Towards Language-Conditioned Observation Models for Visual Object Search

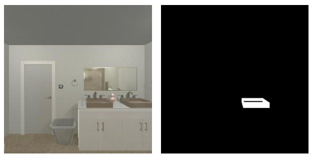
Thao Nguyen, Vladislav Hrosinkov, Eric Rosen, Stefanie Tellex
{thaonguyen, vladislav_hrosinkov, eric_rosen}@brown.edu, stefie10@cs.brown.edu

***Abstract*—Object search is a challenging task because when a human communicates complex language descriptions (*e.g.,* "the red cup on the table"), the robot must perceive the described object as well as handle noisy observations to plan search actions effectively. Previous works map language descriptions to a set of fixed object detectors with predetermined noise models, but these approaches are challenging to scale as new detectors need to be made for each object and complex object attributes may affect the sensor noise of the associated observation model. In this work, we bridge the gap in realistic object search by posing the search problem as a partially observable Markov decision process (POMDP) where the object detector and visual sensor noise in the observation model is determined by a single Deep Neural Network conditioned on complex language descriptions. Our method assumes that the object detector outputs detection accuracy over RGB images, which we incorporate into our language-conditioned observation model (LCOM) to represent dynamically changing sensor noise. With an LCOM, any language description of an object can be used to generate an appropriate object detector and noise model, and training an LCOM only requires supervised image-caption datasets which are readily available. We empirically evaluate our method by comparing against a state-of-the-art object search algorithm in simulation, and demonstrate that planning with our observation model yields a significantly higher average task completion rate (from 0.62 to 0.76), more efficient and quicker object search resulting in a higher average SPL (from 0.53 to 0.65) than with a fixed-noise model. We also demonstrate our method on a Boston Dynamics Spot robot, enabling it to handle complex natural language object descriptions and efficiently find objects in a room-scale environment without the use of fiducial markers.**

I. Introduction

Object search is a challenging task because the robot has incomplete knowledge of the environment, limited field of view, and noisy sensors. When asked to find an object in an environment, the robot must first infer the desired object from the language instruction, then efficiently move around the space to look for the object. Most images captured by the robot in this process will not contain the target object. Furthermore, even when the object is in the robot's field of view, it might not be detected due to occlusion, sensor noise, the viewing angle, the object's distance from the robot, etc.

There have been many previous works on improving the efficiency and accuracy of robot object search by using prior semantic knowledge [11], active visual search [3, 4], object manipulation [13, 22, 5], and belief factorization for multi-object search [20, 24]. However, these works have often assumed that the target objects are specified with very simple language (such as "cup" for the object's class), and thus cannot fully utilize more complex language descriptions (such as "red cup") to avoid exhaustively searching over similar object instances in the environment. Furthermore, the robot is usually assumed to have a fixed-accuracy object detector [11, 4, 20, 24] or that detection noise only comes from occlusion [13, 22]. This is challenging to scale as new occlusion models and detectors have to be made for new objects. Additionally, object detectors' accuracies are often dynamic as they depend on many factors: viewing angle, occlusion, etc. Modeling the detector as having a fixed accuracy prevents the robot from dynamically reasoning about observations it gets from the detector. This means the robot is unable to decide to gather more data instead of trusting a low-confidence detection, or trust a high-confidence detection more easily, leading to reduced object search success rates and efficiencies.


(a) A scene with the target object segmented out ("the sink on the left").

(b) Most images captured by the robot do not contain the target object.

Fig. 1: Our system takes in a language description of the target object and constructs a detector for that object. It addresses the problem that most images captured by a robot when searching for an object do not contain that object by incorporating a modified training process and using the detector's confidence score in a POMDP model for object search.

The computer vision community has developed deep learning models that can detect objects with high accuracy [7, 18], even given complex language descriptions of objects such as "red cup on the left" [8]. However, these models assume that the object is in the input image and must be localized within that image. In contrast, when searching for objects, most images captured by the robot will not contain the object being searched for (Figure 1). Moreover, models for object search do not incorporate detection confidence scores into their search process, which can inform the robot to change its viewpoint to obtain additional data for more accurate detection.

