Multi-Object Search using Object-Oriented POMDPs

Arthur Wandzel, Yoonseon Oh, Michael Fishman, Nishanth Kumar, Lawson L.S. Wong, and Stefanie Tellex

***Abstract*— A core capability of robots is to reason about multiple objects under uncertainty. Partially Observable Markov Decision Processes (POMDPs) provide a means of reasoning under uncertainty for sequential decision making, but are computationally intractable in large domains. In this paper, we propose Object-Oriented POMDPs (OO-POMDPs), which represent the state and observation spaces in terms of classes and objects. The structure afforded by OO-POMDPs support a factorization of the agent's belief into independent object distributions, which enables the size of the belief to scale linearly versus exponentially in the number of objects. We formulate a novel Multi-Object Search (MOS) task as an OO-POMDP for mobile robotics domains in which the agent must find the locations of multiple objects. Our solution exploits the structure of OO-POMDPs by featuring human language to selectively update the belief at task onset. Using this structure, we develop a new algorithm for efficiently solving OO-POMDPs: Object-Oriented Partially Observable Monte-Carlo Planning (OO-POMCP). We show that OO-POMCP with grounded language commands is sufficient for solving challenging MOS tasks both in simulation and on a physical mobile robot.**

I. Introduction

A core capability of robots is to reason about multiple objects under uncertainty. A rescue robot, for example, may be asked to find all human survivors in a disaster site. A service robot, on the other hand, may be asked to find all toys in the living room to clean up a house. In this work, we introduce a novel multi-object search (MOS) task, in which the objective is to find the locations of a number of objects with uncertainty over all possible object locations. One crucial challenge for an MOS task is supporting efficient planning while scaling with the number of objects.

In real-world robotic tasks, the robot may operate without full knowledge of the environment. Partially observable Markov decision processes (POMDPs) provide a means for sequential decision making under uncertainty [1]. POMDPs are an appealing framework for modeling a robot because they capture the observe, predict, and act tasks a robot must perform when interacting within a partially observable world [2]. However, POMDPs are computationally intractable for planning in large domains [3]. A POMDP planner reasons about current and future *beliefs*, which are probability distributions over all possible states. One source of intractability is that the belief space has dimensionality equal to the number of possible states, termed the *curse of dimensionality* [4]. In an MOS task, the belief grows exponentially with the number of objects if naïvely represented.

[1]The authors are with the Brown University Department of Computer Science; Lawson L.S. Wong is with Northeastern University Khoury College of Computer Sciences. Primary email contact: arthur_wandzel@brown.edu.

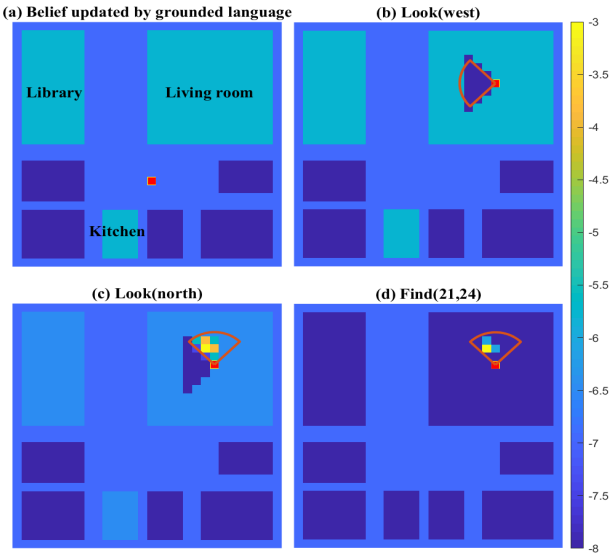

Fig. 1: The agent (red dot) must find the locations of one or more objects (yellow). (a) The belief after a language observation "Find the mugs in the library, living room, or kitchen." (b) and (c) show the belief after a sensor observations via `Look` actions. (d) The agent's belief converges to the true object location after a sequence of actions. (Probabilities in log scale.)

In this work, we formulate the MOS task as an Object-Oriented POMDP (OO-POMDP), a generalization of prior work on OO-MDPs [5]. An OO-POMDP, introduced in this paper, supports convenient representations of robotic tasks that require reasoning about multiple objects under uncertainty. The state, transition, and observation spaces are represented in an OO-POMDP in terms of *classes* and *objects*. With this additional structure, we can factor the belief into object distributions, using an additional assumption that objects are independent. This allows the belief space size to scale linearly (versus exponentially) in the number of objects.

To solve the MOS task we propose an OO-POMDP model called the Multi-Object Search OO-POMDP (MOS OO-POMDP). We exploit the structure of OO-POMDPs to develop an object-scalable planning algorithm called Object-Oriented Partially Observable Monte-Carlo Planning (OO-POMCP), which extends POMCP, a well-known online planning algorithm for large domains [6].

