.

Stone, L. D. (1976). *Theory of optimal search*. Elsevier.

Stone, L. D. (1979). Necessary and sufficient conditions for optimal search plans for moving targets. *Mathematics of Operations Research*, *4*(4), 431–440.

Şucan, I. A., Moll, M., & Kavraki, L. E. (2012, December). The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, *19*(4), 72–82. (https://ompl.kavrakilab.org) doi: 10.1109/MRA.2012.2205651

Sun, J., Li, B., Jiang, Y., & Wen, C.-y. (2016). A camera-based target detection and positioning uav system for search and rescue (SAR) purposes. *Sensors*, *16*(11), 1778.

Sunberg, Z. N., & Kochenderfer, M. J. (2018a). Online algorithms for POMDPs with continuous state, action, and observation spaces. In *International conference on automated planning and scheduling.*

Sunberg, Z. N., & Kochenderfer, M. J. (2018b). Online algorithms for POMDPs with continuous state, action, and observation spaces. In *Twenty-eighth international conference on automated planning and scheduling.*

Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, *112*(1-2), 181–211.

Svensson, J., & Vegh, R. (2011). *A target-capturing problem based on a cyclic pursuit strategy.*

Tang, H., Sun, W., Lin, A., Xue, M., & Zhang, X. (2021). A gwo-based multi-robot cooperation method for target searching in unknown environments. *Expert Systems with Applications*, *186*, 115795.

Tellex, S., Kollar, T., Dickerson, S., Walter, M., Banerjee, A., Teller, S., & Roy, N. (2011). Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 25).

Thomason, J., Murray, M., Cakmak, M., & Zettlemoyer, L. (2019). Vision-and-dialog navigation. In *Conference on robot learning.*

Thomason, J., Murray, M., Cakmak, M., & Zettlemoyer, L. (2020). Vision-and-dialog navigation. In *Conference on robot learning* (pp. 394–406).

Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic robotics.

Tian, Y., Qi, H., & Wang, X. (2002). Target detection and classification using seismic signal processing in unattended ground sensor systems. In *Ieee international conference on acoustics speech and signal processing* (Vol. 4, pp. 4172–4172).

Tsotsos, J. K. (1992). On the relative complexity of active vs. passive visual search. *International journal of computer vision*, *7*(2), 127–141.

Tsotsos, J. K., Verghese, G., Dickinson, S., Jenkin, M., Jepson, A., Milios, E., ... others (1998). Playbot a visually-guided robot for physically disabled children. *Image and vision computing*, *16*(4), 275–292.

Tsuru, M., Escande, A., Tanguy, A., Chappellet, K., & Harad, K. (2021). Online object searching by a humanoid robot in an unknown environment. *IEEE Robotics and Automation Letters*, *6*(2), 2862–2869.

Varda, K. (n.d.). *Protocol Buffers*. http://code.google.com/apis/protocolbuffers/.

Venkatesh, A., Khatri, C., Ram, A., Guo, F., Gabriel, R., Nagar, A., ... others (2018). On evaluating and comparing conversational agents. In *Conference on neural information processing systems (NeurIPS)*.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... others (2020). Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 1–12.

Vogel, A., & Jurafsky, D. (2010). Learning to follow navigational directions. In *Proceedings of the 48th annual meeting of the association for computational linguistics*.

Vogel, J., & Murphy, K. (2007). A non-myopic approach to visual search. In *Fourth canadian conference on computer and robot vision (crv'07)* (pp. 227–234).

Von Neumann, J., & Morgenstern, O. (1947). Theory of games and economic behavior, 2nd rev.

Wandzel, A., Oh, Y., Fishman, M., Kumar, N., Lawson L.S., W., & Tellex, S. (2019). Multi-Object Search using Object-Oriented POMDPs. In *International conference on robotics and automation (icra)*. doi: 10.1109/ICRA.2019.8793888

Wandzel, A., Oh, Y., Fishman, M., Kumar, N., & Tellex, S. (2019). Multi-Object Search using Object-Oriented POMDPs. In *2019 international conference on robotics and automation (icra)*.

Wang, C., Cheng, J., Wang, J., Li, X., & Meng, M. Q.-H. (2018). Efficient object search with belief road map using mobile robot. *IEEE Robotics and Automation Letters*, *3*(4), 3081–3088.

