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

Kaiyu Zheng*, Yoonchang Sung†, George Konidaris*, Stefanie Tellex*
*Brown University, Providence, RI
†MIT CSAIL, Cambridge, MA
{kzheng10, gdk, stefie10}@cs.brown.edu, yooncs8@csail.mit.edu

***Abstract*—Robots operating in household environments must find objects on shelves, under tables, and in cupboards. Previous work often formulate the object search problem as a POMDP (Partially Observable Markov Decision Process), yet constrain the search space in 2D. We propose a new approach that enables the robot to efficiently search for objects in 3D, taking occlusions into account. We model the problem as an object-oriented POMDP, where the robot receives a volumetric observation from a viewing frustum and must produce a policy to efficiently search for objects. To address the challenge of large state and observation spaces, we first propose a per-voxel observation model which drastically reduces the observation size necessary for planning. Then, we present a novel octree-based belief representation which captures beliefs at different resolutions and supports efficient exact belief update. Finally, we design an online multi-resolution planning algorithm that leverages the resolution layers in the octree structure as levels of abstractions to the original POMDP problem. Our evaluation in a simulated 3D domain shows that, as the problem scales, our approach significantly outperforms baselines without resolution hierarchy by 25%–35% in cumulative reward. We demonstrate the practicality of our approach on a torso-actuated mobile robot searching for objects in areas of a cluttered lab environment where objects appear on surfaces at different heights.**

I. Introduction

Robots operating in human spaces, such as homes for the elderly, must find objects such as glasses, mobile phones, or cleaning supplies that could be on the floor, shelves, or tables. This search space is naturally 3D [1]. For humans, finding objects is a frequent task that involves hypothesizing search regions (e.g. kitchen or lounge corner) based on semantic knowledge or past experience [2, 3], but ultimately depends on careful search by moving and looking within a search region [4, 5]. Analogously for robots, finding objects requires the ability to produce an efficient search policy under limited *Field-Of-View* (FOV) within a designated search region, where target objects could be partially or completely occluded.

Searching for a single, static object in 3D is an NP-complete problem [1]. Exhaustive search strategies are hence incapable of handling large search spaces. The problem becomes even more difficult when the robot must find multiple objects, since the space of possible object locations grows exponentially as the number of objects increases [4]. Sensor uncertainty further complicates this problem [6]. The *Partially Observable Markov Devision Process* (POMDP) [7] has been widely adopted as a framework to describe and solve the object search problem [3, 4, 5, 6, 8, 9]. Nevertheless, to ensure the POMDP is manageable to solve, previous works often reduce the search space or robot mobility to 2D.

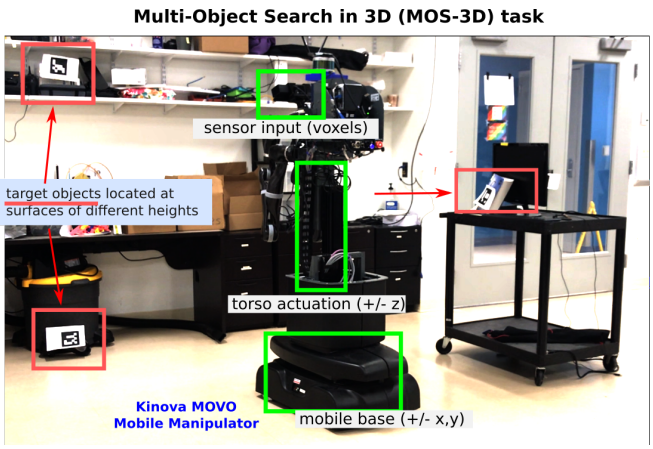

Fig. 1: An example of the MOS-3D problem where a torso-actuated mobile robot is tasked to search for 3 objects in a cluttered lab environment.

In this work, we consider the Multi-Object Search in 3D (MOS-3D) task as a POMDP with 3D state and action spaces. Moreover, we consider a volumetric observation space where the robot receives voxels and their labels within the viewing frustum projected by a mounted camera. This is in contrast with previous works that often consider an observation space of the target object poses, which omit information necessary for planning. Indeed, solving POMDPs at this scale is computationally daunting [10]. The challenge of solving POMDPs lies in the intractability of the exact belief update due to the large state space, and high branching factor for planning due to large observation space.

We pose the problem as an Object-Oriented POMDP (OO-POMDP) which factors the state and observation spaces in terms of objects [4]. To solve this OO-POMDP, we first propose a per-voxel observation model which drastically reduces the size of the observation necessary for planning. Then, we present an octree-based belief representation which captures beliefs at different resolutions and allows efficient exact belief update. Finally, building on recent advances in online

This is a preprint.

POMDP algorithms [11] and abstraction for MDPs [12, 13], we propose an online multi-resolution planning algorithm that leverages the octree belief structure. In this approach, *abstract* OO-POMDP problems at lower resolutions are derived from the ground OO-POMDP problem, and a Monte-Carlo Tree Search (MCTS) based algorithm is employed to solve them simultaneously. The action with highest associated Q-value in its respective MCTS tree is selected for execution.

We evaluate our method in a simulated, discretized 3D domain where a robot with a 6 degrees-of-freedom camera searches for objects randomly generated and placed in the world. The results show that our approach significantly outperforms the baselines without resolution hierarchy as problem size scales and under different levels of sensor noise. We further implement our approach on a torso-actuated mobile robot; The robot can find multiple target objects in areas of a cluttered lab environment where objects appear on surfaces at different heights. This paper shows that such challenging POMDPs can be solved online efficiently with theoretical guarantees at the extreme [11].

In summary, we make the following novel contributions:

- We formulate the MOS-3D problem as an OO-POMDP; The formulation is applicable to any type of robot with localization and navigation capabilities.
- We propose a per-voxel observation model to model the volumetric observation through the viewing frustum projected from a mounted camera which enables online POMDP planning.
- We propose *octree belief*, a novel octree-based belief representation which captures the belief of object locations at different resolutions and show that it affords efficient and exact belief update and belief sampling.
- Furthermore, we derive an abstract observation model and an abstraction weighting scheme, leveraging properties of the octree belief. This results in a novel multi-resolution planning algorithm built upon POUCT [11] that outperforms baselines in simulated experiments.

II. Background

The task of 3D object search by controlling sensing parameters is NP-complete [1]. State-of-the-art methods often employ inference over prior semantic knowledge [3], reduce the state space from 3D to 2D [4, 14, 15, 16], constrain the sensor to be stationary [8, 17], or assume objects are not fully occluded [9]. Several consider physical interaction with the objects during search [5, 8, 9, 17, 18, 19]. In our work, the robot actively moves the mounted camera, for example, through pan or tilt, or through mobile base movements.

Object search is often formulated as a POMDP (Partially Observable Markov Decision Process). Aydemir et al. [3] calculates candidate viewpoints in a 2D plane based on the distribution over the search region. Atanasov et al. [6] considers the motions of the sensor on a sphere observing a tabletop scene. Li et al. [9] assumes all objects exist at the same surface level. Wandzel et al. [4] proposes OO-POMDPs for the multi-object search task in 2D. Xiao et al. [5] attempts to tackle 3D object search in clutter. In this work, found objects are permanently removed from the scene during searching, and the robot is constrained to move between two viewpoints around the scene. Novkovic et al. [19] includes both active movement of sensor on the end effector and manipulation of objects, yet using a guided shaped reward function.

Our work focuses on the multi-object search problem in 3D without reduction in state and action spaces. We use a sparse reward based on achieving the goal.

