

# Multi-Resolution POMDP Planning for Multi-Object Search in 3D

Kaiyu Zheng<sup>†</sup>, Yoonchang Sung\*, George Konidaris<sup>†</sup>, Stefanie Tellex<sup>†</sup>

**Abstract**— Robots operating in households must find objects on shelves, under tables, and in cupboards. In such environments, it is crucial to search efficiently at 3D scale while coping with limited field of view and the complexity of searching for multiple objects. Principled approaches to object search frequently use Partially Observable Markov Decision Process (POMDP) as the underlying framework for computing search strategies, but constrain the search space in 2D. In this paper, we present a POMDP formulation for multi-object search in a 3D region with a frustum-shaped field-of-view. To efficiently solve this POMDP, we propose a multi-resolution planning algorithm based on online Monte-Carlo tree search. In this approach, we design a novel octree-based belief representation to capture uncertainty of the target objects at different resolution levels, then derive abstract POMDPs at lower resolutions with dramatically smaller state and observation spaces. Evaluation in a simulated 3D domain shows that our approach finds objects more efficiently and successfully compared to a set of baselines without resolution hierarchy in larger instances under the same computational requirement. We demonstrate our approach on a mobile robot to find objects placed at different heights in two  $10m^2 \times 2m$  regions by moving its base and actuating its torso.

## I. INTRODUCTION

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 [1, 2], 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 search region under limited field of view (FOV), where target objects could be partially or completely occluded. In this work, we consider the problem setting where a robot must search for multiple objects in a search region by actively moving its camera, with as few steps as possible (Figure 1).

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 works have used Partially Observable Markov Decision Process (POMDP) as a principled decision-theoretic framework for object search [3, 4, 5]. However, to ensure the POMDP is manageable to solve, previous works reduce the search space or robot mobility to 2D [2, 6, 7], although objects exist in rich 3D environments. The key challenges lie in the intractability of maintaining exact belief due to large state space [8], and the high branching factor for planning due to large observation space [9, 10].

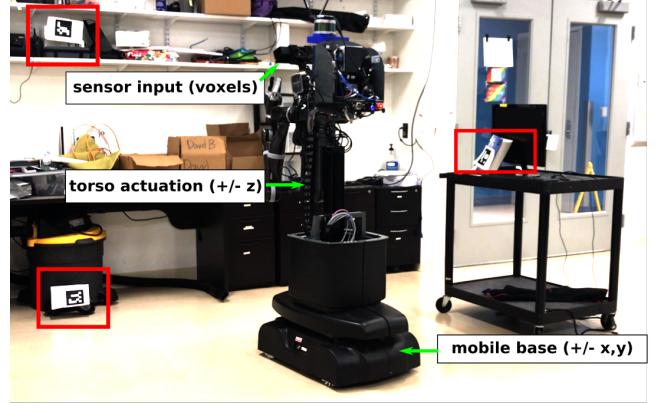


Fig. 1: An example of the 3D-MOS problem where a torso-actuated mobile robot is tasked to search for three objects placed at different heights in a lab environment. The objects are represented by paper AR tags marked by red boxes. Note that the robot must actively move itself due to limited field of view, and the objects can be occluded by the attached obstacles if viewed from the side.

In this paper, 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. [6], 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 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 octree-based belief representation 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 [11, 12]. Finally, a Monte-Carlo Tree Search (MCTS) based online planning algorithm, called Partially-Observable Upper Confidence bounds for Trees (POUCT) [8], 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 approach in a simulated, discretized 3D domain where a robot with a 6 degrees-of-freedom camera searches for objects of different shapes and sizes randomly generated and placed in a grid environment.