Our work addresses these problems by embedding a deep-learned object detector within an Object-Oriented Partially Observable Markov Decision Process (OO-POMDP) [20]. We modify the detector from Hu et al. [8] to handle the large number of images which do not contain the object using a different training process. Additionally, we incorporate the confidence scores from the detector into the POMDP belief updates, enabling the robot to reason dynamically. This allows us to handle complex language descriptions and search for objects more efficiently in household-sized environments. Our contributions in this paper are five-fold:

1) An experimental analysis on confidence scores from language-conditioned visual segmentation models as a proxy for different sources of observation noise.
2) A novel class of visual observation models whose detections and hyperparameters are conditioned on natural language, which we term Language-Conditioned Observation Models (LCOMs).
3) A novel decision making framework for object search that leverages LCOMs to use natural language when accounting for scene-dependent detection accuracy when estimating state uncertainty for planning.
4) A set of experiments in simulation that compare the performance of planning models using LCOMs against fixed-noise sensor models on the object search task.
5) A demonstration of our method on a Spot robot [1], which enables Spot to handle complex natural language object descriptions and efficiently find objects in a room-scale environment without any fiducial markers.

II. Related Work

Related work for robot object search generally falls into one of two categories: *model-based* and *end-to-end policy learning*. Model-based approaches separate state estimation and planning to leverage probabilistic inference, whereas model-free approaches leverage deep neural networks to learn a policy end-to-end.

There is a collection of works that employ deep learning for end-to-end visual and object search [6, 21, 16]. Our work differs from these in that we perform model-based planning to leverage our known dynamics models. Model-based planning has the potential to generalize better to new environments and systems with less training data because we encode a model of the robot's sensor and actuation capabilities, and only use deep learning for the visual processing.

Partially observable Markov decision processes (POMDPs) [9] are a framework for sequential decision making under uncertainty frequently used for object search problems. Li et al. [13] and Xiao et al. [22] treat object search in clutter as a POMDP that can be efficiently solved by using approximate online planners and constraining planning samples based on spatial constraints and conditioning action selection on the current belief state, respectively. However, their observation models are only based on occlusion statistics calculated from object region overlap. Our proposed observation model can instead account for errors not solely derived from occlusion by conditioning on complex language. Danielczuk et al. [5] train a deep learning model to segment colored masks for objects in a pile from RGB-D images and score each mask on whether it belongs to the target object. They, however, use a fixed object priority policy for action selection and assume a fixed sensor pose, while we focus on planning how to explore a space for the purpose of object search by leveraging an active sensor.

Aydemir et al. [4] frame the object search problem as active visual search and calculate candidate viewpoints based on a probability distribution over the search region, which is informed by prior correspondence knowledge between objects and semantic room categories. However, they do not account for sensor error and assume the object to be detected if it is in the robot's field of view. Atanasov et al. [3] plan a sequence of sensor views to effectively estimate an object's orientation. Similar to our work, their observation model can account for pose-dependent sensor error that is not derived from occlusion. They, in contrast, assume the object's position is known and only allow restrictive sensor motions (the camera must always face the object), which is too limited for our object search setting where the robot might not even start facing the object. Wandzel et al. [20] introduce Object-Oriented POMDP (OO-POMDP) to factorize the robot's belief into independent object distributions, enabling the size of the belief to scale linearly in the number of objects, and employ it for efficient multi-object search. Zheng et al. [24] extend OO-POMDP for efficient multi-object search in 3D space. Both these works assume simple language inputs and fixed-accuracy object detectors. Our work builds on these frameworks but explores using a deep-learned detector that takes in a natural language phrase and camera image to create an object detector.

The computer vision community has developed deep learning models trained on object segmentation datasets that can detect objects with high accuracy [7, 18, 8]. The models self-supervisedly learned to output confidence scores with their detection results. The learned confidence scores reflect the models' detection accuracy, which dynamically changes depending on the input images. We build on the model developed by Hu et al. [8] for our object detector because it is trained to handle referring expressions—complex noun phrases to describe objects.

III. Approach

Our approach models multi-object search as an OO-POMDP with an observation model corresponding to a deep-learned object detector. This approach enables us to use OO-POMCP to find policies for the robot to enable it to efficiently find objects with the deep-learned detector.