A. POMDPs and OO-POMDPs

POMDP is a framework for describing sequential decision making problems where the agent does not observe the full environment state. Formally, a POMDP is 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 of the problem. When the agent takes action $a \in \mathcal{A}$, the environment state transitions from $s \in \mathcal{S}$ to $s' \in \mathcal{S}$ following the transitional probability distribution $T(s, a, s') = \Pr(s'|s, a)$. As a result of the action and the transition, the agent receives an observation $o \in \mathcal{O}$ following the observational probability distribution $O(s', a, o) = \Pr(o|s', a)$ and reward $R(s, a) \in \mathbb{R}$. A *history* $h_t = (a_1, o_1, \cdots, a_{t-1}, o_{t-1})$ captures all past actions and observations. The agent maintains a distribution over states given current history $b_t(s) = \Pr(s|h_t)$, referred to as a *belief state*. The agent updates its belief after taking an action and receiving an observation; The exact belief update is given by,

$$b_{t+1}(s') = \frac{\Pr(o|s', a) \sum_s \Pr(s'|s, a) b_t(s)}{\sum_s \sum_{s'} \Pr(o|s', a) \Pr(s'|s, a) b_t(s)}. \quad (1)$$

The task of the agent is to find a policy $\pi(h_t)$ which maximizes the expectation of future discounted rewards with a discount factor γ:

$$V^{\pi}(h_t) = \mathbb{E}\left[\sum_{k=t}^{\infty} \gamma^{k-t} R(s_k, a_k) \;\middle|\; a_k = \pi(h_k) \right]. \quad (2)$$

OO-POMDP [4] (based on OO-MDP [20]) is a particular kind of POMDP that considers the state and observation spaces to be factored by a set of n objects, namely, $\Pr(s'|s, a) = \prod_i \Pr(s'_i|s, a)$ and $\Pr(o|s', a) = \prod_i \Pr(o_i|s', a)$ where $1 \leq i \leq n$. The belief is also factored $b_t(s) = \prod_i b^i_t(s_i) = \prod_i \Pr(s_i|h_t)$, thus can be updated separately for each object. The benefit of using object-oriented factoring for the object search task is that the belief space size grows linearly rather than exponentially as the number of objects increases. In this paper, we use the common notations for POMDPs instead of the 10-tuple OO-POMDP notation. We denote a state or observation about object i using a subscript i, and a belief about i using a superscript, as shown above.

B. POMDP Solvers

Offline POMDP solvers are often not applicable to large POMDP problems due to the time required to compute a policy [21]. Recent online POMDP solvers leverage sparse belief sampling and MCTS to mitigate the curse of dimensionality and the curse of history [11, 22, 23]. State-of-the-art online

solvers include POUCT [11] which extend the UCT algorithm [24] to partially observable domains. Silver and Veness [11] also combines POUCT with particle belief representation to form POMCP [11]. DESPOT [22, 25] is another leading POMDP solver which uses a sparse approximation of the belief tree. In this work, we build upon POUCT, due to its simplicity and theoretical guarantee of optimality.

C. State Abstraction in (PO)MDPs

Our planning algorithm builds upon our intuition that for object search, spatial state abstraction can lead to observation space reduction. State abstraction is realized by an *abstraction function* $\phi : \mathcal{S} \to \hat{\mathcal{S}}$, such that $\hat{s} = \phi(s)$ is the abstract state for ground state s. The *inverse image* $\phi^{-1}(\hat{s})$ is the set of ground states that correspond to $\hat{s}$ under ϕ [12].

A *weighting function* w is required so that transition and reward functions can be written in a Markovian way [13]. Suppose for each $\hat{s} \in \mathcal{S}, \sum_{s \in \phi^{-1}(\hat{s})} w(s) = 1$, then:

$$\Pr(\hat{s}'|\hat{s}, a) = \sum_{s \in \phi^{-1}(\hat{s})} \sum_{s' \in \phi^{-1}(\hat{s}')} \Pr(s'|s, a) w(s), \quad (3)$$

$$R(\hat{s}, a) = \sum_{s \in \phi^{-1}(\hat{s})} R(s, a) w(s). \quad (4)$$

For an abstract state $\hat{s}$, the weight $w(s)$ approximates the *occupancy probability* [1] $\Pr(s|\hat{s})$. Bai et al. [13] suggests that the true occupancy probability is non-stationary and depends on the history of past actions and abstract states. Therefore, it cannot be reasonably approximated as a constant weighting function. For our problem, since object states are not fully observable, the notion of history of abstract states cannot be established for planning. Hence, In this work, we consider a weighting scheme using POMDP belief, which is a valid, non-stationary weighting function. This is explained in detail in Section V.

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 is able to localize itself and move between locations in the search space; Note that this assumption does not necessarily require the robot to be equipped with an environment map prior to searching. The search space 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. Since our approach deals with the search space at different resolutions, 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.

Next, we describe in detail the necessary components to formulate the problem of Multi-Object Search in 3D (MOS-3D) as an OO-POMDP.

[1] Occupancy probability is originally a term in Physics (e.g. [26])

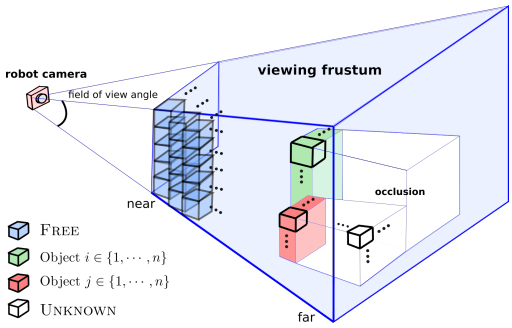

Fig. 2: Illustration of the viewing frustum and volumetric observation. The viewing frustum V consists of $|V|$ voxels, where each $v \in V$ can be labeled as $i \in \{1, \cdots, n\}$, Free or Unknown.

State space. A state of the environment $s = \{s_r, s_1, \cdots, s_n\}$ 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 contains an attribute of the 6D *camera pose* as well as an attribute of the set of *found objects*. The robot state is assumed to be observable to the robot. An object state s_i is described by the attribute of 3D *object pose* at the object's center of gravity, which corresponds to a cell in grid G. Therefore, for simplicity, we say $s_i \in G$. Note that in Section V, we work with object states at different resolution levels. 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. The robot receives an observation through a viewing frustum projected from a mounted camera (Figure 2). 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 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 $i \in \{1, \cdots, n\}$, Free, or Unknown. Free denotes that the voxel is a free space or an obstacle. We include the label Unknown in order 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. Thus, an observation $o = \{(v, d(v)) \mid v \in V\}$ is a set of voxel-label tuples. This can be thought of as the result of voxelization and object segmentation given the raw point cloud.

Here, we describe how o can be factored by objects. 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 $v \in V_i$, $d(v) \in \{i, \text{Free}\}$. Hence, $o = \bigcup_{i=1}^n o_i$ where $o_i = \{(v, d(v)) \mid v \in V_i\}$.

Action space. Searching for objects generally requires three basic capabilities: moving, looking, and declaring an object to

be found. The robot receives an observation through *looking*, and signals a commitment to its belief of the object location through *declaring*. The correctness of declarations can only be determined by, for example, a human who has knowledge about the target objects; Such feedback is not always available and may impose behavior on humans. Therefore, we consider *declaring* to be significantly more costly than the other actions.

Formally, the action space consists of these three types of primitive actions. First, $\textsc{Move}(s_r, g)$ action moves the robot from pose given in s_r to a destination location g. Then, $\textsc{Look}(\theta)$ changes the camera pose to look in the direction specified by $\theta \in \mathbb{R}^3$, and projects a viewing frustum V. Finally, $\textsc{Find}(i, g)$ decalres object i to be found at location g. The implementation of these actions may vary depending on the type of search space or robot.

Reward function. We define a sparse reward function where the robot receives $+1000$ if an object is correctly identified by a FIND action, otherwise the FIND action incurs a -1000 penalty. MOVE and LOOK actions receive a step cost of -1; MOVE receives an additional penalty based on the euclidean distance between the destination and robot location.

Transition function. The transition function is deterministic. Target objects as well as obstacles are static objects, thus $\Pr(s'_i|s, a) = \mathbf{1}(s'_i = s_i)$ for any action. For the robot, we assume it can localize itself perfectly at the resolution of G. Therefore, action $\textsc{Move}(s_r, g)$ changes the camera location to g deterministically, and $\textsc{Look}(\theta)$ changes the camera to look in the direction of θ deterministically. The $\textsc{Find}(i, g)$ action adds i to the set of *found objects* in the robot state only if g is within the viewing frustum determined by s_r. The assumption of deterministic robot state transition, also employed in prior work [4, 27], separates the localization problem from the object search problem.

Observation model. We have 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. Note that an observation is only received when action $a = \textsc{Look}(\cdot)$.

Modeling a distribution over a 3D volume is a challenging, unsolved problem of active research [28, 29]. Representing the geometry of an object requires additional prior knowledge and complexity, that may not lead to significantly different planning behavior to find the object. Thus, instead, we propose a per-voxel observation model which is sufficient for planning, under a key assumption that an object i is contained within a single voxel located at the grid cell $g = s_i$. This assumption drastically reduces the size of the observation space $\mathcal{O}_i$ from a set of varying-sized voxels per object, to just one voxel-label pair per object.

This key assumption is based on the intuition that voxels of a static object, such as a smart phone or a book, typically appear around the center of gravity of the object. We emphasize that this assumption is only used to formulate this observation model, but not the actual observation the robot may receive.