<sup>†</sup>Brown University, Providence, RI. \*MIT CSAIL, Cambridge, MA.  
Email: {kzheng10, gdk, stefie10}@cs.brown.edu, yooncs8@csail.mit.edu

The results show that, as the problem scales, our approach outperforms exhaustive search as well as POMDP baselines without resolution hierarchy under the same computational requirement. We also show that our method is more robust to sensor uncertainty against the POMDP baselines. Finally, we demonstrate our approach on a torso-actuated mobile robot in a lab environment (Figure 6). The robot finds 3 out of 6 objects placed at different heights in two  $10\text{m}^2 \times 2\text{m}$  regions in around 15 minutes.

## II. BACKGROUND

POMDPs compactly represent the robot's uncertainty in target locations and its own sensor [13], and OO-POMDPs factor the domain in terms of objects, which fits the object search problem naturally [6]. Below, we first provide a brief overview of POMDPs and OO-POMDPs. Then, we discuss related work in object search.

### A. POMDPs and OO-POMDPs

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  $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, R, \gamma \rangle$ , where  $\mathcal{S}, \mathcal{A}, \mathcal{O}$  denote the state, action and observation spaces, and the functions  $T(s, a, s') = \Pr(s'|s, a)$ ,  $O(s', a, o) = \Pr(o|s', a)$ , 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 environment state to transition from  $s \in \mathcal{S}$  to  $s' \in \mathcal{S}$ . The environment in turn returns the agent an observation  $o \in \mathcal{O}$  and reward  $r \in \mathbb{R}$ . A *history*  $h_t = (ao)_{1:t-1}$  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  $b_{t+1}(s') = \eta \Pr(o|s', a) \sum_s \Pr(s'|s, a) b_t(s)$  where  $\eta = \sum_s \sum_{s'} \Pr(o|s', a) \Pr(s'|s, a) b_t(s)$  is the normalizing constant. The task of the agent is to find a policy  $\pi(b_t) \in \mathcal{A}$  which maximizes the expectation of future discounted rewards  $V^\pi(b_t) = \mathbb{E} [\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, \pi(b_{t+k})) | b_t]$  with a discount factor  $\gamma$ .

An Object-Oriented POMDP (OO-POMDP) [6] (generalization of OO-MDP [14]) is a POMDP that considers the state and observation spaces to be factored by a set of  $n$  objects where each belongs to a class with a set of attributes. A simplifying assumption is made for the 2D MOS domain 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.

Offline POMDP solvers are often too slow to be practical for large domains [15]. State-of-the-art online POMDP solvers leverage sparse belief sampling and MCTS to scale up to domains with large state spaces and to address the curse of history [8, 16, 9]. POMCP [8] is one such algorithm which combines particle belief representation with Partially Observable UCT (POUCT), which extends the UCT algorithm [17] to POMDPs and is proved to be asymptotically optimal [8]. We build upon POUCH due to its optimality and simplicity of implementation.

### B. Related Work

Previous work primarily address the computational complexity of object search by hypothesizing likely regions based on object co-occurrence [1, 18], semantic knowledge [2] or language [6], reducing the state space from 3D to 2D [6, 19, 20, 21], or constrain the sensor to be stationary [5, 22]. Our work 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 [3, 21, 23]. 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. [2] first infer a room to search in then perform search by calculating candidate viewpoints in a 2D plane. Li et al. [7] plan sensor movements online, yet assume objects are placed at the same surface level in a container with partial occlusion. Xiao et al. [3] 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. [6] formulates 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.

## III. MULTI-OBJECT SEARCH IN 3D

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  $g \in G \subseteq \mathbb{R}^3$  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:

**State space  $\mathcal{S}$ .** An environment state  $s = \{s_1, \dots, s_n, s_r\}$  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  $\mathcal{S}_i^l = G^l$ .

**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

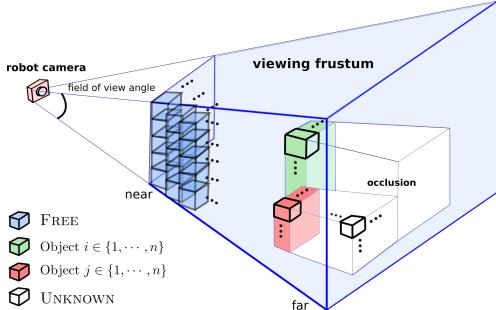


Fig. 2: Illustration of the viewing frustum and volumetric observation. The viewing frustum  $V$  consists of  $|V|$  voxels, where each  $v \in V$  can be labeled as  $i \in \{1, \dots, n\}$ , FREE or UNKNOWN.