Wang, X., Huang, Q., Celikyilmaz, A., Gao, J., Shen, D., Wang, Y.-F., ... Zhang, L. (2019). Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 6629–6638).

Wang, Y., Giuliari, F., Berra, R., Castellini, A., Del Bue, A., Farinelli, A., ... Setti, F. (2020). Pomp: Pomcp-based online motion planning for active visual search in indoor environments. *arXiv preprint arXiv:2009.08140*.

Washburn, A. R. (1980). On a search for a moving target. *Naval Research Logistics Quarterly*, *27*(2), 315–322.

Wei, W., Le, Q., Dai, A., & Li, J. (2018). Airdialogue: An environment for goal-oriented dialogue research. In *Proceedings of the 2018 conference on empirical methods in natural language processing* (pp. 3844–3854).

Wen, T.-H., Vandyke, D., Mrkšić, N., Gašić, M., Rojas-Barahona, L. M., Su, P.-H., ... Young, S. (2017, April). A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th conference of the European chapter of the association for computational linguistics: Volume 1, long papers* (pp. 438–449). Valencia, Spain: Association for Computational Linguistics. Retrieved from https://aclanthology.org/E17-1042

Weng, X., Wang, J., Held, D., & Kitani, K. (2020). 3d multi-object tracking: A baseline and new evaluation metrics. In *2020 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 10359–10366).

Whitney, D., Rosen, E., MacGlashan, J., Wong, L. L., & Tellex, S. (2017). Reducing errors in object-fetching interactions through social feedback. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1006–1013).

Williams, L. G. (1967). The effects of target specification on objects fixated during visual search. *Acta psychological*, *27*, 355–360.

Wise, M., Ferguson, M., King, D., Diehr, E., & Dymesich, D. (2016). Fetch and freight: Standard platforms for service robot applications. In *Workshop on autonomous mobile service robots.*

Wixson, L. E., & Ballard, D. H. (1994). Using intermediate objects to improve the efficiency of visual search. *International Journal of Computer Vision*, *12*(2-3), 209–230.

Wong, L. L., Kaelbling, L. P., & Lozano-Pérez, T. (2013). Manipulation-based active search for occluded objects. In *2013 ieee international conference on robotics and automation* (pp. 2814–2819).

Wortsman, M., Ehsani, K., Rastegari, M., Farhadi, A., & Mottaghi, R. (2019). Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 6750–6759).

Xia, F., Zamir, A. R., He, Z., Sax, A., Malik, J., & Savarese, S. (2018). Gibson env: Real-world perception for embodied agents. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 9068–9079).

Xiang, Y., Schmidt, T., Narayanan, V., & Fox, D. (2017). Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*.

Xiao, Y., Katt, S., ten Pas, A., Chen, S., & Amato, C. (2019). Online planning for target object search in clutter under partial observability. In *Proceedings of the international conference on robotics and automation.*

Xu, N., Huo, C., Zhang, X., Cao, Y., Meng, G., & Pan, C. (2021). Dynamic camera configuration learning for high-confidence active object detection. *Neurocomputing*, *466*, 113–127.

Xue, H., Liu, C., Wan, F., Jiao, J., Ji, X., & Ye, Q. (2019). Danet: Divergent activation for weakly supervised object localization. In *Proceedings of the ieee/cvf international conference on computer vision* (pp. 6589–6598).

Yang, W., Wang, X., Farhadi, A., Gupta, A., & Mottaghi, R. (2019). Visual semantic navigation using scene priors. *International Conference on Learning Representations (ICLR)*.

Ye, N., Somani, A., Hsu, D., & Lee, W. S. (2017). Despot: Online POMDP planning with regularization. *Journal of Artificial Intelligence Research*, *58*, 231–266.

Ye, X., Lin, Z., Li, H., Zheng, S., & Yang, Y. (2018). Active object perceiver: Recognition-guided policy learning for object searching on mobile robots. In *2018 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 6857–6863).

Ye, X., & Yang, Y. (2021, June). Hierarchical and partially observable goal-driven policy learning with goals relational graph. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition.*

Ye, Y., & Tsotsos, J. K. (1997). Sensor planning for 3D object search,. *Computer Vision and Image Understanding*, *73*, 145-168.

Yilmaz, A., Javed, O., & Shah, M. (2006). Object tracking: A survey. *Acm computing surveys (CSUR)*, *38*(4), 13–es.