Under this assumption, there are simply two cases: the voxel corresponding to the given object location s'_i is contained in V_i, or not. When $s'_i \notin V_i$, either $d(s'_i) = \textsc{Unknown}$ or $s'_i \notin V$. In this case, $\Pr(o_i|s', a)$ is a uniform distribution, as the observation o_i contains no information regarding the object state. When $s'_i \in V_i$, then either $d(s_i) = i$, indicating correctly identifying the object, or $d(s_i) = \textsc{Free}$, indicating sensing failure. For any other voxel $v \in V_i$ with $v \neq s_i$, $d(v) = \textsc{Free}$; The event of sensing error regarding these voxels is all captured by the failure case where $d(s_i) = \textsc{Free}$. Hence, in this case, the observation o_i, when given state s', can be reduced from a set to just a single voxel-label tuple, $(s'_i, d(s_i))$. Thus, $\Pr(o_i|s', a) = \Pr(s'_i, d(s_i)|s', a) = \Pr(d(s_i)|s', a)$. We define $\Pr(d(s_i) = i|s', a) = \alpha$ and $\Pr(d(s_i) = \textsc{Free}|s', a) = \beta$. Thus, α and β are the parameters which control the reliability of the observation model. Note that the belief update in Equation 1 does not require α and β to be normalized probabilities.

IV. Octree Belief Representation

One premise in [4] is that factoring the state space by objects significantly reduces the size of the belief space, thus permitting exact belief representation and exact belief update. However, because the exact belief update requires nested iterations over the state space, the tabular exact belief representation used in [4] may be manageable in 2D, but it is not scalable to 3D domains. 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.

To address this issue, we propose a representation for belief over 3D object locations (Figure 3). The representation is constructed incrementally as observations are received, in the structure of an *octree*, which allows efficient belief sampling. Additionally, by leveraging the per-voxel observation model, we present a belief update algorithm that only concerns grid cells associated with voxels in V that carry information. Thanks to the octree structure, the belief update at the ground level can automatically propogate upwards in the tree which allows the maintenance of belief at multiple resolutions. We exploit this property of the octree belief and develop a hierarchical planning algorithm that leverages the resolution layers in the octree structure as levels of abstractions to the original OO-POMDP problem (see Section III).

A. Definition

We represent a belief state $b^i_t(s_i)$ for object i as an *octree*. 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 $l - 1$. Each node has a value $\textsc{Val}^i_t(g^l) \in \mathbb{R}$, which represents the unnormalized belief that $s^l_i = g^l$, that is, object i is located at grid cell g^l. Denote the set of nodes at resolution level $k < l$ that reside in a subtree rooted at g^l as $\textsc{Ch}^k(g^l)$. By definition,

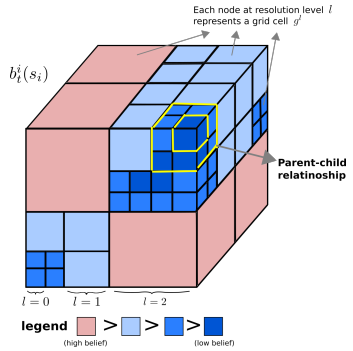

Fig. 3: Illustration the octree belief representation $b_t^i(s_i)$. The color on a node g^l indicates the belief $\textsc{Val}_t^i(g^l)$ that the object is located within g^l. The highlighted grid cells indicate parent-child relationship between a grid cell at resolution level $l = 1$ (parent) and one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

for $b_t^i(g^l) = \Pr(g^l|h_t) = \sum_{c\in \textsc{Ch}^k(g^l)} \Pr(c|h_t)$. Thus, with a normalizer $\textsc{Norm}_t = \sum_{g\in G} \textsc{Val}_t^i(g)$, we can rewrite the normalized belief as:

$$b_t^i(g^l) = \frac{\textsc{Val}_t^i(g^l)}{\textsc{Norm}_t} = \sum_{c\in \textsc{Ch}^k(g^l)} \left(\frac{\textsc{Val}_t^i(c)}{\textsc{Norm}_t}\right), \tag{5}$$

which means $\textsc{Val}_t^i(g^l) = \sum_{c\in \textsc{Ch}^k(g^l)} \textsc{Val}_t^i(c)$.

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 $\textsc{Val}_0^i(g^l) = \sum_{c\in \textsc{Ch}^1(g^l)} \textsc{Val}_0^i(c) = |\textsc{Ch}^1(g^l)|$.

We refer to this representation of belief state as *octree belief*. It is a structure which can yield the belief of object state at different resolution levels. Next, we describe the belief update and sampling algorithm for the octree belief representation.

B. Belief Update

Previously, we defined a per-voxel observation model for $\Pr(o_i|s', a)$ which is a uniform distribution if $s_i' \notin V_i$, or reduces o_i to simply $d(s_i')$ if $s_i' \in V_i$. This suggests that the belief update only needs to happen for voxels that are in 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 $v \in V_i$ from $\textsc{Val}_t^i(g)$ to $\textsc{Val}_{t+1}^i(g)$ using the observation probability $\Pr(d(v)|s', a)$. This essentially updates the belief of $s_i = g$ to $b_{t+1}^i(s_i)$. The normalizer is updated to make sure b_{t+1}^i is normalized; The updated normalizer should only be valid after all voxels in V have been processed, which means:

$$\textsc{Norm}_{t+1} = \textsc{Norm}_t + \sum_{s_i\in V_i} \left(\textsc{Val}_{t+1}^i(s_i) - \textsc{Val}_t^i(s_i)\right). \tag{6}$$

This update equation implies that for grid cells outside of the FOV, the beliefs are kept the same. This is consistent with our observation model definition, where $\Pr(o_i|s', a)$ is a uniform distribution under this situation. Since this is an unnormalized probability, we can set $\Pr(o_i|s', a) = 1$ regardless of o_i. Hence, for all $s_i \notin V_i$, $\textsc{Val}_{t+1}(s_i) = \Pr(o_i|s', a)\textsc{Val}_t(s_i) = \textsc{Val}_t(s_i)$.

This algorithm has a complexity of $O(|V|\log(|G|)$ where $|V|$ is the size of the viewing frustum and $|G|$ is the size of the search space; Inserting nodes and updating values of nodes in the octree only requires traversing the tree depth-wise.

Next, we provide a lemma to justify the normalizer update, and a theorem that shows this belief update procedure is exact.

Algorithm 1: OctreeBeliefUpdate $(b_t^i, o_i, V_i) \to b_{t+1}^i$

```
// Let Ψ(b_i^t) denote the octree underlying b_i^t.
for v ∈ V_i do
    s_i ← v;   // State s_i ∈ G at grid cell corresponding to voxel v
    if g ∉ Ψ(b_i^t) then
        Insert node at g to Ψ(b_i^t);
    end
    // Now computing the belief update b_{t+1}^i(s_i)
    Val_{t+1}^i(s_i) ← Pr(d(v)|s', a) Val_t^i(s_i);
    Norm_{t+1} ← Norm_t + Val_{t+1}^i(s_i) − Val_t^i(s_i);
end
```

Lemma. (Normalizer Update) *The normalizer* $\textsc{Norm}_t$ *at time* t *can be updated by adding the incremental update of values as in Equation (6).*

Proof: We derive equation (6) as follows:

$$\textsc{Norm}_{t+1} = \sum_{s_i\in G} \textsc{Val}_{t+1}^i(s_i) \tag{7}$$

$$= \sum_{s_i\in V_i} \textsc{Val}_{t+1}^i(s_i) + \sum_{s_i\notin V_i} \Pr(o_i|s', a)\textsc{Val}_{t+1}^i(s_i) \tag{8}$$

$$= \sum_{s_i\in V_i} \textsc{Val}_{t+1}^i(s_i) + \sum_{s_i\notin V_i} \textsc{Val}_{t+1}^i(s_i) \tag{9}$$

$$= \sum_{s_i\in V_i} \textsc{Val}_{t+1}^i(s_i) + \sum_{s_i\in G} \textsc{Val}_t^i(s_i) - \sum_{s_i\in V_i} \textsc{Val}_t^i(s_i) \tag{10}$$

$$= \sum_{s_i\in V_i} \textsc{Val}_{t+1}^i(s_i) + \textsc{Norm}_t - \sum_{s_i\in V_i} \textsc{Val}_t^i(s_i) \tag{11}$$

$$= \textsc{Norm}_t + \sum_{s_i\in V_i} \left(\textsc{Val}_{t+1}^i(s_i) + \textsc{Val}_t^i(s_i)\right) \tag{12}$$

Equation (7) is the definition of the normalizer at time $t+1$. We can decompose Equation (8) into two cases where the object i is inside of V_i and outside of V_i. In the latter case, since the observation model is a uniform distribution, the unnormalized probability $\Pr(o_i|s', a) = 1$ for all s_i not in V_i, resulting in Equation (9). Equation (10) can be obtained by set complement of $s_i \in V_i$. Equation (11) is derived by the normalizer definition. Finally, Equation (12) proves Equation (6). ■