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 paper 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  $i \in \{1, \dots, n\}$ , FREE, or UNKNOWN (Figure 2). 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  $V_1, \dots, V_n$  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.<sup>1</sup> Hence, the observation  $o = \bigcup_{i=1}^n o_i$  where  $o_i = \{(v, d(v)) | v \in V_i\}$ .

**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:  $\text{MOVE}(s_r, g)$  action moves the robot from pose in  $s_r$  to destination  $g \in G$  stochastically.  $\text{LOOK}(\theta)$  changes the camera pose to look in the direction specified by  $\theta \in \mathbb{R}^3$ , and projects a viewing frustum  $V$ .  $\text{FIND}(i, g)$  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.

**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  $\text{MOVE}(s_r, g)$  and  $\text{LOOK}(\theta)$  change the camera location and direction to  $g$  and  $\theta$  following a domain-specific

<sup>1</sup>The FOV  $V$  is fixed for a given camera pose in the robot state, therefore excluding UNKNOWN voxels does not lose information.

stochastic dynamics function. The action  $\text{FIND}(i, g)$  adds  $i$  to the set of *found objects* in the robot state only if  $g$  is within the FOV determined by  $s_r$ .

**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  $R_{\max} \gg 0$  if an object is correctly identified by a FIND action, otherwise the FIND action incurs a  $R_{\min} \ll 0$  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.

#### A. Observation Model

We have previously described how a volumetric observation  $o$  can be factored by objects into  $o_1, \dots, o_n$ . 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 [24]. 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 V) 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  $\alpha$  and  $\beta$  do not necessarily sum to one because the belief update equation does not require the observation model to be normalized, as explained in Section II-A. Thus, hyperparameters  $\alpha$  and  $\beta$  independently control the reliability of the observation model.

#### IV. OCTREE BELIEF REPRESENTATION

Particle belief representation [8, 16] 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  $b_t^i(s_i)$  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. III-A).

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

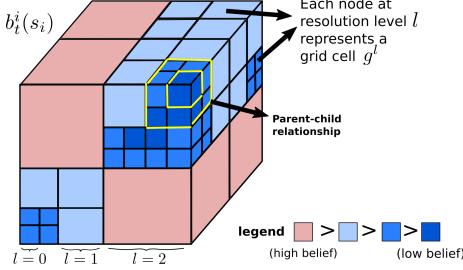


Fig. 3: Illustration the octree belief representation  $b_t^i(s_i)$ . The color on a node  $g^l$  indicates the belief  $\text{VAL}_t^i(g^l)$  that the object is located within  $g^l$ . The highlighted grid cells indicate parent-child relationship between a grid cell at resolution level  $l = 1$  (parent) and one at level  $l = 0$ .

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 3). Each node stores a value  $\text{VAL}_t^i(g^l) \in \mathbb{R}$ , 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  $\text{CH}^k(g^l)$ . By definition,  $b_t^i(g^l) = \Pr(g^l|h_t) = \sum_{c \in \text{CH}^k(g^l)} \Pr(c|h_t)$ . Thus, with a normalizer  $\text{NORM}_t = \sum_{g \in G} \text{VAL}_t^i(g)$ , we can rewrite the normalized belief as:

$$b_t^i(g^l) = \frac{\text{VAL}_t^i(g^l)}{\text{NORM}_t} = \sum_{c \in \text{CH}^k(g^l)} \left( \frac{\text{VAL}_t^i(c)}{\text{NORM}_t} \right), \quad (1)$$

which means  $\text{VAL}_t^i(g^l) = \sum_{c \in \text{CH}^k(g^l)} \text{VAL}_t^i(c)$ . 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  $\text{VAL}_0^i(g) = 1$  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  $\text{VAL}_0^i(g^l) = \sum_{c \in \text{CH}^1(g^l)} \text{VAL}_0^i(c) = |\text{CH}^1(g^l)|$ .