We also explore the use of language as input for updating priors on object uncertainty. The belief for each object distribution is selectively updated at task onset by an initial grounded language statement. A human operator, for example, can express the statement, "Find the mugs in the library, living room, or kitchen," which directly modifies the robot's belief to express uncertainty over locations in the referenced rooms for a referenced class of objects (Figure 1(a)). The structure of OO-POMDPs make it simple

to express observation models involving objects, as in this example. Additionally, belief updates can be easily restricted to involve only the relevant objects. Our implementation of the MOS OO-POMDP also uses a topological map generated by Rapidly-exploring Random Trees (RRT) [7] and a fan-shaped sensor with limited range, which accounts for sensor errors. Figure 1(b-d) show an actual trace of actions inferred by our model when searching for objects.

We validate the performance of our approach from experiments conducted in both simulated and real-world environments. In a complex MOS task with 8 objects, ≈ 12 actions, $\approx 10^9$ observations, $\approx 10^{27}$ states, and large-scale belief spaces with $\approx 10^{21}$ dimensions, OO-POMCP achieves high performance in planning, while considering a number of types of language observations categorized as *misinformed*, *no information*, *ambiguous* and *informed*.

II. Related Work

Object search has received considerable attention in the service robotics community. The problem originated as an extension of active visual search [8, 9, 10, 11] and since then has focused on exploiting additional structure so as to scale to realistic environments [12], which includes object-to-object co-occurrence [13, 14, 15], object-to-scene co-occurrence [16, 17, 18, 19, 20], relational affordances [21], and scene ontologies [22]. All such works, however, have focused on single object search. Nie et al. [23] has investigated object search in single-room cluttered environments with an unknown number of objects, but does not focus on scaling to large domains. The multi-object search task proposed in this paper represents a natural generalization of current object search literature and to date has not been investigated.

Use of language for directly reducing uncertainty over spatial locations is a novel contribution in object search. In the area of collaborative robotics, natural language commands have been used for spatial reasoning in robot navigation [24, 25, 26] without modeling environment uncertainty. Whitney et al. [27] looks at reducing uncertainty in a POMDP with observations from human speech for a tabletop item-delivery task. Our work similarly looks at language as a source of information, however, in the context of object search. Walter et al. [28] described an approach for combining language and physical sensor observations to map an environment; our approach assumes a known map but uses language to induce posteriors over object locations.

Planning with large belief spaces in POMDPs has been addressed by estimating the belief via sampling [6, 29, 30], compressing the belief space into a more tractable lower-dimensional subspace [31], or defining the belief only over components of the state relevant for decision-making [32].

OO-POMDPs extend Object-Oriented Markov Decision Processes (OO-MDPs) [5] to POMDPs by factorizing both state *and* observation spaces in terms of objects. OO-MDPs use object state abstractions to enable the agent to reason about objects and relations between objects. Work that focuses on relational representations in reinforcement learning include Relational-MDPs [33] and Relational-POMDPs [34].

A POMDP [1] is defined as a 7-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, \Omega, O \rangle$. The state $s \in \mathcal{S}$ contains all task-relevant information for decision-making. An agent selects actions $a \in \mathcal{A}$ resulting in a next state s' defined by the *transition function*: $\mathcal{T}(s, a, s') = p(s'|s, a)$. Execution of an action yields an observation $z \in \Omega$ derived by the *observation function* $O(s, a, z) = p(z|s, a)$. The reward of each action is defined by the *reward function* $\mathcal{R}$ with discount factor γ. The agent maintains a *belief*, b, which is a probability distribution over possible states in $\mathcal{S}$ that the agent could be in. Taking an action a and receiving an observation z results in a new belief b' for a particular next state via a belief update:

$$b'(s') = \eta\, O(s', a, z) \sum_{s \in \mathcal{S}} \mathcal{T}(s, a, s')\, b(s)\ , \qquad (1)$$

where η is the normalizing constant.

III. Technical Approach

In this paper, we develop an OO-POMDP formulation of the MOS task. The task objective is for an agent to find the (x, y) location of n object instances, which are situated among a set of locations in a 2-D environment. We assume that the layout of the environment is known. Each object belongs to some class $c \in \mathcal{C}$. The OO-POMDP framework provides structure for factorizing the belief for efficient inference. We introduce two contributions that exploit this factorization:

1) Observations from language commands
2) OO-POMCP algorithm

In later sections, we demonstrate the selectivity and efficiency of object-specific belief updates, respectively, through language commands and the OO-POMCP algorithm.

A. MOS OO-POMDP

We formulate the MOS task as an OO-POMDP which supports convenient representation of robotic tasks that require reasoning about multiple objects under uncertainty. As with a conventional object-oriented programming language, there are two levels of organization in an OO-POMDP: classes and objects. An OO-POMDP is represented as a 10-tuple $\langle \mathcal{C}, \textsc{Att}(c), \textsc{Dom}(a), Obj, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, \Omega, O \rangle$ [5].

In an OO-POMDP, the state space, $\mathcal{S}$ and observation space, Ω, are factored into a set of objects $Obj = \{obj_1, \ldots, obj_n\}$. Each object obj_i is an instance of a particular class, $c \in \mathcal{C}$ that consists of a set of class-specific attributes $Att(c) = \{c_{a_1}, \ldots, c_{a_m}\}$. Each attribute $a \in Att(c)$ has a domain of possible values $Dom(a)$. The dual-factorization of $\mathcal{S}$ and Ω allows the observation function to exploit shared structure so as to define observations grounded in the state with varying degrees of specificity: over a class of objects, a single object, or an object's attribute.