Theorem 1. (Belief Update) *The updated belief* b_{t+1}^i *according to the belief update algorithm where for any* $s_i \in G$,

$b^i_{t+1}(s_i) = \textsc{Val}^i_{t+1}(s_i)/\textsc{Norm}_{t+1}$, *is equivalent to the exact belief update rule in Equation (1), given the context of the MOS-3D problem and the use of the per-voxel observation model.*

Proof: We start with the exact belief update for the belief over states of object i. We rewrite Equation (1) here for a complete proof:

$$b^i_{t+1}(s'_i) = \frac{\Pr(o_i|s',a)\sum_s \Pr(s'_i|s,a)b^i_t(s_i)}{\sum_s\sum_{s'}\Pr(o_i|s',a)\Pr(s'_i|s,a)b^i_t(s_i)} \tag{13}$$

Because objects are static, $\Pr(s'_i|s,a) = \mathbf{1}(s'_i = s_i)$. Therefore, Equation (13) becomes:

$$b^i_{t+1}(s'_i) = \frac{\Pr(o_i|s',a)b^i_t(s'_i)}{\sum_{s'}\Pr(o_i|s',a)b^i_t(s'_i)} \tag{14}$$

$$= \frac{\Pr(o_i|s',a)\frac{\textsc{Val}^i_t(s'_i)}{\textsc{Norm}_t}}{\sum_{s'}\Pr(o_i|s',a)\frac{\textsc{Val}^i_t(s'_i)}{\textsc{Norm}_t}} \tag{15}$$

$$= \frac{\Pr(o_i|s',a)\textsc{Val}^i_t(s'_i)}{\sum_{s'}\Pr(o_i|s',a)\textsc{Val}^i_t(s'_i)} \tag{16}$$

$$= \frac{\textsc{Val}^i_{t+1}(s'_i)}{\textsc{Norm}_{t+1}} \tag{17}$$

Substituting the belief definition into $b^i_t(s'_i)$ in Equation (14) gives Equation (15). The term $\textsc{Norm}_t$ dies out to become Equation (16). The numerator of Equation (16) is equivalent to $\textsc{Val}^i_{t+1}(s'_i)$ while the denominator of Equation (17) becomes $\textsc{Norm}_{t+1}$ according to Lemma 1. This concludes the proof. ■

C. Sampling

Besides exact belief update, we show that octree belief also affords exact belief sampling in logarithmic time complexity with respect to the size of the search space $|G|$.

For object i, the probability that state $s^l_i \in G^l$ is sampled from the belief distribution is the belief $b^i_t(s^l_i) = \Pr(s^l_i|h_t)$. However, since the tree may not be completely built at the resolution level l, a complete mapping from $\mathcal{S}^l_i$ to probability is not readily available. Instead, we sample from $\mathcal{S}^l_i$ 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. The goal of the sampling algorithm is to output a sample $s^{l_{des}}_i$ with probability $b^i_t(s^{l_{des}})$. To achieve this, first note that, if $s^{l_{des}}_i$ is sampled, then all nodes in the octree that cover $s^{l_{des}}_i$ is also implicitly sampled, which form a sequence of nodes $s^{l_{max}}_i, \cdots, s^{l_{des}+2}_i, s^{l_{des}+1}_i$, starting from the root of the octree $\Psi(b^i_t)$ down to the resolution level right above l_{des}. Also, the event that s^{l+k}_i is sampled is independent from other samples given that s^{l+k+1}_i is sampled. Hence,

$$\begin{aligned}&\Pr(s^{l_{des}}_i|h_t)\\ &= \Pr(s^{l_{max}}_i, \cdots, s^{l_{des}+2}_i, s^{l_{des}+1}_i, s^{l_{des}}_i|h_t)\\ &= \Pr(s^{l_{des}}_i|s^{l_{des}+1}_i, h_t)\cdots\Pr(s^{l_{max}-1}_i|s^{l_{max}}_i, h_t)\end{aligned} \tag{18}$$

Therefore, the task of sampling $s^{l_{des}}$ is translated into sampling a sequence of samples $s^{l_{max}}_i, \cdots, s^{l_{des}+2}_i, s^{l_{des}+1}_i, s^{l_{des}}_i$, each according to the distribution $\Pr(s^l_i|s^{l+1}_i, h_t) = \frac{\textsc{Val}^i_t(s^l_i)}{\textsc{Val}^i_t(s^{l+1}_i)}$. Sampling from this probability distribution is efficient, as the sample space, i.e. the children of node s^{l+1}_i is only of size 8. Therefore, this sampling scheme yields a sample $s^{l_{des}}$ exactly according to $b^i_t(s^{l_{des}})$ with time complexity $O(\log(|G|))$.

V. Multi-Resolution Planning via Abstractions

We exploit the belief at different resolutions encoded in the octree belief for planning; Specifically, we aim at reducing the branching factor due to large observation space in MCTS through spatial state abstraction. Next, we introduce state abstraction for OO-POMDPs, followed by the *abstract MOS-3D* description that can be derived from the original MOS-3D problem, which solves the same task approximately.

In our context, we consider object-oriented factoring, such that $\hat{s} = \phi(s) = \bigcup_i \phi_i(s_i)$, where $\phi_i : \mathcal{S}_i \to \hat{\mathcal{S}}_i$ is the abstraction function for the state space of object i. Under the assumption that the transition of object states is independent given current state, $\Pr(\hat{s}'|\hat{s},a) = \prod_i \Pr(\hat{s}'_i|\hat{s},a)$. Given weighting function w such that $w(s_i) = \sum_{s_i\in\phi_i^{-1}(\hat{s}_i)} w(s_i) = 1$

$$\Pr(\hat{s}'_i|\hat{s},a) = \sum_{s\in\phi^{-1}(\hat{s})}\sum_{s'_i\in\phi_i^{-1}(\hat{s}'_i)} \Pr(s'_i|s,a)w(s_i) \tag{19}$$

As Li et al. [12], it can be shown that $\sum_{\hat{s}_i}\Pr(\hat{s}'_i|\hat{s},a) = 1$, making this a valid transition function. Hence, abstract state transition in OO-POMDP occurs independently for each object i (including the robot) following Equation (19). Next, we describe how this extends to the abstraction of observations for the MOS-3D problem.

A. From State Abstraction to Observation Abstraction

We consider spatial abstraction of object states via an abstraction function $\phi_i : \mathcal{S}_i \to \mathcal{S}^l_i$, which transforms, for object i, from the ground-level object state s_i to an *abstract object state* s^l_i 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 it is. For any abstract state s^l, the weight $w(s_i)$ where $s_i \in \phi_i^{-1}(s^l_i)$ is computable through octree belief:

$$w(s_i) = \Pr(s_i|s^l_i, h_t) = \frac{\Pr(s_i|h_t)}{\Pr(s^l_i|h_t)} = \frac{\textsc{Val}^i_t(s_i)}{\textsc{Val}^i_t(s^l_i)} \tag{20}$$

This is valid since $\sum_{s_i\in\phi_i^{-1}(s^l_i)}\Pr(s_i|s^l_i,h_t) = 1$

Using state abstraction, the sample space of the belief reduces in size to $|G^l|$. To reduce the branching factor of MCTS planning, we design an *abstract observation model* for object i from which an *abstract observation* o^l_i can be sampled. The abstract observation o^l_i is a set of voxel-label tuples $(v^l, d(v^l))$, where each voxel v^l is at resolution level l, and $d(v^l) \in \{i, \textsc{Free}\}$. Again, as in the observation model formulation (Section III), we can make use of the assumption that an object is contained within a single voxel given state $\hat{s}$ and regard o^l_i as a single voxel-label pair $(s^l_i, d(s^l_i))$. The abstract observation model is given by $\Pr(o^l_i|\hat{s}',a,h_t)$; h_t is

required since the problem is no longer Markovian due to state abstraction. Then, for $\hat{s} \in \mathcal{S}^l$, it can be shown that

$$\Pr(o_i^l|\hat{s}', a, h_t) = \Pr(s_i^l, d(s_i^l)|\hat{s}', a, h_t) \tag{21}$$

$$= \sum_{s_i \in \phi_i^{-1}(s_i^l)} \Pr(d(s_i^l)|s_i, \hat{s}', a, h_t) w(s_i) \tag{22}$$

The full derivation is provided in the appendix.