A. Planning Framework

To model the multi-object search (MOS) problem, we assume access to an occupancy-grid map, M, which is an $m \times n$ grid that marks locations as either empty or occupied, and is used for defining the space of positions and directions in the environment. We assume an object is completely contained within one of the grid cells in the map. Our main contribution is the novel language-conditioned observation model (LCOM), which modifies the observation model dynamically based on the results of the deep-learned object detector, and which we describe in detail in Section III-B. Formally, we define the MOS OO-POMDP problem as a 10 tuple: $< Obj, L, S, A, T, R, \gamma, \Omega, h_L, O >$. The tuple components are described in detail in Section B in the Appendix.

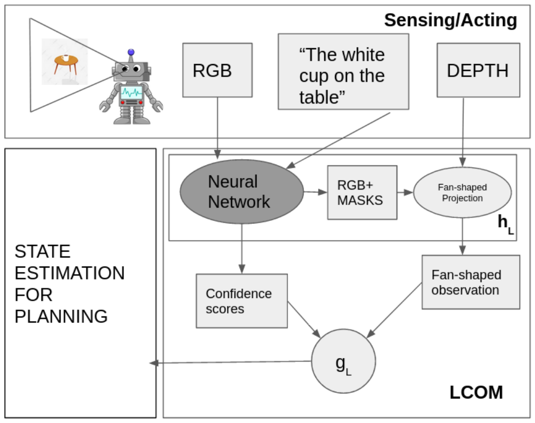


Fig. 2: **LCOM Overview**: The robot receives an RGB-D image and language description of the object, which go into h_L to produce language-conditioned confidence scores for our fan-shaped detected observations. The confidence scores are then transformed by g_L into a noise model for the detected observations, which is used to update the belief about the object's location via state estimation. Ovals are algorithms, and rectangles are data. The shaded oval is learned.

B. Language-Conditioned Observation Model (LCOM)

Figure 2 presents an overview of LCOM. When we receive an RGB-D sensor observation, ω, we can transform it into our fan-shaped sensor observation z^s and associated confidence scores c^s by using the language-conditioned observation mapper $h_L(\omega) = z^s, c^s$. LCOMs are independent of any particular instantiation of h_L as long as they satisfy the functional definition described in Section B, and for the rest of this section's discussion we treat h_L as a black box function. In our experiments, we instantiate h_L using a deep neural network.

Since we model the multi-object search as an OO-POMDP, we only discuss z_s^i and c_s^i when considering the state of object i. For object i, we treat z_i^s as having a probability of being drawn from three different events: a true positive (A_i), a false positive (B_i), or a true or false negative (C_i). More formally, let A_i be when z_i^s is from object i and $z_i^s \in V$, B_i be when $z_i^s \in V$ but z_i^s comes from other sources besides object i, and C_i be when $z_i^s = NULL$. For the $Find(X, Y)$ action, we assume a perfect sensor model where observations resulting from it are always true and give perfect information about the potential object at location (X, Y) (*i.e.,* observations resulting from $Find$ are not language-conditioned). In simulation this is reflected by knowing the ground truth state, and in real-life this can be reflected by asking a human to verify the selected location. For the $Look$ action, we parameterize the probability of each of the three events and the noise model for the location of the object conditioned on that event based on the associated confidence score c_i^s. Therefore, we can decompose the observation model $p(z_i^s|s, a)$ into:

$$p(z_i^s|s, a, c_i^s) = \sum_{e_i \in \{A_i, B_i, C_i\}} p(z_i^s|e_i, s, a, c_i^s)p(e_i|s, a, c_i^s) \tag{1}$$

If event A_i is selected, the observation is distributed normally with μ_i being object i's position and with a dynamic variance σ that depends on c_i^s based on g_L: $p(z_i^s|e_i, s, a, c_i^s) = Norm(z_i^s|o_i, \sigma)$. If event B_i is selected, the observation is distributed uniformly within the sensor region: $p(z_i^s|e_i, s, a, c_i^s) = \frac{1}{|Z|}$. If event C_i is selected, the null observation has a nearly 1 probability while any other observation has nearly 0 probability, which we implement with additive smoothing. For $p(e_i|s, a, c_i^s)$, we use g_L to map c_i to a true positive rate (which implicitly defines the false positive rates), which determines the probability of event A_i, B_i, and C_i being drawn. To summarize, g_L maps confidence scores to both a) variances for the normally-distributed probability of the objects position during true positive/negative events, and b) the probability of a true/false positive event, which determines the probability of event A_i, B_i, C_i from being drawn. In both cases, we map the continuous value of c_i^s to a discrete range of hyper-parameter values that represent high-confidence and low-confidence parameters for each setting respectively.