The state of the MOS OO-POMDP is decomposed into the state of the robot s_r and the state of objects s_{obj}. The state of n objects is defined as $s = \bigcup_{i=1}^{n} s_i$, where s_i encodes the cell containing the object as a one-hot vector.

We make use of three maps: an occupancy-grid map $\mathcal{M}_o$, a semantic map $\mathcal{M}_s$, and a topological map $\mathcal{M}_t$. We assume

that $\mathcal{M}_o$ and $\mathcal{M}_s$ are given. $\mathcal{M}_o$ is a $m \times m$ grid that marks each grid cell as empty or occupied by fixed, static structures such as walls, while $\mathcal{M}_s$ maps from grid cells to a single room in a set of rooms $\mathcal{R}$. $\mathcal{M}_t$ provides a sparse graph representation of $\mathcal{M}_o$ for navigation. We reduce the number of cells considered for navigation using $\mathcal{M}_t$ by applying the RRT algorithm on $\mathcal{M}_o$, as inspired by Wang et al. [19]. In the center of each room, we run RRT for a fixed time interval, which constructs a star-like graph. Each node is a sparsely sampled cell in $\mathcal{M}_o$ and each edge is a path between nodes. We reject an edge that is shorter than D for sparsity, D being the depth of the fan-shaped sensor region. An off-the-shelf planner computes a navigation policy per edge for traversing between nodes. All room centers are connected to each other by additional edges.

We define the OO-POMDP components as follows:

- $\mathcal{C}$: the object classes like $Robot$, Mug, and Key.
- $\textsc{Att}(c)$: all classes contain the attribute of the (X, Y)-location.
- $\textsc{Dom}(a)$: the range of the (X, Y)-location is the set of all possible locations, L, consisting of all cells in rooms.
- Obj: the set of objects. Obj excludes the agent, is finite (with known cardinality n), and fixed in advanced.
- $\mathcal{A}$: `Look(`d`)` projects a fan-shaped sensing region of depth D in one of the four cardinal directions d; `Find(`l`)` marks a single location l as containing an object; `Move(`e`)` moves an agent to a connected node via edge $e \in \mathcal{M}_t$; `Move(`r`)` moves an agent to an adjoining room $r \in \mathcal{M}_s$.
- $\mathcal{T}$: All actions are assumed to be deterministic.
- $\mathcal{R}$: the agent receives for each object $+1000$ for `Find(`l`)` if l corresponds to an object's true location and -1000 if incorrect. All other actions get -10. `Move` actions receive an extra cost of the Euclidean distance from the start to end location. The experiment task ends after n `Find` actions.
- O and Ω are defined in a later section. They consist of language observations (Section III-C) and direct object observations from a fan-shaped sensor (Section III-D).

B. Beliefs over Multiple Objects

One challenge for OO-POMDPs is to efficiently manage a belief distribution that scales with multiple objects. In the MOS OO-POMDP, uncertainty is defined over L possible object locations. The state space grows exponentially with the number of objects: $|\mathcal{S}| = \prod_{i=1}^{n} |\mathcal{S}_{obj_i}| = |L|^n$, where $\mathcal{S}_{obj_i}$ denotes the state space of object obj_i. A POMDP planner must reason over beliefs with dimension equal to the size of the state space.

In this work, we tackle the problem of representing the belief over multiple objects by exploring one possible independence assumption. We assume that objects are independent, thereby allowing the belief b to be factored into n components, each defined over the state space of a particular object obj_i: $b = \prod_{i=1}^{n} b_i$. Crucially, this independence assumption enables the size of the belief to scale *linearly* in the number of objects. While the dimension of the unfactored belief is $|L|^n$, the dimension of the factored belief is $n|L|$. A core technical contribution of OO-POMDPs is to provide structure for defining object-specific observations z_i, restricted to a component of the state $s_i \in \mathcal{S}_{obj_i}$, to support such a factorization. The belief for object i is updated by:

$$b_i'(s_i) = \eta\, p(z_i|s_i)\, b_i(s_i)\,, \tag{2}$$

where z_i denotes an observation for object i and η is a normalization factor. The observation z_i is an observation by the sensor z_i^s or the by the language command z_i^l. While z_i^s is received throughout agent planning, an initial language command yields z_i^l only once at the beginning of each task.

C. Observations from Language Commands

Observations in an OO-POMDP can reference a class, object, or attribute while directly mapping to the state. In this section, we look at selectively updating the belief via object-specific observations afforded by language observations.

We consider the scenario in which a human operator tells the robot: "The mugs are in the kitchen, and the books are in the library". We assume a keyword language model that maps words in the language command into a set of classes and room labels. A language command can be represented by a set of pairs (c_i, R_i), where $c_i \in \mathcal{C}$ is a class of objects and R_i is a set of cells in the referenced rooms.