Regarding $\Pr(d(s_i^l)|s_i, \hat{s}', a, h_t)$, there are again two cases: s_i is in the FOV V_i defined by $s_r \in \hat{s}'$, or not. When $s_i \notin V_i$, this probability is uniform, and we let $\Pr(d(s_i^l)|s_i, \hat{s}', a, h_t) = 1$ for consistency with the octree belief update. When $s_i \in V_i$, the probability is again an indication of sensor behavior. As in the ground level observation model (Section III), $\Pr(d(s_i^l) = i|s_i, \hat{s}', a, h_t) = \alpha$, indicating correct detection, and $\Pr(d(s_i^l) = \textsc{Free}|s_i, \hat{s}', a, h_t) = \beta$, indicating sensor misbehavior. It may be inefficient to sample from this abstract observation model, if the resolution level l is high. In our simulation experiments, we approximate this observation model by sampling k ground states from $\phi_i^{-1}(s_i^l)$ according to their weights. 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. Our empirical evaluation suggests this approach leads to significant performance improvement.

B. Action abstraction

Since state abstraction lowers the resolution of the search space, it is natural to consider the benefit of moving the robot over a longer distance at each planning step. Therefore, we consider a simple scheme of temporal abstraction over MOVE actions. Following the *options framework* [30], each move option, denoted by $\textsc{MoveOp}(s_r, g)$, has an initiation set of the current location of the robot in s_r, a policy $\pi : \mathcal{S}_r \rightarrow \textsc{Move}$ that produces primitive MOVE actions, and a termination condition of whether the robot reaches g. Note that no observation is received during the execution of a MOVEOP. The primitive LOOK and FIND actions are kept unchanged.

C. Abstract MOS-3D

Using the techniques above, state, observation, and action abstractions leads to abstract spaces $\hat{\mathcal{S}}$, $\hat{\mathcal{A}}$, $\hat{\mathcal{O}}$, as well as abstract state transition function $\hat{T}$ and observation function $\hat{O}$. Since the state and action abstractions are derived directly from the ground level state space $\mathcal{S}$ and primitive action space $\mathcal{A}$, the reward function of the original MOS-3D problem can be reused. Hence, we arrive at an *abstract* OO-POMDP problem $\langle \hat{\mathcal{S}}, \hat{\mathcal{A}}, \hat{\mathcal{O}}, \hat{T}, \hat{O}, R, \gamma \rangle$ that solves the same task as the original MOS-3D problem but smaller in size, and it is parameterized by a resolution level l, and a set of motion options. We refer to this problem as *Abstract MOS-3D*.

D. Multi-Resolution Planning Algorithm (MR-POUCT)

Abstract MOS-3D are smaller than the original MOS-3D 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.

To this end, we propose an online planning algorithm based on MCTS which solves a number of Abstract MOS-3D problems in parallel, and selects an action from $\hat{\mathcal{A}}$ with the highest Q-value for execution; The algorithm is formally presented in Algorithm 2. The algorithm takes as input the current history h_t, the planning horizon H, and a set of Abstract MOS-3D problems $\mathcal{P}$, which can be defined based on the dimensionality of the search space and the particular object search setting. Each problem $P^l \in \mathcal{P}$ at resolution level l is given by the tuple $\langle \mathcal{S}^l, \hat{A}, \mathcal{O}^l, \hat{T}, \hat{O}, R, \gamma \rangle$. Note that P^l has associated motion options in addition to primitive LOOK and FIND, together forming the action space $\hat{\mathcal{A}}$. Then, a *generative function* is derived from an Abstract MOS-3D instance P^l, which is used directly by the POUCT algorithm to sample state transition, observation, and reward. Thus, all problems in $\mathcal{P}$ are solved online in parallel, each by a separate POUCT. Since POUCT constructs a value function with optimality guarantee as the number of simulations approaches infinity, each P^l is theoretically solved optimally at the extreme. The final action with the highest Q-value in its respective POUCT search tree is chosen as the output.

Algorithm 2: MR-POUCT $(\mathcal{P}, h_t, H) \rightarrow \hat{a}$

procedure Plan(h_t)
  foreach $P \in \mathcal{P}$ *in parallel* **do**
    $\mathcal{G} \leftarrow$ GenerativeFunction(P);
    $Q_P(h_t, \hat{a}) \leftarrow$ POUCT($\mathcal{G}, h_t, H$);
  end
  $\hat{a} \leftarrow \text{argmax}_{\hat{a}}\{Q_P(h_t, \hat{a})|P \in \mathcal{P}\}$;
  return $\hat{a}$
procedure GenerativeFunction *(P)*
  // Returns a function $\mathcal{G}$ that generates $(\hat{s}', \hat{o}', \hat{r}') \sim \mathcal{G}(\hat{s}, \hat{a})$
  // Recall that $P = \langle \hat{\mathcal{S}}, \hat{\mathcal{A}}, \hat{\mathcal{O}}, \hat{T}, \hat{O}, R, \gamma \rangle$
  $\mathcal{G} \leftarrow$ **func** $(\hat{s} \in \hat{\mathcal{S}}, \hat{a} \in \hat{\mathcal{A}})$
    $\hat{s}' \sim \hat{T}(\hat{s}, \hat{a}, \hat{s}')$;
    $\hat{o}' \leftarrow$ Null;
    $r \leftarrow r_{step}$; // r_{step} is the step cost
    if $\hat{a}$ is a LOOK action **then**
      $\hat{o}' \sim \hat{O}(\hat{s}', \hat{a}, \hat{o})$;
    // With $s_r \in \hat{s}, s_r' \in \hat{s}'$,
    if $\hat{a}$ is a $\textsc{MoveOp}(s_r, s_r')$ **then**
      $r \leftarrow$ CumulativeReward($s_r, \hat{a}$);
    return $\hat{s}', \hat{o}', r$
  end
  return $\mathcal{G}$

We note here that the original POUCT does not consider planning with options. Nevertheless, our algorithm only makes use of abstract *motion* actions MOVEOP which are broken down into deterministic primitive actions with no observation received in intermediate steps. Therefore, a MOVEOP can be treated as a single motion action; By taking a MOVEOP option, the agent receives the discounted cumulative reward of each MOVE action that form the policy for the option. Please refer to Silver and Veness [11] for details of the POUCT

algorithm; The algorithm itself is provided in the appendix (Section VIII-B).

VI. Evaluation

We first describe an implementation of MOS-3D in a simulated robot, where we evaluate the scalability of the algorithm, and its ability to handle sensor uncertainty. We then demonstrate its functionality on a torso-actuated mobile robot in a cluttered lab setting.

A. Simulated Domain

The simulated domain aims to reflect the essence of the MOS-3D problem and investigate the properties of the proposed method comprehensively (Figure 4). Each instance of the simulated domain is defined by a tuple (m, n, d), where the robot is tasked to search for n randomly generated, randomly placed objects in a search space G with size $|G| = m^3$. Initially, the robot has a *uniform prior* over object locations. The robot is equipped with a camera that projects a viewing frustum with a FOV angle of 45 degrees, an aspect ratio of 1.0, a near plane at 1.0, and a far plane at distance d grid cells away from the robot. The near plane of 1.0 means that the viewing frustum can include an object one grid cell away from the robot, and a larger value of d means the viewing frustum can capture more voxels of the search space. The purpose of this design is that 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 which directly correlates with the FOV range d.

The space of primitive MOVE actions is as follows. Along each axis in the Cartesian coordinate system, there are two primitive motion actions corresponding to both directions of the axis, resulting in six in total. For example, along the x axis, actions $+x$ or $-x$ increases or decreases the x component by 1 grid cell. There are six LOOK actions, one for each direction of every axis. Finally, there is one FIND action, which declares not-yet-observed all objects within the viewing frustum as found. Thus, the total number of primitive actions is 13. If multiple objects are present within one viewing frustum when the FIND is taken, only the maximum reward of $+1000$ is provided. If no new object is present in the viewing frustum and FIND is taken, the agent receives a negative reward of -1000. The task terminates either when the total planning time limit is reached or n FIND actions are taken.

Implementation of Abstract MOS-3D. As described in Section V, to define an Abstract MOS-3D, we specify a resolution level l as well as a set of MOVEOP options. Similar to the ground problem, the agent has a total of six MOVEOP options; Each MOVEOP moves the robot along a certain direction for m_{step} grid cells. For example, when $m_{step} = 4$, one MOVEOP along $+x$ direction can be decomposed into 4 primitive MOVE actions along the $+x$ direction. Hence, an Abstract MOS-3D in the simulated domain can be defined by a tuple (l, m_{step}).