#### A. Belief Update

We have defined a per-voxel observation model for  $\Pr(o_i|s', a)$ , which is reduced 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 1. This algorithm updates the value of the ground-level node  $g$  corresponding to each voxel  $v \in V_i$  as  $\text{VAL}_{t+1}^i(g) = \Pr(d(v)|s', a)\text{VAL}_t^i(g)$ . The normalizer is updated to make sure  $b_{t+1}^i$  is normalized

*Lemma 1:* The normalizer  $\text{NORM}_t$  at time  $t$  can be correctly updated by adding the incremental update of values as in Algorithm 1.

*Proof:* The normalizer must be equal to the sum of node values at the ground level for the next belief  $b_{t+1}^i$  to be valid (Equation 1). That is,  $\text{NORM}_{t+1} = \sum_{s_i \in G} \text{VAL}_{t+1}^i(s_i)$ . 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

---

**Algorithm 1:** 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  $\Psi(b_t^i)$  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  $s_i \notin \Psi(b_t^i)$  then
        | Insert node at  $s_i$  to  $\Psi(b_t^i)$ ;
    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

```

---

$s_i \notin V_i$ , the *unnormalized* observation model is uniform, thus  $\text{VAL}_{t+1}^i(s_i) = \Pr(d(s_i)|s', a)\text{VAL}_t^i(s_i) = \text{VAL}_t^i(s_i)$ . Therefore,  $\text{NORM}_{t+1} = \sum_{s_i \in V_i} \text{VAL}_{t+1}^i(s_i) + \sum_{s_i \notin V_i} \text{VAL}_t^i(s_i)$ . 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  $\text{NORM}_t$ , we obtain  $\text{NORM}_{t+1} = \text{NORM}_t + \sum_{s_i \in V_i} (\text{VAL}_{t+1}^i(s_i) - \text{VAL}_t^i(s_i))$  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.

#### B. 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. Given resolution level  $l$ , we sample from  $S_i^l$  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 a state is sampled. If  $s_i^{l_{des}}$  is sampled, then all nodes in the octree that cover  $s_i^{l_{des}}$ , i.e.,  $s_i^{l_{max}}, \dots, s_i^{l_{des}+2}, s_i^{l_{des}+1}$ , 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, the task of sampling  $s_i^{l_{des}}$  is translated into sampling a sequence of samples  $s_i^{l_{max}}, \dots, s_i^{l_{des}+2}, s_i^{l_{des}+1}, s_i^{l_{des}}$ , each according to the distribution  $\Pr(s_i^l|s_i^{l+1}, h_t) = \frac{\text{VAL}_t^i(s_i^l)}{\text{VAL}_t^i(s_i^{l+1})}$ . 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  $s_i^{l_{des}}$  exactly according to  $b_t^i(s_i^{l_{des}})$  with time complexity  $O(\log(|G|))$ .

#### V. MULTI-RESOLUTION PLANNING VIA ABSTRACTIONS

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 POUCHT 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.

### A. Abstract 3D-MOS

We adopt the abstraction scheme in Li et al. [11] 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  $\phi_i : \mathcal{S}_i \rightarrow \mathcal{S}_i^l$  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  $\phi_i$  [11].

**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  $\text{MOVEOP}(s_r, g)$  is an *option* [25] 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  $\text{FIND}(i, g)$  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  $\text{FIND}(i, g)$  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 [12],  $f_i$  transitions to  $f'_i$  following

$$\Pr(f'_i | f_i, s_i^l, h_t, \text{FIND}(i, g)) \quad (2)$$

$$= \sum_{s_i \in \phi_i^{-1}(s_i^l)} \Pr(f'_i | s_i, f_i, \text{FIND}(i, g)) \Pr(s_i | s_i^l, h_t). \quad (3)$$

The above is consistent with the abstract transition function in the works [11, 12] 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  $i$ :

$$\Pr(o_i^l | \hat{s}', a, h_t) = \Pr(d(s_i^l) | \hat{s}', a, h_t) \quad (4)$$

$$\approx \Pr(d(s_i^l) | s_i^l, s_r, a, h_t) \quad (5)$$

$$= \sum_{s_i \in \phi_i^{-1}(s_i^l)} \Pr(d(s_i^l) | s_i, s_r, a) \Pr(s_i | s_i^l, h_t). \quad (6)$$

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

---

### Algorithm 2: 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  $P_l = \langle \hat{\mathcal{S}}, \hat{\mathcal{A}}, \hat{\mathcal{O}}, \hat{T}, \hat{O}, R, \gamma, l \rangle$ 
     $\mathcal{G} \leftarrow \text{GenerativeFunction}(P_l);$ 
     $Q_P(b_t, \hat{a}) \leftarrow \text{POUCT}(\mathcal{G}, h_t, d);$ 
  end
   $\hat{a} \leftarrow \text{argmax}_{\hat{a}} \{Q_P(b_t, \hat{a}) | P \in \mathcal{P}\};$ 
  return  $\hat{a}$ 