Zeng, Z., Röfer, A., & Jenkins, O. C. (2020). Semantic linking maps for active visual object search. In (pp. 1984–1990).

Zengin, U., & Dogan, A. (2011). Cooperative target pursuit by multiple uavs in an adversarial environment. *Robotics and Autonomous Systems*, *59*(12), 1049–1059.

Zhang, D., Han, J., Cheng, G., & Yang, M.-H. (2021). Weakly supervised object localization and detection: A survey. *IEEE transactions on pattern analysis and machine intelligence*, *44*(9), 5866–5885.

Zhang, G., & Garg, D. P. (2008). Mobile multi-robot control in target search and retrieval. In *Dynamic systems and control conference* (Vol. 43352, pp. 677–683).

Zhang, Y., Tian, G., Lu, J., Zhang, M., & Zhang, S. (2019). Efficient dynamic object search in home environment by mobile robot: A priori knowledge-based approach. *IEEE Transactions on Vehicular Technology*, *68*(10), 9466–9477.

Zhang, Y., Tian, G., Shao, X., Liu, S., Zhang, M., & Duan, P. (2021). Building metric-topological map to efficient object search for mobile robot. *IEEE Transactions on Industrial Electronics*, *69*(7), 7076–7087.

Zhao, W., & Chen, W. (2021). Efficient planning for object search task based on hierarchical pomdp. In *2021 ieee international conference on robotics and biomimetics (robio)* (pp. 1593–1598).

Zheng, K. (2021). ROS Navigation Tuning Guide. In A. Koubaa (Ed.), *Robot operating system (ros)* (p. 197-226). Springer International Publishing.

Zheng, K., Bayazit, D., Mathew, R., Pavlick, E., & Tellex, S. (2021b). Spatial language understanding for object search in partially observed cityscale environments. In *International Conference on Robot and Human Interactive Communication (RO-MAN)*.

Zheng, K., Chitnis, R., Sung, Y., Konidaris, G., & Tellex, S. (2022). Towards optimal correlational object search. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Zheng, K., Paul, A., & Tellex, S. (2023). A system for generalized 3D multi-object search. In *IEEE International Conference on Robotics and Automation (ICRA)*. (accepted)

Zheng, K., & Pronobis, A. (2019). From pixels to buildings: End-to-end probabilistic deep networks for large-scale semantic mapping. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Zheng, K., Sung, Y., Konidaris, G., & Tellex, S. (2021a). Multi-resolution POMDP planning for multi-object search in 3D. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. (IROS RoboCup Best Paper Award)

Zheng, K., & Tellex, S. (2020). pomdp_py: A framework to build and solve POMDP problems. In *ICAPS 2020 workshop on planning and robotics (PlanRob)*.

Zhu, M., Zhao, B., & Kong, T. (2022). Navigating to objects in unseen environments by distance prediction. *arXiv preprint arXiv:2202.03735*.

Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., & Farhadi, A. (2017). Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 ieee international conference on robotics and automation (icra)* (pp. 3357–3364).

Zilberstein, S. (1996). Using anytime algorithms in intelligent systems. *AI magazine*, *17*(3), 73–73.

Zou, Z., Shi, Z., Guo, Y., & Ye, J. (2019). Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*.

REVISION NOTICE

The present document reflects the following revisions:

- January 13th, 2023: submitted in full to the Brown University Graduate School.
- January 20th, 2023: revised the chapter on GenMOS (Chapter 5) following paper acceptance to ICRA 2023. In particular, added Figure 5.1, and clarified contributions in Section 5.1. Added definitions of default value and initial value of an octree belief node in Definition 1 and 2 in Chapter 4, then referenced in Chapter 5 for clarity.
- January 22nd, 2023: completed the acknowledgements. Fixed writing bugs in the `pomdp_py` appendix and added a figure based on (Zheng & Tellex, 2020).
- March 4th, 2023: updated Chapter 5 on GenMOS with additional simulation results (Greedy and Random baselines) and discussion from the camera-ready.
- May 3rd, 2023: fixed a typo of index k (should be 0, but was 1) in Equation 2.12 in Appendix 2.2.5 on the derivation of POMDP Bellman equation.
- June 25th, 2023: fixed several minor typos in Equation 2.4 in Section 2.2 regarding the t suffix in a.