A statement (c_i, R_i) can be encoded as an observation by $z_i^l \in [0,1]^{m^2}$ where each element z_{ij}^l represents the probability that object i occupies the cell j. Error as a result of the robot misinterpreting the human command or the human commanding the robot to the wrong room is captured by the term ψ below. Formally:

$$z_{ij}^l = \begin{cases} \frac{(1-\psi)}{A} & \text{if the cell } j \in R_i, \\ \frac{\psi}{m^2 - A} & \text{otherwise} \end{cases} \tag{3}$$

where A is the number of cells in the referenced rooms and ψ is a small value between 0 and 1. For objects that are referenced but without rooms (i.e. $R_i = \emptyset$), $z_{ij}^l = 1/m^2$ for all j. Given that object i is in location s_i, the probability of z_i^l is simply their dot product: $p(z_i^l|s_i) = s_i \cdot z_i^l$. Thus the agent's belief for i can be updated with z_i^l alone.

Whereas the initial belief is assumed to be a uniform distribution, a language observation updates each b_i object distribution, resulting in a modified belief that is sensitive to the knowledge of the human operator. A human operator may possess general knowledge (e.g., that mugs tend to be in the kitchen) as well as domain-specific knowledge (e.g., in my house mugs also tend to be in the bathroom).

D. Fan-Shaped Sensor Model

In this section, we propose the observation model for the sensor: $p(z^s|s)$. The sensor is modeled as a discretized fan-shaped sensing region, $\mathcal{V}$, with a limited field of view and depth D. Observations from the sensing region, z^s, consist of n object-specific observations $z_i^s \in \mathcal{V} \cup \{\texttt{NULL}\}$. If object i is not detected by the sensor, $z_i^s = \texttt{NULL}$. Otherwise, z_i^s is the location where object i is detected in $\mathcal{V}$.

Each z_i^s is obtained from one of three mutually exclusive and exhaustive events, A_i, B_i, and C_i. Let the event A_i be $z_i^s \in \mathcal{V}$ and z_i^s is from the object i. Let the event B_i be

Algorithm 1 OO-POMCP

$b \leftarrow$ Prior
$s \leftarrow$ InitialState
for $i = 0$ to ACTIONS **do**
  $T \leftarrow \{\}$
  for $j = 0$ to SIMULATIONS **do**
    $\hat{s} \leftarrow$SAMPLE(b)
    SIMULATE($\hat{s}, \{\}, 0$)
  $a \leftarrow \arg\max_a V(ha)$
  $(s', z, r) \leftarrow \mathcal{E}(s, a)$
  $b \leftarrow$ UPDATE(b, a, z)
  $s \leftarrow s'$
  if TerminationCondition() **then** break

function UPDATE(b, a, z)
  for $o \in Obj$ **do**
    $b'_o(s') \leftarrow \eta O(z|s', a) \sum_{s \in \mathcal{S}} T(s'|a, s) b_o(s)$
  return $\prod b'_o(s')$

function SAMPLE(b)
  for $o \in Obj$ **do**
    $\hat{s}_o \sim b_o$
  return $\cup \hat{s}_o$

function SIMULATE($s, h, depth$)
  if $\gamma^{depth} < \epsilon$ **then** return 0
  if $h \notin T$ **then**
    for all $a \in \mathcal{A}$ **do**
      $T(ha) \leftarrow \langle N_{init}(ha), V_{init}(ha) \rangle$
    return ROLLOUT($s, h, depth$)
  $a \leftarrow$ selectMaxAction()
  $(s', z, r) \sim \mathcal{G}(s, a)$
  $R \leftarrow r + \gamma \cdot$ SIMULATE($s', hao, depth+1$)
  ~~$B(h) \leftarrow B(h) \cup \{s\}$~~
  $T(ha) \leftarrow \langle N(ha) + 1, V(ha) + \frac{R - V(ha)}{N(ha)} \rangle$
  return R

POMCP functions include SIMULATE and ROLLOUT. OO-POMCP additions are SAMPLE and UPDATE. Crossed-out text denotes removal from POMCP.

$z_i^s \in \mathcal{V}$ and z_i^s is from other sources. Let the event C_i be $z_i^s = \texttt{NULL}$. Thus we can decompose $p(z_i^s|s)$ as follows:

$$p(z_i^s|s) = \sum_{e_i \in \{A_i, B_i, C_i\}} p(z_i^s|e_i, s) p(e_i|s)$$

If event A occurs, the observation is normally distributed with μ_i being the true object i position: $p(z_i^s|A_i, s) = \eta' \cdot f(z_i^s|\mu_i, \Sigma_i)$, for $z_i^s \in \mathcal{V}$. The covariance matrix is defined by $\Sigma_i = \sigma^2 \mathbf{I}^{2\times 2}$ and η' is the normalization factor. If event B occurs, we assume that the false positive detection could have come equally likely from any location in $\mathcal{V}$: $p(z_i^s|B_i, s) = \frac{1}{|\mathcal{V}|}$, if $z_i^s \in \mathcal{V}$. If event C occurs, z_i^s should be always NULL and $p(z_i^s = \texttt{NULL}|C_i, s) = 1$. Note that it is impossible for A and B to get the observation $z_i^s = \texttt{NULL}$.