B. Setup

We perform two sets of evaluations, one on the *scalability* of the algorithm, and the other on *sensor quality*, which tests the ability of the algorithm to handle different noise settings of the sensor model.

In these experiments, we compare our approach with the following baselines. First, we investigate the benefit of object-oriented factoring the MOS-3D problem versus no factoring by comparing *POUCT* with per-object octree belief representation[2] against *POMCP* [11], which uses particle belief representation without object-oriented factoring. Second, more interestingly, we investigate the benefit of leveraging the belief over the octree resolution hierarchy by comparing our algorithm, *Multi-Resolution POUCT* (MR-POUCT) against flat POUCT which solves the ground POMDP problem directly. We also conduct an ablation test where we remove the state abstraction from MR-POUCT but keep the action abstraction, which means the agent can plan to move at larger step sizes, but does not consider spatial state or observation abstraction and only has access to ground-level belief. We refer to this baseline as *MA-POUCT* (Macro-Action POUCT). As a lower bound, we also provide the performance of a *Random* agent which executes actions at random.

In both experiment settings, each algorithm 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. We set discount factor $\gamma = 0.99$. The undiscounted cumulative reward as well as the number of detected objects are reported to measure the performance.

Scalability. We experimented with 4 different settings of search space size $m \in \{4, 8, 16, 32\}$ and 3 settings of number of objects $n \in \{2, 4, 6\}$. The FOV range d is chosen such that the upper bound on the percentage $\rho(d)$ of the grids covered by one projection of the viewing frustum decreases as the world size m increases. Precisely, when $m = 4$, $\rho(4) \approx 17.2\%$; when $m = 8$, $\rho(6) \approx 8.8\%$; when $m = 16$, $\rho(10) \approx 4.7\%$; when $m = 32$, $\rho(16) \approx 2.6\%$. Essentially, as $\rho(\cdot)$ decreases, the object search problem becomes more difficult since one LOOK action results in a smaller volume of observation. The sensor is assumed to be near-perfect, with $\alpha = 10^5$ and $\beta = 0$.

Sensor Quality. In this experiment, we investigate the sensitivity of our method with respect to changes in the parameters α and β of the observation model. According to the belief update algorithm in Section IV-B, 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 $\beta \in \{0.3, 0.8\}$. Lastly, a fixed problem difficulty of $(16, 2, 10)$ is used to conduct this experiment.

[2]We emphasize that without octree belief representation, POUCT with traditional tabular exact belief update is not applicable to solve this problem at large scale.

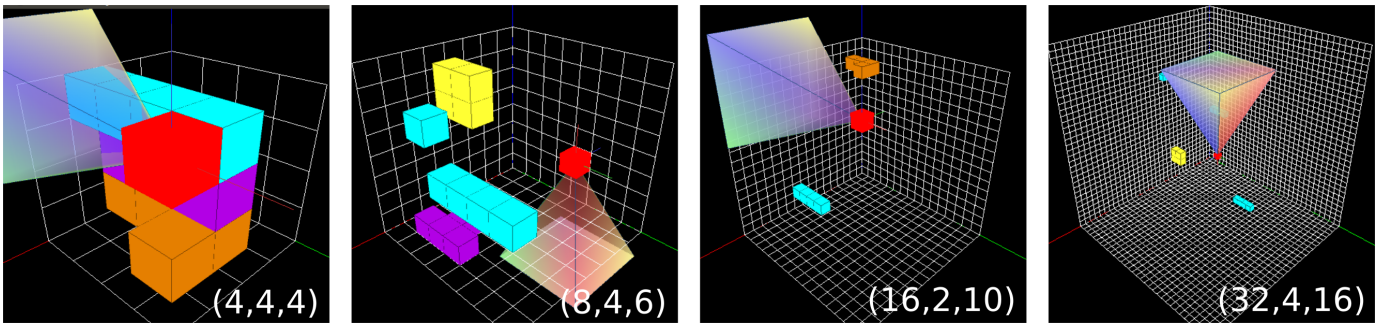

Fig. 4: Simulated environment for 3D object search. The robot (represented as a red cube) is able to project a viewing frustum to observe the search space, where objects are represented by sets of cubes. The environment can be scaled to increase difficulty of the problem; The tuple (m, n, d) at lower-right 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

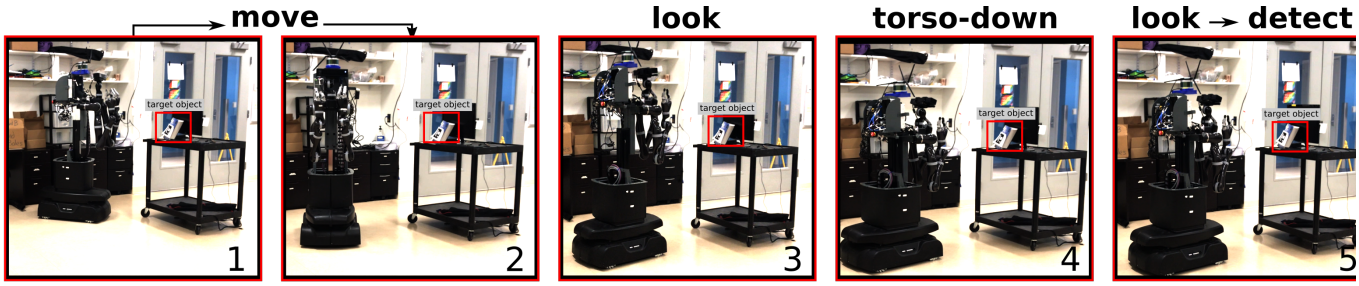

Fig. 5: The mobile robot first navigates in front of a portable table (1-2). It then takes a LOOK action to observe the space in front (3), and no target is observed as a result of this LOOK action since the torso is too high. Then, the robot decides to lower its torso (4), takes another LOOK action in the same direction, and then take 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.

C. Results and Discussions

Overall, the results indicate that our multi-resolution planning algorithm is significantly superior than all baselines, both as the difficulty of the problem increases, and under different settings of sensor noise. In Table I–IV, we provide the full evaluation results. Each data point is aggregated from 40 trials; In each trial, an instance of MOS-3D is generated and the same instance is used to evaluate all planning algorithms. We discuss these results below.

Scalability. First, we notice clear advantage of exact belief update and object-oriented factoring over the alternative particle based representation without object-oriented factoring. Particle deprivation happens quickly in the simulated domain, and the behavior degenerates to a random agent as in [11, 4]. On the other hand, when the search space or the number of target objects is small (e.g. in (4,4,4) and (32,2,16)), the flat POUCT and MA-POUCT are competitive with MR-POUCT. Yet, as the problem becomes more difficult, MR-POUCT consistently outperforms these baselines by a significant margin of an average improvement in cumulative reward around $25\% \sim 35\%$ in settings with $m = 16$ or $m = 32$. Our analysis of the agent behavior indicates that MR-POUCT is able to produce search policies which explores the environment more efficiently. We observed that using only action abstraction without resolution hierarchy results in the agent hesitating to take LOOK actions to actually observe the environment, while taking many MOVEOP options which deteriorates the reward. This showcases the value of our state and observation abstraction approach leveraging octree belief. Additionally, MR-POUCT consistently finds more objects than the baselines as the problem scales. The problems with $(32, \cdot, 16)$ appears to be difficult for any approach in comparison to find multiple objects. This attributes to the fact that each viewing frustum in this scenario captures only 2% of the environment, which impacts the search efficiency.

Sensor Quality. Our results show that the parameter α, which essentially describes the likelihood for the robot to trust that a voxel is labeled by an object, affects the sensing quality more significantly than β, which describes the likelihood for the robot to trust that a voxel is labeled as FREE. In fact, our results indicate that there is no significant damage to any algorithm's performance when β varies as long as $\beta < 1$. We observed that MR-POUCT produces consistently better performance in all of the parameter settings. This attributes to the more efficient search policy that MR-POUCT is able to generate under sensing noise.

D. Demonstration on a Torso-Actuated Mobile Robot

We demonstrate that our approach is scalable to real world settings by implementing the MOS-3D problem as well as

MR-POUCT for a mobile robot setting. We use the Kinova MOVO Mobile Manipulator robot, which has an actuated torso that can raise up to around 0.5m and lower down to around 0.05m, 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, 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 could be surrounded by clutter in the lab. Thus, the robot instantiates an instance of the MOS-3D problem once it navigates to a search region. Different from the simulated domain, in this MOS-3D implementation, the MOVE actions are implemented based on a topological graph constructed on top of a metric occupancy grid map. The neighbors of a graph node form the motion action space when the robot is at that node. Since this motion action space is already an abstraction over the metric grid map, we do not impose MOVEOP to the Abstract MOS-3D in this case. 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 5 total objects in the two search regions. One sequence of actions (Figure 5) show that the robot decides to lower its torso in order to LOOK and FIND an object. Our supplementary video contains the footage with visualization of the volumetric observation and octree belief update.