```

---

$\Pr(s_i | s_i^l, h_t)$  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 [26]: We sample  $k$  ground states from  $\phi_i^{-1}(s_i^l)$  according to their weights.<sup>2</sup> Then we set  $d(s_i^l) = i$  if the majority of these samples have  $d(s_i) = i$ , and  $d(s_i^l) = \text{FREE}$  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.

### B. 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.

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 2. 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 [8] for details on POUCT). We call this algorithm Multi-Resolution POUCT (MR-POUCT).

<sup>2</sup>We tested  $k = 10$  and  $k = 40$  and observed similar search performance. We used  $k = 10$  in our experiments.

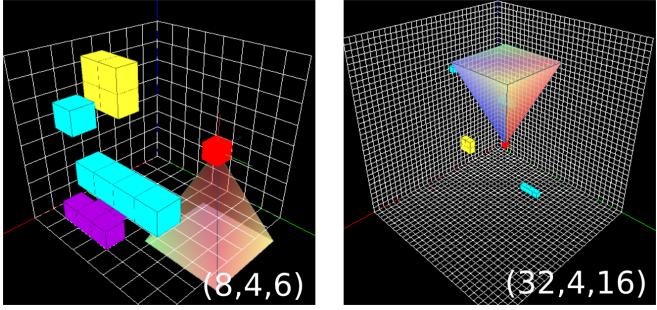


Fig. 4: Simulated environment for 3D object search. The robot (represented as a red cube) can project a viewing frustum to observe the search space, where objects are represented by sets of cubes. Search space size scales from  $4^3$  to  $32^3$ . The tuple  $(m, n, d)$  at lower-right defines the problem instance.

## VI. EXPERIMENTS

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 and a study on a real robot.

### A. Simulation Evaluation

**Setup.** We implement our approach in a simulated environment designed to reflect the essence of the 3D-MOS domain (Figure 4). 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  $\gamma$  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.<sup>3</sup> 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 5. Particle deprivation happens quickly due to large observation space, and the behavior degenerates to a random agent, causing POMCP to 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.

We then investigate the performance of our method with respect to changes in sensing uncertainty, controlled by the parameters  $\alpha$  and  $\beta$  of the observation model. According to the belief update algorithm in Section IV-A, a noisy but functional sensor should increase the belief  $VAL_t^i(g)$  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

<sup>3</sup>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.