We define the probability of the events as $p(A_i|s) = \alpha_i$, $p(B_i, |s) = \beta_i$, $p(C_i|s) = \gamma_i$, where $\alpha_i + \beta_i + \gamma_i = 1$. These parameters are defined differently depending on whether the object i is in the sensing region or not:

$$(\alpha_i, \beta_i, \gamma_i) = \begin{cases} (\epsilon, \frac{1-\epsilon}{2}, \frac{1-\epsilon}{2}) & \text{if the object } i \text{ is in } \mathcal{V} \\ (\frac{1-\epsilon}{2}, \frac{1-\epsilon}{2}, \epsilon) & \text{if the object } i \text{ is not in } \mathcal{V}, \end{cases}$$

where ϵ acts to define the overall accuracy of the sensor. We can model a perfect detector with $\sigma = 0, \epsilon = 1$.

E. OO-POMCP Algorithm

Partially Observable Monte-Carlo Planning (POMCP) is a well-known online POMDP planning algorithm that has demonstrated success in scaling to large POMDPs [6]. POMCP applies Monte-Carlo Tree Search (MCTS) [35] to POMDPs to estimate both the Q-value for selecting a real action and the next belief. In Algorithm 1, the two functions that are part of POMCP are SIMULATE and ROLLOUT (ROLLOUT is omitted for space; see [6] for details).

OO-POMCP differs from POMCP in its representation of the belief as a collection of independent object distributions, b_0, as manifested in the functions SAMPLE and UPDATE in Algorithm 1. More importantly, OO-POMCP does not estimate the next belief b' while estimating the Q-values but performs a full belief update, reflected in UPDATE, by exactly updating each of the more manageable object components. We next go over the OO-POMCP algorithm in more detail and then discuss its improvements.

A forward search tree T is constructed each decision cycle by iteratively sampling particles, $\hat{s}$, from the current belief for SIMULATE. Each node in the tree, T, represents a particular sequence of action and observations called a history $h = \{a_1, z_1, ..., a_t, z_t\}$. $T(h)$ contains a value estimate of h calculated by the sum of discounted rewards encountered in h, $V(h)$, divided by the number of times h was visited $N(h)$: $V(h) = \frac{\sum R_i}{N(h)}, R_i = \sum_{k=t}^{\infty} \gamma^k r_k$.

Histories in the tree are sampled by recursively calling the function SIMULATE with a black-box simulator, $\mathcal{G}(s, a)$. If a history has not been encountered, then it is initialized and added to the tree; otherwise each sampled history adds the discounted reward and increments its count. After a fixed number of simulations, the maximum estimated Q-value $V(ha)$ is selected from the tree to execute a real action in the environment $\mathcal{E}(s, a)$, yielding a real observation to update the belief to give b', and ending the decision cycle.

POMCP estimates the next belief b' while sampling histories: $B(h) \leftarrow B(h) \cup \{s\}$. The belief $B(h)$ is a multiset of particles (e.g. $\{s_1, s_2, s_1...\}$), which implicitly captures the frequency of a given particle. $B(h)$ is stored for each next action-observation pair. After an action is chosen and executed, the belief update occurs by setting b' directly to be the $B(h)$ of the action-observation pair that corresponds to the true executed action and resulting observation. OO-POMCP, in contrast, *separates* belief estimation and Q-value estimation into two separate processes. Like POMCP, OO-POMCP samples particles from each object distribution in SAMPLE to perform SIMULATE and estimate Q-values. However, OO-POMCP then performs an *explicit, exact* belief update (outside of constructing T) per object distribution b_o in UPDATE. This is possible because it is computationally tractable to represent each b_o in the factored belief.

One shortcoming of POMCP is failing to sufficiently approximate a large b'. In MOS the joint belief grows exponentially as the number of objects increases. Further-

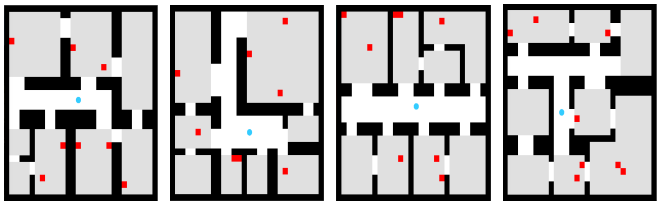

Fig. 2: Simulated 30×30 domains with agent (blue circle), objects (red squares), and walls (black squares). Light grey marks uncertain locations.

more, the difficulty of sampling enough particles in POMCP is compounded because many particles are rejected. The probability of keeping a particle consistent with the correct action-observation pair is $\lambda = \frac{1}{|\mathcal{A}||\Omega|}$ when assuming a uniform distribution over action-observation pairs, but for rare action-observation pairs λ is strictly less.