VII. CONCLUSION

While prior work primarily constrain object search in 2D, we address this problem in 3D. To this end, we formulate 3D multi-object search problem as an Object-Oriented POMDP that factors the state and observation spaces by objects. We define the observation space to be volumetric, based on a realistic frustum-shaped field-of-view. To solve this POMDP, we first propose a per-voxel observation model to reduce the observation space necessary for planning, making an assumption that the object is contained within one voxel. Then, we introduce a novel belief representation, *octree belief* that captures the belief of object locations at different resolutions and allows efficient, exact belief update and sampling. Finally, we leverage the octree belief to define an abstract version of the 3D multi-object search problem. This leads to the MR-POUCT algorithm, a multi-resolution extension of POUCT [11]. We show in simulation that this algorithm is more scalable and robust against sensor noise than the baselines, and demonstrate this approach on a torso-actuated mobile robot.

This approach has its limitations. The assumption of a target object contained within one voxel does not hold for scenarios where the geometry of the objects is crucial for the search policy. Considering object geometry in belief space is a challenging direction of future work. Future work also include exploiting relations between objects, and searching for dynamic objects.

REFERENCES

[1] Yiming Ye and John K Tsotsos. Sensor planning in 3d object search: its formulation and complexity. In *The 4th International Symposium on Artificial Intelligence and Mathematics, Florida, USA*. Citeseer, 1996.

[2] Thomas Kollar and Nicholas Roy. Utilizing object-object and object-scene context when planning to find things. In *2009 IEEE International Conference on Robotics and Automation*, pages 2168–2173. IEEE, 2009.

[3] Alper Aydemir, Andrzej Pronobis, Moritz Göbelbecker, and Patric Jensfelt. Active visual object search in unknown environments using uncertain semantics. *IEEE Transactions on Robotics (T-RO)*, 29(4):986–1002, August 2013. doi: 10.1109/TRO.2013.2256686. URL http://www.pronobis.pro/publications/aydemir2013tro.

[4] Arthur Wandzel, Yoonseon Oh, Michael Fishman, Nishanth Kumar, and Stefanie Tellex. Multi-object search using object-oriented pomdps. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7194–7200. IEEE, 2019.

[5] Yuchen Xiao, Sammie Katt, Andreas ten Pas, Shengjian Chen, and Christopher Amato. Online planning for target object search in clutter under partial observability. In *Proceedings of the International Conference on Robotics and Automation*, 2019.

[6] N. Atanasov, B. Sankaran, J. Le Ny, G. Pappas, and K. Daniilidis. Nonmyopic view planning for active object classification and pose estimation. *IEEE Trans. on Robotics (TRO)*, 30(5):1078–1090, 2014. doi: https://www.doi.org/10.1109/TRO.2014.2320795.

[7] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[8] Michael Danielczuk, Andrey Kurenkov, Ashwin Balakrishna, Matthew Matl, David Wang, Robert Martn-Martn, Animesh Garg, Silvio Savarese, and Ken Goldberg. Mechanical search: Multi-step retrieval of a target object occluded by clutter. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019.

[9] Jue Kun Li, David Hsu, and Wee Sun 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)*, pages 5701–5707. IEEE, 2016.

[10] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*, 147(1-2):5–34, 2003.

[11] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pages 2164–2172, 2010.

[12] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. In *ISAIM*, 2006.

TABLE I: **Scalability** - Cumulative Reward; $(\alpha, \beta) = (10^5, 0)$
The rows are ordered in increasing levels of problem scale and difficulty.

(m, n, d)	Random	POMCP	POUCT	MA-POUCT	MR-POUCT
(4,2,4)	-2317.47 ± 183.88	-1938.95 ± 185.02	**1169.42 ± 95.88**	963.24 ± 100.84	1041.97 ± 109.35
(4,4,4)	-3103.15 ± 245.61	-2807.97 ± 229.78	2040.42 ± 119.66	**2316.54 ± 114.78**	2264.59 ± 107.36
(4,6,4)	-2843.45 ± 326.70	-3433.82 ± 318.84	3093.93 ± 146.70	3172.31 ± 146.91	**3367.83 ± 92.40**
(8,2,6)	-2314.90 ± 195.95	-2447.80 ± 147.23	1169.78 ± 113.88	389.93 ± 86.82	**1273.26 ± 95.24**
(8,4,6)	-3135.40 ± 321.45	-3418.05 ± 254.46	2390.78 ± 128.28	2012.64 ± 137.81	**2597.55 ± 144.93**
(8,6,6)	-3640.65 ± 293.89	-4005.22 ± 306.43	3164.93 ± 161.62	3482.96 ± 154.93	**3606.97 ± 134.15**
(16,2,10)	-2798.93 ± 112.13	-2701.93 ± 127.16	542.75 ± 117.93	139.29 ± 99.94	**1142.00 ± 104.93**
(16,4,10)	-4068.15 ± 209.67	-4062.35 ± 200.28	1590.58 ± 174.31	1377.36 ± 111.54	**2070.02 ± 160.76**
(16,6,10)	-5212.75 ± 306.96	-5327.35 ± 250.80	2367.70 ± 180.66	2746.30 ± 140.21	**2939.23 ± 159.55**
(32,2,16)	-2750.15 ± 120.40	-2788.78 ± 137.15	**625.55 ± 105.31**	-638.41 ± 67.09	541.76 ± 118.36
(32,4,16)	-4736.77 ± 136.49	-4758.07 ± 146.54	742.08 ± 131.86	112.57 ± 117.92	**1234.38 ± 135.73**
(32,6,16)	-5025.95 ± 322.13	-5919.20 ± 240.52	1284.08 ± 145.55	1229.17 ± 144.75	**1516.08 ± 148.35**

TABLE II: **Scalability** - Number of Detected Objects; $(\alpha, \beta) = (10^5, 0)$

(m, n, d)	Random	POMCP	POUCT	MA-POUCT	MR-POUCT
(4,2,4)	0.38 ± 0.09	0.54 ± 0.09	**1.27 ± 0.10**	1.18 ± 0.12	1.18 ± 0.12
(4,4,4)	1.18 ± 0.15	1.45 ± 0.16	2.58 ± 0.13	**3.02 ± 0.14**	2.90 ± 0.09
(4,6,4)	2.85 ± 0.25	2.58 ± 0.24	4.35 ± 0.19	**4.65 ± 0.17**	4.58 ± 0.13
(8,2,6)	0.38 ± 0.09	0.33 ± 0.07	1.25 ± 0.11	0.60 ± 0.09	**1.48 ± 0.09**
(8,4,6)	1.18 ± 0.19	1.05 ± 0.17	2.98 ± 0.13	2.77 ± 0.17	**3.12 ± 0.13**
(8,6,6)	2.23 ± 0.20	2.02 ± 0.22	4.40 ± 0.16	4.62 ± 0.15	**4.92 ± 0.12**
(16,2,10)	0.15 ± 0.06	0.20 ± 0.06	0.70 ± 0.12	0.45 ± 0.09	**1.35 ± 0.10**
(16,4,10)	0.60 ± 0.12	0.65 ± 0.13	2.02 ± 0.20	1.95 ± 0.13	**2.52 ± 0.16**
(16,6,10)	1.15 ± 0.17	1.15 ± 0.16	2.90 ± 0.20	3.40 ± 0.18	**3.58 ± 0.17**
(32,2,16)	0.17 ± 0.06	0.05 ± 0.03	**0.80 ± 0.10**	0.15 ± 0.06	0.72 ± 0.12
(32,4,16)	0.17 ± 0.06	0.20 ± 0.07	0.90 ± 0.13	0.60 ± 0.10	**1.43 ± 0.15**
(32,6,16)	0.82 ± 0.13	0.70 ± 0.14	1.65 ± 0.18	1.65 ± 0.16	**1.73 ± 0.15**

[13] Aijun Bai, Siddharth Srivastava, and Stuart J Russell. Markovian state and action abstractions for mdps via hierarchical mcts. In *IJCAI*, pages 3029–3039, 2016.

[14] Chaoqun Wang, Jiyu Cheng, Jiankun Wang, Xintong Li, and Max Q-H Meng. Efficient object search with belief road map using mobile robot. *IEEE Robotics and Automation Letters*, 3(4):3081–3088, 2018.

[15] Alejandro Sarmiento, Rafael Murrieta, and Seth 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)*, volume 2, pages 1153–1158. IEEE, 2003.