We note that LCOMs depend on visual input to detect potential objects in the image and report confidence scores that are used to define the sensor noise in the observation model. During state estimation with real-robot hardware, acquiring visual input is straightforwardly done by capturing images with the robot's RGB cameras. During planning, however, acquiring visual input may be challenging because it requires synthesizing novel images based on the pose of the robot and potential location of the target object. For this reason, when performing visual object search in our experiments, we only use LCOMs for updating the agent's belief during state estimation, and use a fixed observation model similar to Wandzel et al. [20] during planning based only on the 2D geometries of the known occupancy-grid map M, and we leave investigation of different 3D scene representations that capture appearance to enable planning with LCOMs to future work.

C. Object Detector

We build upon the segmentation model developed by Hu et al. [8] for our object detector as it can handle referring expressions—complex noun phrases to describe objects. The model takes in a referring expression and RGB image and outputs scores for every image pixel, which are then binarized and returned as the predicted mask for the image region described by the language. However, the original model by Hu et al. [8] was trained on the ReferIt dataset [10] which mostly contains outdoor images, whereas we are interested in detecting indoor household objects. We, therefore, additionally trained the model on the RefCOCO dataset [10] which contains referring expression annotations for segmented objects from the COCO dataset of common objects in context [14].

The original model was primarily trained on positive examples, meaning that most images contained the target object, and the model only had to localize the object within the image. In contrast, when used for object search, most images fed to the model will not contain the desired object. Thus filtering a large number of true negatives without missing the rare

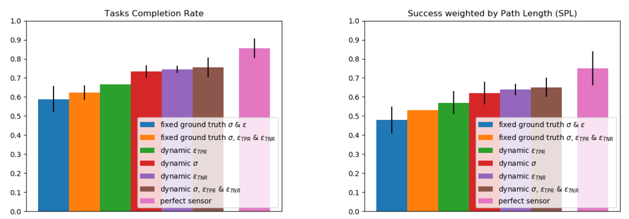


Fig. 3: **Simulation Results**: Average task completion rates and success-weighted path lengths for a simulated perfect sensor, and static and dynamic observation models with the deep-learned sensor.

true positive is key to good performance in search tasks. We augmented the model's training data with negative examples where the object described by the referring expression does not appear in the image, and thus the model should return an empty segmentation mask. This is to adapt the model to our object search setting where the desired object is usually not in the robot's field of view. Trained on the augmented data, our model achieved a true negative rate of **0.918**, a significant improvement on the original model's **0.124**.

We now describe our instantiation of h_L for our experiments based on the deep-learned segmentation model. The model takes in the RGB image and language L and outputs a segmentation mask—a binary image which identifies pixels that are part of the target object described by L. If the mask is empty, the model did not detect the object and $z_i^s = NULL$. Otherwise, we take the average of the depth value at each pixel in the mask and the coordinates of the mask's center point and project it into a location (X, Y) in the robot's fan-shaped sensing region (*i.e.,* fan-shaped projection) and return (X, Y) as z_i^s . We also retain the model's original output score for each pixel, which we average over all pixels in the mask and use as the confidence value c_i^s for the detection.

IV. Experiments and Results

Our aim is to test the hypothesis that language-conditioned observation models combined with object-oriented POMDPs can increase the speed and accuracy of a robot's ability of finding objects in complex environments. We evaluated our system both in a wide variety of environments in simulation, as well as on a real physical robot.

A. Simulation Results

We use scenes from the AI2-THOR simulator [12] to conduct our experiments. The experimental setup is further described in Section D in the Appendix. Results appear in Figure 3. We present both the task completion rate—the percentage of time the robot successfully finds the object, and success weighted by (normalized inverse) path length (SPL) [2]. We tested each different version of our model 3 times and report the average and standard error in their performances. We present results for fixed optimal values of the sensor parameters computed from the scenes in our dataset. Our deep-learned sensor achieved a true positive rate (TPR) of 0.581, true negative rate (TNR) of 0.918, and covariance of 0.827 for the normal distribution over the desired object's position. As done in Wandzel et al. [20], we set the observation model's σ and ϵ values to 0.827 and 0.581, respectively. However, we realized the benefit of also representing our sensor's true negative rate and thus modified their equation to maintain separate epsilon values for the positive and negative detection cases (ϵ_{TPR} and ϵ_{TNR}, respectively), and set ϵ_{TNR} to 0.918. This lead to a slightly improved task completion rate and SPL.