Small values of λ result in *particle decay*: gradually losing particles when recomputing b' over successive planning cycles [6]. Particle decay is harmful when b' does not contain the true environment state, resulting in suboptimal behavior, and fatal when no samples exist to represent the next belief (because all were rejected after encountering a rare observation), resulting in random behavior. One partial solution is to resample K particles per planning cycle, known as *particle reinvigoration*; however, this cannot recover from cases where the true environment state is no longer in b'.

An explicit belief update in OO-POMCP, however, guarantees that b' is computed according to the true action-observation pair, thereby eliminating particle decay. This allows OO-POMCP to plan in a robust fashion. A factored belief, furthermore, allows for greater sample efficiency because $|\mathcal{S}_{obj_i}|^n$ particles can be represented by $n|\mathcal{S}_{obj_i}|$ particles within a factored belief for n objects. OO-POMCP extends POMCP to support robust planning and sample efficiency while scaling to domains with many objects.

IV. Evaluation

A. Results in Simulation

In the simulation experiments, we test our approach on an MOS task in four domains. Each domain consists of a 30×30 grid composed of 8 rooms with hallways and doorways (see Figure 2). Objects can exist in any room with about 500 possible object locations in each simulated domain. The complexity of each domain, when considering 8 objects, is approximately 12 actions, 10^9 observations, 10^{27} states, and a belief space of $500^8 \approx 10^{21}$ dimensions, representing a challenging POMDP.

The performance of OO-POMCP is contrasted against six methods: *Known* gives an upper bound on performance by solving the task with OO-POMCP and known object locations; *Random* executes a random policy giving a lower bound on performance; *Iterative OO-POMCP* decomposes the MOS task into n single-object POMDPs, each solved by OO-POMCP. This condition validates how much MOS is improved by shared information across objects. When the belief distribution is defined over all objects, as in OO-POMCP, observations reduce uncertainty across all objects. An agent must search a location once for n objects versus n times (once per each object). Finally, *.95/.5* and *.90/1.0* refer to running OO-POMCP with sensor noise respectively defined by the accuracy ϵ and standard deviation σ used to parameterize Σ. All other methods had sensor model settings of 1.0 and 0 respectively.

We conduct three experiments that vary the number of objects, number of simulations (number of calls to `SIMULATE`), and informativeness of the language-based prior. Figure 3 illustrates the cumulative reward measured for each experiment. Each data point is the mean of 100 trials, 25 trials of each domain, with randomized object locations in each trial. A trial terminates when executing 200 actions or executing n `FIND` actions. The agent searches for objects with uniform uncertainty over all possible object locations in the first and second experiment.

In the first experiment, we investigate how the algorithms scale with the number of objects with 10^4 simulations. *OO-POMCP* outperforms *Iterative OO-POMCP* and *POMCP* even with significant sensor noise. This is because *OO-POMCP* maintains a belief distribution over all objects, which exploits shared information (vs. *Iterative OO-POMCP*) while avoiding particle decay (vs. *POMCP*). *Iterative OO-POMCP*'s performance does not improve because the algorithm cannot exploit shared observational information across objects, which results in re-searching empty object locations. *POMCP*'s performance regresses to *Random* because it cannot efficiently estimate large beliefs even with 10^3 additional states sampled per planning cycle for particle reinvigoration.

In the second experiment, we investigate the sample efficiency of the algorithms by modifying the number of simulations with 4 objects. All algorithms improve with more simulations. The rate at which *OO-POMCP* and its variants improve, however, reflects that a smaller number of simulations are necessary to estimate good Q-values. We conjecture that this is related to the number of *relevant* samples s that `SIMULATE` receives during planning. *POMCP* often results in an inaccurate set of particles for the current belief b due to *particle decay*. A simulation is irrelevant if *POMCP* executes `SIMULATE` on a state that is incorrectly believed to be likely, or does not execute `SIMULATE` on a state that is incorrectly believed to be unlikely. *OO-POMCP* avoids this issue by using *exact* factored belief updates, hence all calls to `SIMULATE` begin from relevant samples of the current state. This simulation-sample efficiency enables *OO-POMCP* to scale to large domains.

In the third experiment, we investigate the effect of the informativeness of the language observation type on performance with 10^4 simulations and 4 objects. A language command provides an observation that updates the agent's initial uniform belief. Language observations are categorized as *misinformed*, *no information*, *ambiguous*, and *informed*. *Misinformed* is the case where the language command does not reflect the ground truth; *no information* omits room information entirely; *ambiguous* references multiple rooms; *informed* references one room. In this experiment, *misinformed* and *ambiguous* references 3 rooms and ψ (error rate)