[16] Xinkun Nie, Lawson LS Wong, and Leslie Pack Kaelbling. Searching for physical objects in partially known environments. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5403–5410. IEEE, 2016.

[17] Mehmet R Dogar, Michael C Koval, Abhijeet Tallavajhula, and Siddhartha S Srinivasa. Object search by manipulation. *Autonomous Robots*, 36(1-2):153–167, 2014.

[18] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017.

[19] Tonci Novkovic, Remi Pautrat, Fadri Furrer, Michel Breyer, Roland Siegwart, and Juan Nieto. Object finding in cluttered scenes using interactive perception. *arXiv preprint arXiv:1911.07482*, 2019.

[20] Carlos Diuk, Andre Cohen, and Michael L Littman. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 240–247, 2008.

[21] Stéphane Ross, Joelle Pineau, Sébastien Paquet, and Brahim Chaib-Draa. Online planning algorithms for pomdps. *Journal of Artificial Intelligence Research*, 32: 663–704, 2008.

[22] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun

TABLE III: **Quality of Sensor Model** - Cumulative Reward; $(m, n, k) = (16, 2, 10)$
The rows are ordered in decreasing levels of noise.

(α, β)	Random	POMCP	POUCT	MA-POUCT	MR-POUCT
(10, 0.8)	-2718.20 ± 131.14	-2616.60 ± 138.28	124.95 ± 80.52	-327.82 ± 4.53	**211.64 ± 84.34**
(10, 0.3)	-2731.72 ± 119.43	-2761.15 ± 122.00	-27.07 ± 52.71	-334.78 ± 5.56	**201.33 ± 85.73**
(100, 0.8)	-2645.22 ± 129.94	-2658.35 ± 145.72	450.77 ± 99.36	-215.55 ± 49.33	**537.50 ± 114.14**
(100, 0.3)	-2548.60 ± 156.00	-2723.82 ± 128.86	482.88 ± 106.70	-264.98 ± 35.89	**559.46 ± 93.54**
(500, 0.8)	-2776.65 ± 113.66	-2690.70 ± 144.55	585.05 ± 119.80	13.11 ± 84.80	**794.97 ± 124.58**
(500, 0.3)	-2694.10 ± 126.93	-2765.28 ± 124.76	431.70 ± 101.98	-61.92 ± 72.00	**709.06 ± 103.21**
(10^3, 0.8)	-2946.15 ± 82.78	-2763.50 ± 133.98	456.27 ± 86.62	-17.40 ± 94.40	**1153.85 ± 131.90**
(10^3,0.3)	-2796.22 ± 112.62	-2688.05 ± 139.65	609.33 ± 100.52	-173.95 ± 59.36	**944.89 ± 103.48**
(10^4, 0.8)	-2791.95 ± 115.62	-2642.82 ± 131.43	712.48 ± 100.73	234.84 ± 94.61	**969.00 ± 120.47**
(10^4, 0.3)	-2948.70 ± 84.87	-2740.88 ± 133.48	711.15 ± 111.62	236.29 ± 103.20	**1056.07 ± 115.67**
(10^5, 0.8)	-2835.07 ± 106.70	-2643.18 ± 166.36	637.12 ± 112.34	97.97 ± 90.03	**954.59 ± 110.88**
(10^5, 0.3)	-2951.53 ± 82.95	-2759.85 ± 121.10	608.42 ± 113.27	20.91 ± 94.54	**944.53 ± 98.41**

TABLE IV: **Quality of Sensor Model** - Number of Detected Objects; $(m, n, k) = (16, 2, 10)$

(α, β)	Random	POMCP	POUCT	MA-POUCT	MR-POUCT
(10, 0.8)	0.20 ± 0.07	0.23 ± 0.07	0.30 ± 0.09	0.00 ± 0.00	**0.40 ± 0.09**
(10, 0.3)	0.17 ± 0.06	0.17 ± 0.06	0.15 ± 0.07	0.00 ± 0.00	**0.42 ± 0.09**
(100, 0.8)	0.23 ± 0.07	0.17 ± 0.07	0.60 ± 0.10	0.10 ± 0.05	**0.80 ± 0.10**
(100, 0.3)	0.28 ± 0.08	0.15 ± 0.06	0.62 ± 0.10	0.05 ± 0.03	**0.72 ± 0.09**
(500, 0.8)	0.15 ± 0.06	0.20 ± 0.07	0.75 ± 0.12	0.35 ± 0.09	**0.97 ± 0.11**
(500, 0.3)	0.20 ± 0.06	0.12 ± 0.05	0.62 ± 0.11	0.30 ± 0.09	**0.88 ± 0.10**
(10^3, 0.8)	0.07 ± 0.04	0.15 ± 0.07	0.68 ± 0.10	0.33 ± 0.10	**1.35 ± 0.12**
(10^3, 0.3)	0.17 ± 0.07	0.12 ± 0.05	0.80 ± 0.11	0.17 ± 0.07	**1.20 ± 0.10**
(10^4, 0.8)	0.15 ± 0.06	0.23 ± 0.07	0.90 ± 0.10	0.55 ± 0.09	**1.20 ± 0.11**
(10^4, 0.3)	0.07 ± 0.04	0.15 ± 0.06	0.90 ± 0.12	0.55 ± 0.10	**1.25 ± 0.12**
(10^5, 0.8)	0.12 ± 0.05	0.23 ± 0.08	0.80 ± 0.11	0.42 ± 0.09	**1.18 ± 0.10**
(10^5, 0.3)	0.07 ± 0.04	0.17 ± 0.06	0.78 ± 0.11	0.35 ± 0.09	**1.15 ± 0.10**

Lee. Despot: Online pomdp planning with regularization. In *Advances in neural information processing systems*, pages 1772–1780, 2013.

[23] Zachary N Sunberg and Mykel J Kochenderfer. Online algorithms for pomdps with continuous state, action, and observation spaces. In *Twenty-Eighth International Conference on Automated Planning and Scheduling*, 2018.

[24] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[25] Nan Ye, Adhiraj Somani, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. *Journal of Artificial Intelligence Research*, 58:231–266, 2017.

[26] Massoud Kaviany. *Heat transfer physics*. Cambridge University Press, 2014.

[27] Tirthankar Bandyopadhyay, Chong Zhuang Jie, David Hsu, Marcelo H Ang, Daniela Rus, and Emilio Frazzoli. Intention-aware pedestrian avoidance. In *Experimental Robotics*, pages 963–977. Springer, 2013.

[28] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Transactions on Graphics (TOG)*, 21(4):807–832, 2002.

[29] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[30] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

VIII. Appendix

A. Derivation of the Abstract Observation Model

In Section V-A, we described an abstract observation model that uses the same weighting mechanism as state abstraction.

Algorithm 3: POUCT $(\mathcal{G}, h_t, H) \rightarrow a$

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 $depth > H$ **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$(s', hao, depth + 1)$;
  $N(h) \leftarrow N(h) + 1$;
  $N(ha) \leftarrow N(ha) + 1$;
  $V(ha) \leftarrow V(ha) + \frac{R - V(ha)}{N(ha)}$;
  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$(s', hao, depth + 1)$;

Below, we show the derivation of this model.

$$\Pr(o_i^l|\hat{s}', a, h_t), \tag{23}$$
$$= \Pr(s_i^l, d(s_i^l)|\hat{s}', a, h_t), \tag{24}$$
$$= \sum_{s_i \in \phi^{-1}(s_i^l)} \Pr(s_i, d(s_i^l)|\hat{s}', a, h_t), \tag{25}$$
$$= \sum_{s_i \in \phi^{-1}(s_i^l)} \Pr(d(s_i^l)|s_i, \hat{s}', a, h_t) \Pr(s_i|\hat{s}', a, h_t). \tag{26}$$

Note that the state of object i is independent from other object states and the action a. Hence,

$$= \sum_{s_i \in \phi^{-1}(s_i^l)} \Pr(d(s_i^l)|s_i, \hat{s}', a, h_t) \Pr(s_i|s_i^l, h_t), \tag{27}$$

Now, we apply our definition of the weighting function based on the belief:

$$= \sum_{s_i \in \phi^{-1}(s_i^l)} \Pr(d(s_i^l)|s_i, \hat{s}', a, h_t) w(s_i). \tag{28}$$

B. POUCT Algorithm

We slightly modified the POUCT (Partially Observable UCT) algorithm presented in Silver and Veness [11] in order to match the input and output of Algorithm 2. This algorithm is described in Algorithm 3. Note that this is the same algorithm as POMCP [11] without particle belief representation.