We then show results with values for each of the three observation model parameters set dynamically (based on the deep-learned detector's output confidence score), and the performance with all three parameters set dynamically. Lastly, we show the performance with a simulated noisy "perfect" sensor whose noise model perfectly matches the model used for planning by the POMDP. As expected, the performance for the perfect sensor is the best. This is because the sensor observations are being generated from ground truth with exact noise models. This provides an upper bound on our system's performance, and also indicates that if we used a more realistic sensor model, our system has the potential to perform even better. In particular, the perfect sensor will sample multiple images with the same viewpoint independently, which is not true for the deep-learned detector. All versions of our system with a dynamic observation model outperform the static versions, with a slight, insignificant benefit for the all-three version. In addition, all but one of the dynamic versions' improvements are statistically significant because the standard error bars do not overlap with those of the static versions. Overall, these results demonstrate that using a dynamic observation model significantly improves the ability of our system to find objects quickly and efficiently in realistic environments.

B. Real-World Demonstration

We provide a real-world demonstration on the Boston Dynamics Spot robot[1]. The robot takes as input a discretized map of the environment and a typed natural language phrase describing the desired object. RGB and Depth images are taken from two separate cameras in the robot's hand, and pixel correspondence between the two images is computed using the intrinsic and extrinsic matrices of both cameras. Spot moves through the environment by taking steps that are 0.6 meters in length, and all decisions are driven by the OO-POMDP until it finds the object. Scenes from our demonstration and the corresponding belief updates from using LCOMs with real robot hardware are shown in Figure 5 in the Appendix. The robot was asked to find "the green mug on the left" and successfully did so in a small number of actions. This demonstration shows our system runs on a real-world platform in a realistically sized environment, computes a policy and observations efficiently, and enables a real robot to efficiently search for and find objects.

[1]The video is available at https://youtu.be/3Z4XQUQXCsY

References

[1] Spot® - The Agile Mobile Robot. URL https://www.bostondynamics.com/products/spot.

[2] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On Evaluation of Embodied Navigation Agents. *arXiv preprint arXiv:1807.06757*, 2018.

[3] Nikolay Atanasov, Bharath Sankaran, Jerome Le Ny, George J Pappas, and Kostas Daniilidis. Nonmyopic View Planning for Active Object Detection. *arXiv preprint arXiv:1309.5401*, 2013.

[4] Alper Aydemir, Andrzej Pronobis, Moritz Göbelbecker, and Patric Jensfelt. Active Visual Object Search in Unknown Environments Using Uncertain Semantics. *IEEE Transactions on Robotics*, 29(4):986–1002, 2013.

[5] Michael Danielczuk, Andrey Kurenkov, Ashwin Balakrishna, Matthew Matl, David Wang, Roberto Martín-Martín, Animesh Garg, Silvio Savarese, and Ken Goldberg. Mechanical Search: Multi-Step Retrieval of a Target Object Occluded by Clutter. In *Proceedings of the International Conference on Robotics and Automation*, pages 1614–1621. IEEE, 2019.

[6] Aleksandra Faust, Kenneth Oslund, Oscar Ramirez, Anthony Francis, Lydia Tapia, Marek Fiser, and James Davidson. PRM-RL: Long-range Robotic Navigation Tasks by Combining Reinforcement Learning and Sampling-based Planning. In *Proceedings of the International Conference on Robotics and Automation*, pages 5113–5120. IEEE, 2018.

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[8] Ronghang Hu, Marcus Rohrbach, and Trevor Darrell. Segmentation from Natural Language Expressions. In *Proceedings of the European Conference on Computer Vision*, pages 108–124. Springer, 2016.

[9] 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.

[10] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. ReferItGame: Referring to Objects in Photographs of Natural Scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 787–798, 2014.

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

[12] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017.

[13] Jue Kun Li, David Hsu, and Wee Sun Lee. Act to See and See to Act: POMDP Planning for Objects Search in Clutter. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5701–5707. IEEE, 2016.

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *Proceedings of the European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[15] Omid Madani, Steve Hanks, and Anne Condon. On the Undecidability of Probabilistic Planning and Infinite-Horizon Partially Observable Markov Decision Problems. In *AAAI/IAAI*, pages 541–548, 1999.