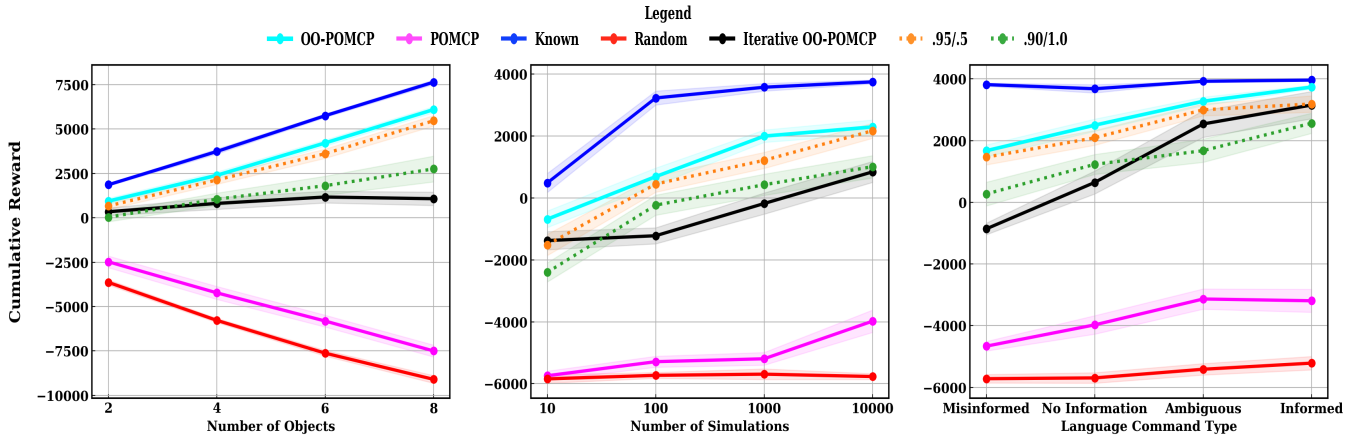

Fig. 3: Algorithm performance for three experiments. The vertical axis measures cumulative reward. Each point is the mean of 100 trials. The cumulative reward roughly corresponds to the number of objects found (+1000 for correctly / -1000 for incorrectly choosing an object).

Fig. 4: Experiment with a language command "Find mugs in the kitchen and a book in the library". The first image shows an occupancy grid map and semantic map of the environment. Red and blue circles indicate mugs and book respectively, and the blue box indicates the robot MOVO. The robot directly moves to the kitchen, looks around, and finds the first mug at 02:59. The inset images (red box) is the robot's view. The second mug is found in the kitchen at 03:27, after 6 actions. The robot next moves to the library, looks around again, and then finds the book at 05:14 (3 additional actions).

was set to .05. All algorithms consistently improve as the informativeness of the language observation increases, except *POMCP*. This is because in the *informed* case all objects aggregate in one room. This results in rare observations corresponding to a specific configuration of multiple detected objects. Rare observations cause *POMCP* to not receive enough particles to estimate the next belief.

B. Real World Results

OO-POMCP was evaluated using a Kinova MOVO mobile robot. The robot searches for objects in a $13.33\,\mathrm{m} \times 6.50\,\mathrm{m}$ physical environment composed of a five rooms. In Figure 4, the occupancy grid map and the semantic map of the environment are shown. The workspace of the robot is decomposed into a 20×20 grid map. The semantic map consists of `Kitchen`, `Library`, `LivingRoom`, `Storage`, and `RoboticsLab`. If the robot finds an object, it moves its torso up and down.

Figure 4 shows the sequence of actions of the robot. The language command is "Find a book in the library and mugs in the kitchen," which is processed by Google's Cloud Speech-to-Text API. The robot detects the book and mugs using AR tags with its Microsoft Kinect 2 sensor. Since the robot receives selective information from the human, it does not have to visit all rooms; instead, it searches the kitchen and the library directly. Over 5 trials, the robot took 11.20 actions on average to find all objects, with an average reward of 2907 and average execution time of 364s (including preprocessing time for building the topological graph). These results demonstrate that our approach scales to a mobile robot in real-world environments. An example of the robot's execution can be seen in our video attachment.

V. Conclusion and Future Work

In this paper, we defined an OO-POMDP formulation of a novel multi-object search (MOS) task. Our solution enables searching for multiple objects in a large space by factorizing the agent's belief in terms of objects. By leveraging a key object-independence assumption, we demonstrated that OO-POMCP can efficiently maintain object-specific factored beliefs, and observations from both our visual sensor and natural language support factored belief updates as well. As a result, our method outperforms POMCP, achieving high performance on large MOS tasks with approximately 12 actions, 10^9 observations, 10^{27} states, and large-scale belief spaces with 10^{21} dimensions. We show that our solution scales to a physical mobile robot using a LIDAR for localization and a Microsoft Kinect 2 for object detection, enabling it to efficiently find objects, incorporate information from language, and operate in real-world environments.

Our solution uses a keyword language model for generating language commands as a proof of concept. In future work we plan to develop a realistic model trained on unrestricted natural language commands. The observation from language commands, furthermore, currently occur only at the onset of the task. In future work, we aim to create an observation model with the capacity to receive language commands throughout the duration of the task, enabling collaborative human-in-the-loop multi-object search.

Acknowledgment

This work is supported by the National Science Foundation (IIS-1637614 and IIS-1652561), the National Aeronautics and Space Administration (NNX16AR61G), DARPA (W911NF-10-2-0016 and D15AP00102), and with fellowship support from the Croucher Foundation.