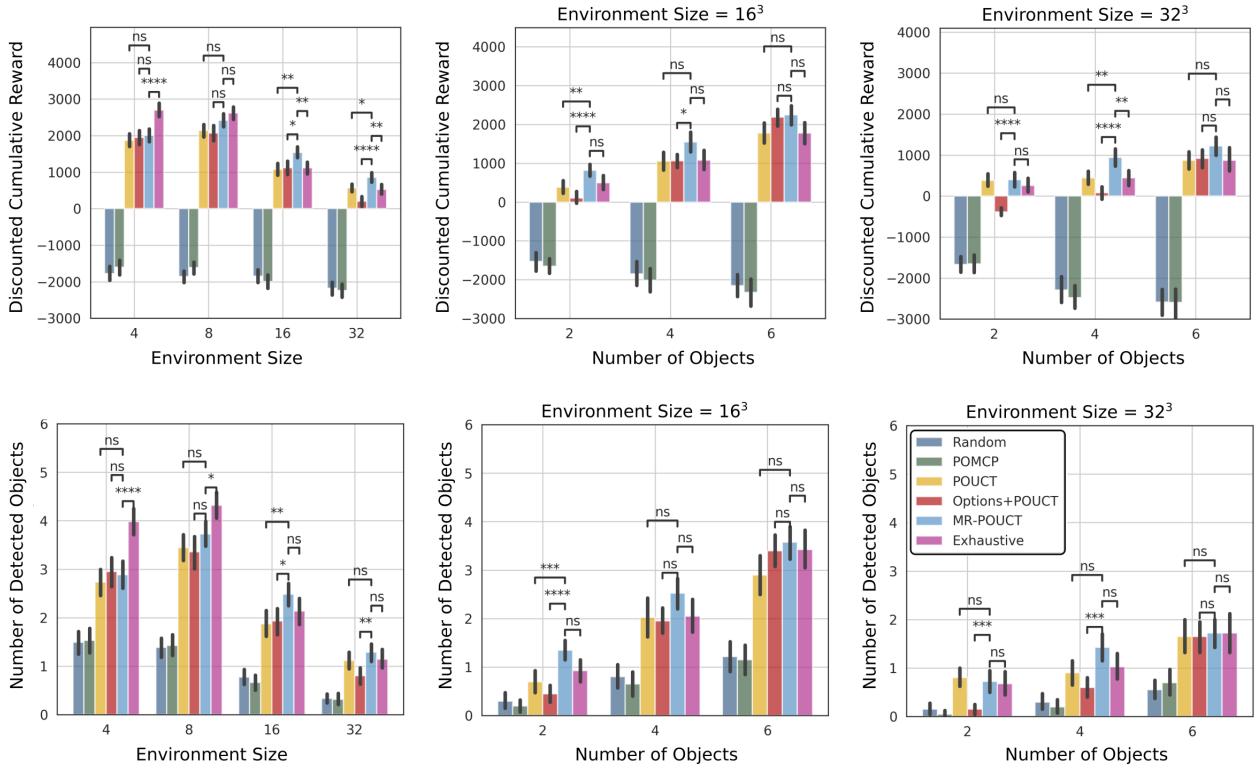


Fig. 5: Discounted cumulative reward and number of detected objects as the environment size ( $m$ ) increases and as the of number of objects ( $n$ ) increases. Exhaustive search performs well in small-scale environments (4 and 8) where exploration strategy is not taken advantage of. In large environments, 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 ( $p > 0.05$ ), \* ( $p \leq 0.05$ ), \*\* ( $p \leq 0.01$ ), \*\*\* ( $p \leq 0.001$ ), \*\*\*\* ( $p \leq 0.0001$ ).