[16] Farzad Niroui, Kaicheng Zhang, Zendai Kashino, and Goldie Nejat. Deep Reinforcement Learning Robot for Search and Rescue Applications: Exploration in Unknown Cluttered Environments. *IEEE Robotics and Automation Letters*, 4(2):610–617, 2019.

[17] A Alan B Pritsker. *Introduction to Simulation and SLAM II*. Halsted Press, 1984.

[18] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[19] David Silver and Joel Veness. Monte-Carlo Planning in Large POMDPs. In *Advances in Neural Information Processing Systems*, pages 2164–2172, 2010.

[20] Arthur Wandzel, Yoonseon Oh, Michael Fishman, Nishanth Kumar, Wong Lawson LS, and Stefanie Tellex. Multi-Object Search using Object-Oriented POMDPs. In *Proceedings of the International Conference on Robotics and Automation*, pages 7194–7200. IEEE, 2019.

[21] Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Learning to Learn How to Learn: Self-Adaptive Visual Navigation Using Meta-Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6750–6759, 2019.

[22] 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*, pages 8241–8247. IEEE, 2019.

[23] Kaiyu Zheng and Stefanie Tellex. pomdp_py: A Framework to Build and Solve POMDP Problems. *arXiv preprint arXiv:2004.10099*, 2020.

[24] Kaiyu Zheng, Yoonchang Sung, George Konidaris, and Stefanie Tellex. Multi-Resolution POMDP Planning for Multi-Object Search in 3D. *arXiv preprint arXiv:2005.02878*, 2020.

[25] Kaiyu Zheng, Yoonchang Sung, George Konidaris, and Stefanie Tellex. Multi-resolution POMDP planning for multi-object search in 3D. In *IEEE/RSJ International*

Conference on Intelligent Robots and Systems, 2021. Robocup Best Paper.

Appendix

A. Preliminaries

POMDPs are a framework for modeling sequential decision making problems where the environment is not fully observable. Formally, a POMDP can be defined as a tuple $< S, A, \Omega, T, O, R >$, where S, A, Ω denote the state, action, and observation spaces of the problem, respectively. After the agent takes action $a \in A$, the environment state transitions from $s \in S$ to $s' \in S$ following the transitional probability distribution $T(s, a, s') = p(s'|s, a)$. As a result of its action and the state transition, the agent receives an observation $z \in \Omega$ following the observational probability distribution $O(s', a, z) = p(z|s', a)$, and a real-valued reward $R(s, a)$.

Because the environment is partially observable, the agent does not have full knowledge of the current state s and instead maintains a *belief state* b which is a probability distribution over the states in S. The agent starts with an initial belief b_0 and updates its belief after taking an action and receiving an observation:

$$b'(s') = O(s', a, z) \sum_{s \in S} T(s, a, s') b(s) \tag{2}$$

A policy π is a mapping from belief states to actions. The agent's task is to find a policy that maximizes the expected sum of discounted rewards given an initial belief:

$$V^{\pi}(b_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \middle| a_t = \pi(b_t)\right] \tag{3}$$

where the discount factor γ determines the impact of future rewards on current decision making.

POMDPs, however, are computationally intractable for exact planning in large domains [15]. OO-POMDPs were introduced by Wandzel et al. [20] to factor the state and observation spaces in terms of objects: $p(s'|s, a) = \prod_i p(s'_i|s, a)$ and $p(z|s', a) = \prod_i p(z'_i|s, a)$, with $1 \le i \le n$ where n is the number of objects. The belief is also factored $b(s) = \prod_i b_i(s_i)$ and can be updated separately for each object. This allows the belief space size to scale linearly instead of exponentially in the number of objects. To leverage this computational gain, Wandzel et al. [20] propose a variant of Partially Observable Monte-Carlo Planning (POMCP) [19], which they term OO-POMCP, where an exact belief update is calculated with the factored state instead of approximating it by using a particle representation that is updated based on rollouts from the Monte Carlo tree search during planning.

B. Object Search Formulation

We define the MOS OO-POMDP problem as a 10 tuple: $< Obj, L, S, A, T, R, \gamma, \Omega, h_L, O >$.