REFERENCES

[1] L. P. Kaelbling, M. L. Littman, and A. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, pp. 99–134, 1998.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[3] O. Madani, S. Hanks, and A. Condon, "On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems," in *National Conference on Artificial Intelligence (AAAI)*, 1999.

[4] J. Pineau, G. Gordon, and S. Thrun, "Anytime point-based approximations for large POMDPs," *Journal of Artificial Intelligence Research*, vol. 27, no. 335–380, 2006.

[5] C. Diuk, A. Cohen, and M. L. Littman, "An object-oriented representation for efficient reinforcement learning," *International Conference on Machine Learning (ICML)*, 2008.

[6] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Neural Information Processing Systems (NIPS)*, 2010.

[7] S. M. LaValle and James J. Kuffner, Jr., "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[8] S. Chen, Y. Li, and N. M. Kwok, "Active vision in robotic systems: A survey of recent developments," *International Journal of Robotics Research*, vol. 30, no. 11, pp. 1343–1377, 2011.

[9] K. Sjö, D. G. López, C. Paul, P. Jensfelt, and D. Kragic, "Object search and localization for an indoor mobile robot," *Journal of Computing and Information Technology*, vol. 17, no. 1, pp. 67–80, 2009.

[10] F. Saidi, O. Stasse, K. Yokoi, and F. Kanehiro, "Online object search with a humanoid robot." in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[11] K. Shubina and J. K. Tsotsos, "Visual search for an object in a 3D environment using a mobile robot," *Computer Vision and Image Understanding*, vol. 114, no. 5, pp. 535–547, 2010.

[12] L. Kunze, M. Beetz, M. Saito, H. Azuma, K. Okada, and M. Inaba, "Searching objects in large-scale indoor environments: A decision-theoretic approach," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[13] 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 (ICRA)*, 2009.

[14] J. Elfring, S. Jansen, R. van de Molengraft, and M. Steinbuch, "Active object search exploiting probabilistic object–object relations," in *RoboCup 2013: Robot World Cup XVII*. Springer, 2013, pp. 13–24.

[15] L. L. S. Wong, L. P. Kaelbling, and T. Lozano-Pérez, "Manipulation-based active search for occluded objects," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[16] A. Aydemir, A. Pronobis, M. Göbelbecker, and P. Jensfelt, "Active visual object search in unknown environments using uncertain semantics," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 986–1002, 2013.

[17] M. Lorbach, S. Höfer, and O. Brock, "Prior-assisted propagation of spatial information for object search," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[18] L. Kunze, K. K. Doreswamy, and N. Hawes, "Using qualitative spatial relations for indirect object search," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[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] D. Joho and W. Burgard, "Searching for objects: Combining multiple cues to object locations using a maximum entropy model," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[21] B. Moldovan and L. De Raedt, "Occluded object search by relational affordances," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[22] Z. Saigol, B. Ridder, M. Wang, R. Dearden, M. Fox, N. Hawes, D. M. Lane, and D. Long, "Efficient search for known objects in unknown environments using autonomous indoor robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Workshop on Task Planning for Intelligent Robots in Service and Manufacturing*, 2015.

[23] X. Nie, L. L. S. Wong, and L. P. Kaelbling, "Searching for physical objects in partially known environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[24] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy, "Understanding natural language commands for robotic navigation and mobile manipulation," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2011.

[25] D. Arumugam, S. Karamcheti, N. Gopalan, L. L. S. Wong, and S. Tellex, "Accurately and efficiently interpreting human-robot instructions of varying granularities," in *Robotics: Science and Systems (RSS)*, 2017.

[26] N. Gopalan, D. Arumugam, L. L. S. Wong, and S. Tellex, "Sequence-to-sequence language grounding of non-Markovian task specifications," in *Robotics: Science and Systems (RSS)*, 2018.

[27] D. Whitney, E. Rosen, J. MacGlashan, L. L. S. Wong, and S. Tellex, "Reducing errors in object-fetching interactions through social feedback," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[28] M. R. Walter, S. Hemachandra, B. Homberg, S. Tellex, and S. Teller, "A framework for learning semantic maps from grounded natural language descriptions," *International Journal of Robotics Research*, vol. 33, no. 9, pp. 1167–1190, 2014.

[29] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces." in *Robotics: Science and systems (RSS)*, 2008.

[30] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP planning with regularization," in *Neural Information Processing Systems (NIPS)*, 2013.

[31] N. Roy, G. Gordon, and S. Thrun, "Finding approximate POMDP solutions through belief compression," *Journal of Artificial Intelligence Research*, vol. 23, pp. 1–40, 2005.

[32] S. C. Ong, S. W. Png, H. David, and W. S. Lee, "Planning under uncertainity for robotic tasks with mixed observability," *International Journal of Robotics Research*, vol. 29, no. 8, pp. 1053–1068, 2010.

[33] C. Guestrin, D. Koller, C. Gearhart, and N. Kanodia, "Generalizing plans to new environments in relational MDPs," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[34] C. Wang and R. Khardon, "Relational partially observable MDPs," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

[35] R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo tree search," in *International Conference on Computers and Games*, 2006.