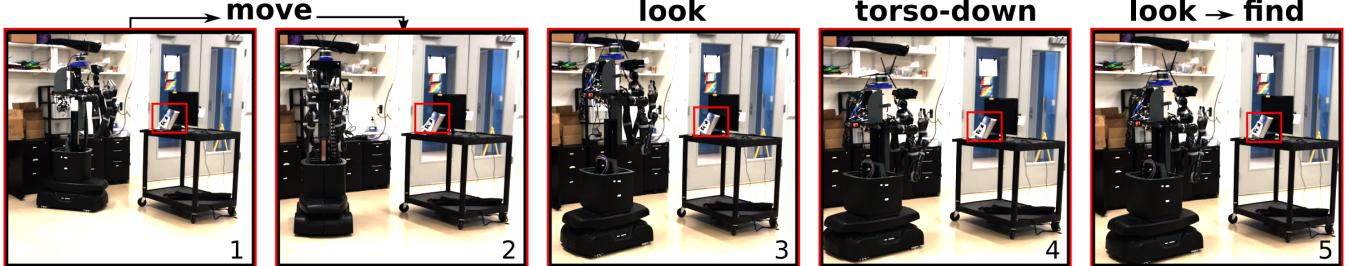


Fig. 6: Example action sequence produced by the proposed approach. The mobile robot first navigates in front of a portable table (1-2). It then takes a LOOK action to observe the space in front (3), and no target is observed since the torso is too high. The robot then decides to lower its torso (4), 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.

of (16, 2, 10) is used to conduct this experiment. Results in Figure 7 show that MR-POUCT is consistently better in all parameter settings. We observe that  $\beta$  has almost no impact to any algorithm’s performance as long as  $\beta < 1$ , whereas decreasing  $\alpha$  changes the agent behavior such that it must decide to LOOK multiple times before being certain.

#### B. Demonstration on a Torso-Actuated Mobile 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  $10\text{m}^2 \times 2\text{m}$  (Figure. 6), 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 neighbors of a

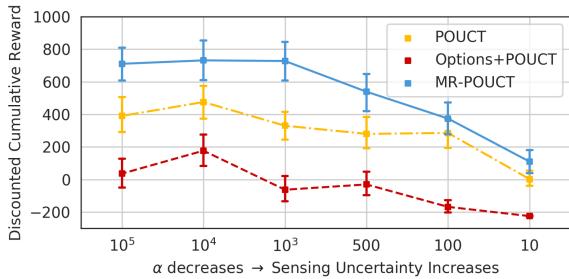


Fig. 7: Discounted cumulative reward with 95% confidence interval as the sensing uncertainty increases, aggregating over the  $\beta$  settings.

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 6) shows that the robot decides to lower its torso in order to LOOK and FIND an object.<sup>4</sup> A failure mode is that the object may not be covered by any viewpoint and thus not detected; this can be improved with a denser topological map, or by considering destinations of MOVE actions sampled from the continuous search region.

## VII. CONCLUSION

We present a POMDP formulation of multi-object search in 3D with volumetric observation space and solve it with a novel multi-resolution planning algorithm. Our evaluation 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.

One limitation of the presented work is that the assumption of object independence, though beneficial computationally, may discard useful object dependence information in some cases. Optimal search for correlated objects becomes important. In addition, we do not explicitly reason over object geometry in the observation model. Considering belief over geometric appearances is a challenging future direction. Finally, incorporating a heuristic rollout policy may be a promising direction for more realistic object search problems while sacrificing optimality.

## ACKNOWLEDGEMENTS

The authors thank Selena Ling for help with the simulator. This work was supported by the National Science Foundation under grant number IIS1652561, ONR under grant number N00014-17-1-2699, the US Army under grant number W911NF1920145, Echo Labs, STRAC Institute, and Hyundai.