1) Obj: is a set of N objects that exist in the environment (not including the robot). Each object $o_i \in Obj$ has a 2D position attribute $(x_i, y_i) = o_i$, representing its discrete position in the occupancy-grid map M. One of these objects, $o_d \in Obj$, is defined as the desired object and is used to define the reward function.
2) L: is a string of words, representing the natural language command given by the human, for example "The red cup on the table." L is only used to condition the visual observation model and transform raw RGB-D images into our fan-shaped sensor model. We defer more details to Section III-B. In this work we assume L to be given at the start of the interaction and remain constant throughout the interaction, and defer handling dynamical language to future work.
3) S: is a set of states, where each state $s \in S$ is an $N + 1$ dimensional vector $(o_1, ..., o_N, r) = s$, where o_i represents an instantiated 2D position for object i, and $r = (r_x, r_y, r_o)$ is an instantiated 2D position and discrete orientation (NORTH, EAST, SOUTH, WEST) for the robot in the occupancy-grid map M. We assume r is fully observable and the N object poses are only partially observable, yielding a mixed-observable state. This assumption is equivalent to assuming our robot is equipped with a LIDAR sensor and has previously run SLAM [17] and localized itself within that map.
4) A: is a set of actions the robot can execute to move around the map, observe object locations, and declare the desired objects as found. Specifically, we have three types of parameterized actions:
   a) $Move$(*DIR*): points the robot in direction *DIR* and moves the robot one grid in that direction, with *DIR* being either NORTH, EAST, SOUTH, or WEST.
   b) $Look$: has the robot execute a look action from its current position and orientation (r_x, r_y, r_o).
   c) $Find(X, Y)$: has the robot attempt to find the desired object o_d at grid cell (X, Y). If o_d is at (X, Y), the action will mark the object as found and terminate the episode.
5) T: is a deterministic transition function, where $Move$ actions transition the robot to different states by changing its position and orientation (r_x, r_y, r_o). $Find$ actions can transition the robot to a terminal state after finding the desired object.
6) R: is a reward function, where all $Move$ actions receive -2 reward, $Look$ receives a -1 reward, and $Find(X, Y)$ receives a 1000 reward when done at the location of the desired object, $(X, Y) == (x_{o_d}, y_{o_d})$, and -1000 otherwise.
7) γ: is the discount factor, which we set to 0.9
8) Ω: is the set of observations from our sensor, where each $\omega \in \Omega$ is a pair of RGB and depth images of size 300×300 pixels.
9) h_L: $\Omega \to z^s, c^s$ is a language-conditioned observation-mapping function that transforms raw RGB-D images into the same fan-shaped sensor model as described in [20] along with confidence scores for each object detection. The discretized fan-shaped region V only

provides a limited field of view where observations of the sensing region, z^s, consist of $|N|$ object-specific observations $z_i^s \in V \cup \{NULL\}$. If object i is not detected by the sensor, $z_i^s = NULL$. Otherwise, z_i^s is the location (X, Y) where object i is detected in V. c^s consist of all the $|N|$ object-specific observation confidence scores, c_i^s, which are real values in the $[0, 1]$ range. c_i^s represents the confidence value of observation z_i^s on object i. We describe how h_L is used for LCOMs in Section III-B, and our particular instantiation of h_L for our experiments in Section III-C.

10) O: is the Language-Conditioned Observation Model (LCOM), which assigns probabilities to observations z_t based on the current state s_t, action a_t, and natural language command L. The $Move$ actions always produce the $NULL$ observation, the $Look$ action produces noisy fan-shaped measurements conditioned on the language, and the $Find(X, Y)$ action produces an observation that is $NULL$ for every object in the sensing region except for any potential object at the location (X, Y). We discuss the observation model in more detail in the following subsection.

C. Success weight by Path Length Metric (SPL)

SPL is calculated as:

$$\frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{max(p_i, l_i)}$$

where N is the total number of tasks, l_i is the shortest path from the agent to the goal for task i, p_i is the path the agent actually took for the task, and S_i is a binary indicator of success in the task. For our experiments, p_i is the number of actions the agent actually took to search for the object, and l_i is the lowest number of actions needed to find the object. If the agent achieves a higher task completion rate but took more steps overall to find the objects, it will have a lower increase in its SPL. We collected l_i by performing planning with a perfect sensor with no noise.