## REFERENCES

- [1] T. Kollar and N. Roy, “Utilizing object-object and object-scene context when planning to find things,” in *IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 2168–2173.
- [2] A. Aydemir, A. Pronobis, M. Göbelbecker, and P. Jensfelt, “Active visual object search in unknown environments using uncertain semantics.” *IEEE Transactions on Robotics (T-RO)*, vol. 29, no. 4, pp. 986–1002, Aug. 2013.
- [3] Y. Xiao, S. Katt, A. ten Pas, S. Chen, and C. Amato, “Online planning for target object search in clutter under partial observability,” in *Proceedings of the International Conference on Robotics and Automation*, 2019.
- [4] N. Atanasov, B. Sankaran, J. L. Ny, G. Pappas, and K. Daniilidis, “Nonmyopic view planning for active object classification and pose estimation,” *IEEE Trans. on Robotics (TRO)*, 2014.
- [5] M. Danielczuk, A. Kurenkov, A. Balakrishna, M. Matl, D. Wang, R. Martín-Martín, A. Garg, S. Savarese, and K. Goldberg, “Mechanical search: Multi-step retrieval of a target object occluded by clutter,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 1614–1621.
- [6] A. Wandzel, Y. Oh, M. Fishman, N. Kumar, and S. Tellex, “Multi-Object Search using Object-Oriented POMDPs,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.
- [7] J. K. Li, D. Hsu, and W. S. Lee, “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)*. IEEE, 2016.
- [8] D. Silver and J. Veness, “Monte-Carlo planning in large POMDPs,” in *Advances in neural information processing systems*, 2010.
- [9] Z. N. Sunberg and M. J. Kochenderfer, “Online algorithms for POMDPs with continuous state, action, and observation spaces,” in *Twenty-Eighth International Conference on Automated Planning and Scheduling*, 2018.
- [10] N. P. Garg, D. Hsu, and W. S. Lee, “DESPOT- $\alpha$ : Online POMDP planning with large state and observation spaces,” in *Robotics: Science and Systems*, 2019.
- [11] L. Li, T. J. Walsh, and M. L. Littman, “Towards a unified theory of state abstraction for MDPs,” in *ISAIM*, 2006.
- [12] A. Bai, S. Srivastava, and S. J. Russell, “Markovian state and action abstractions for MDPs via hierarchical MCTS,” in *IJCAI*, 2016.
- [13] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.
- [14] C. Diuk, A. Cohen, and M. L. Littman, “An object-oriented representation for efficient reinforcement learning,” in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 240–247.
- [15] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, “Online planning algorithms for POMDPs,” *Journal of Artificial Intelligence Research*, vol. 32, pp. 663–704, 2008.
- [16] A. Soman, N. Ye, D. Hsu, and W. S. Lee, “DESPOT: Online POMDP planning with regularization,” in *Advances in neural information processing systems*, 2013, pp. 1772–1780.
- [17] L. Kocsis and C. Szepesvári, “Bandit based Monte-Carlo planning,” in *European conference on machine learning*. Springer, 2006.
- [18] L. E. Wixson and D. H. Ballard, “Using intermediate objects to improve the efficiency of visual search,” *International Journal of Computer Vision*, vol. 12, no. 2-3, pp. 209–230, 1994.
- [19] C. Wang, J. Cheng, J. Wang, X. Li, and M. Q.-H. Meng, “Efficient object search with belief road map using mobile robot,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3081–3088, 2018.
- [20] A. Sarmiento, R. Murrieta, and S. A. Hutchinson, “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)*, 2003.
- [21] X. Nie, L. L. Wong, and L. P. Kaelbling, “Searching for physical objects in partially known environments,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016.
- [22] M. R. Dogar, M. C. Koval, A. Tallavajhula, and S. S. Srinivasa, “Object search by manipulation,” *Autonomous Robots*, 2014.
- [23] L. L. Wong, L. P. Kaelbling, and T. Lozano-Pérez, “Manipulation-based active search for occluded objects,” in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2814–2819.
- [24] J. J. Park, P. Florene, J. Straub, R. Newcombe, and S. Lovegrove, “Deepsdf: Learning continuous signed distance functions for shape representation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 165–174.
- [25] R. S. Sutton, D. Precup, and S. Singh, “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning,” *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.
- [26] A. Shapiro, “Monte Carlo sampling methods,” *Handbooks in operations research and management science*, vol. 10, pp. 353–425, 2003.

<sup>4</sup>Video footage with visualization of volumetric observations and octree belief update is available at <https://zkytony.github.io/3D-MOS/>.