D. Simulation Experiments

AI2-THOR consists of 120 near photo-realistic 3D scenes spanning four different categories: kitchen, living room, bedroom, and bathroom. We select a subset of 15 scenes along with 30 target objects (for an average of 2 objects/scene) for our experiments. Example images of the scenes used in our experiments can be found in Figure 4. The average size of a scene is 4×4 meters, which we discretize into a 16×16 cell grid map with each cell being 0.25×0.25 meters.

We use the OO-POMDP framework for multi-object search, but for simplicity only tested the system on single-object search tasks. We built upon the MOS OO-POMDP implementation by Zheng and Tellex [23] in the pomdp_py library. We modeled the OO-POMDP as having no prior knowledge of the target object's location; thus it had a uniform initial belief state over all possible object locations. We used a planning depth of 3, exploration constant of 10000, and gave the agent a maximum time of 10 minutes and 10 $Find$ actions to complete each object search task. We generated simple natural language descriptions of the objects in our experiments as input to the agent.

Fig. 4: **Simulated Scenes**: example images of the AI2-THOR scenes used in our experiments. The scene categories are: kitchen *(top left)*, living room *(top right)*, bedroom *(bottom left)*, and bathroom *(bottom right)*.

E. Real-World Demonstration

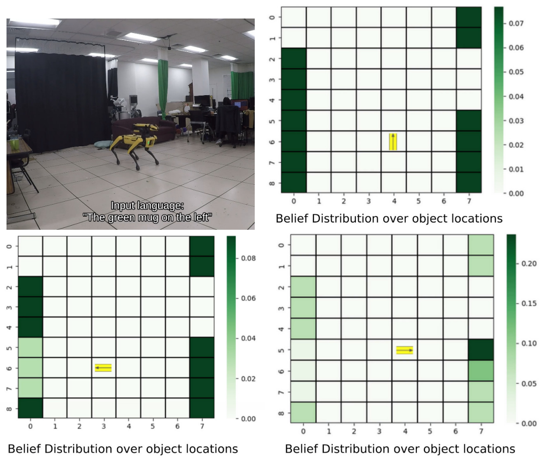


Fig. 5: **Real Robot Demonstration**: sample images from our real robot experiments with the Spot using LCOMs to find an object. Full video footage of the robot executing the task, the incoming sensor data, and the LCOM outputs is available at https://youtu.be/3Z4XQUQXCsY. *top left:* the robot is turned on and tasked with finding "the green mug on the left." *top right:* the robot's initial belief about the target object's location, which is uniform. *bottom left:* the robot moves and looks at a part of the room where the object is not located, and updates its belief that the object is most likely somewhere else. *bottom right:* the robot moves and looks where the object is actually located, and after updating its belief has maximum likelihood estimate at the target object's true location.

F. Conclusion

Our contribution is a novel observation model that makes use of the detector's confidence score to better model the detection accuracy. This enables us to perform object search using a real object detector in realistic environments. Our system can also handle complex language descriptions of objects. In addition, our method can be easily adapted to new environments without having to relearn parameters for the observation model.

Our model only considers 2D space. In future work, we plan to extend to 3D models, building on Zheng et al. [25] to model the 3D structure of objects. This extension will enable the robot to reason about different 3D viewpoints as well as predicting the structure of a partially observed object in order to gather more views to identify and localize it.

Additionally, our model does not have the ability to reason about the likelihood of different views of the same object leading to improved performance at detecting/localizing that object. Our observation model as implemented assumes that each observation is independent, so that if the robot observes the same scene from the same viewpoint, it will become more and more certain whether the object is present or not. However, in practice, when viewing the same or similar image from the same viewpoint, a deep-learned detector will give the same results; the observations are not independent samples. In the future, we could address this problem by creating a new observation model based on inverse graphics and an expected 3D model of the object appearance, enabling the robot to predict the next best view to maximally reduce its uncertainty about the object's location.

Overall we see object search as a central problem for human-robot interaction, as finding, localizing, and then grasping an object is a first step for almost anything a person wants the robot to do in the physical world. Embedding object search as a sub-component of a more sophisticated dialog system can enable the robot to engage in collaborative dialog with a human partner to interpret complex natural language commands, find and manipulate objects being referenced, and fluidly collaborate with a person to meet their needs.

G. Acknowledgments

The authors would like to thank Nick DeMarinis for all his support and help. This work was supported by NSF under grant awards IIS-1652561 and CNS-2038897, AFOSR under grant award FA9550-21-1-0214, and Echo Labs.