2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 1-5, 2023. Detroit, USA

Language-Conditioned Observation Models for Visual Object Search

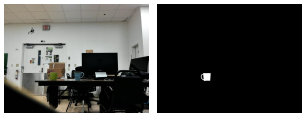
Thao Nguyen, Vladislav Hrosinkov, Eric Rosen, Stefanie Tellex
{thaonguyen, vladislav_hrosinkov, eric_rosen}@brown.edu, stefie10@cs.brown.edu

***Abstract*— Object search is a challenging task because when given complex language descriptions (*e.g.,* "find the white cup on the table"), the robot must move its camera through the environment and recognize the described object. Previous works map language descriptions to a set of fixed object detectors with predetermined noise models, but these approaches are challenging to scale because new detectors need to be made for each object. 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. We incorporate the neural network's outputs 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 readily available supervised image-caption datasets. 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 0.46 to 0.66) and more efficient and quicker object search than with a fixed-noise model. We 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.**

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 [1], active visual search [2, 3], object manipulation [4, 5, 6], and belief factorization for multi-object search [7, 8]. 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 "white 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 [1, 3, 7, 8] or that detection noise only comes from occlusion [4, 5]. This is challenging to scale as new occlusion models and detectors have to be made for new objects. Additionally, 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, potentially leading to reduced object search success rates and efficiencies. The computer vision community has developed deep learning models that can detect objects with high accuracy [9, 10], even given complex language descriptions of the desired object such as "white cup on the left" [11, 12, 13]. However, these models often assume that the object is somewhere 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).



(a) A scene with the ground truth segmentation mask for the target object ("the green mug on the left").

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

Fig. 1: Our system takes as input a natural language description of the target object, and constructs a detector for that object based on the description. 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 confidence score of the detector in a POMDP model for object search.

Our work addresses these problems by embedding a deep-learned object detector within a Partially Observable Markov Decision Process (POMDP) [14]. Our approach takes as input a language description of an object, and uses it to condition a camera-based observation model that is used to plan object-search actions and update the agent's belief about the object's pose. To achieve this, we modify the training process for the detector from Hu et al. [11] to handle the large number of images which do not contain the target object, and incorporate the detector's confidence scores into the POMDP belief updates. This allows us to handle complex language descriptions and search for objects more efficiently in household environments by reasoning dynamically. Our contributions are five-fold:

1) An experimental analysis on confidence scores outputted from language-conditioned visual segmentation mod-

2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-9190-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/IROS55552.2023.10341492

els as a proxy for different sources of observation noise.

2) A novel class of visual observation models (Language-Conditioned Observation Models – LCOMs) whose detections and parameters are conditioned on natural language.

3) A novel decision making framework for object search that leverages LCOMs to use natural language to account for scene-dependent detection accuracy when estimating state uncertainty for planning.

4) A set of experiments on simulated robot hardware 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 Boston Dynamics Spot robot [15], which enables Spot to handle complex natural language object descriptions and efficiently find objects in a room-scale environment, without using 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 [16, 17, 18, 19] or modular differentiable components [20, 21]. 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 visual processing.

POMDPs [22] are a framework for sequential decision making under uncertainty frequently used for object search problems. Li et al. [4] and Xiao et al. [5] 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. [6] 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. [3] 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 knowledge of correspondences 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. [2] plan a sequence of sensor views to effectively estimate an object's orientation. These approaches are similar to our work in that they account for realistic sensor model errors, but unlike our work they do not use a general camera-based object detector.

Wandzel et al. [7] 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. [8] extend OO-POMDP for efficient multi-object search in 3D space. Both of these works assume simple language inputs and fixed-accuracy object detectors. Our work builds on these frameworks but instead explores using a deep-learned detector that takes as input a natural language phrase and camera image to create an object detector that models varying levels of accuracy.

The computer vision community has developed deep learning models trained on object segmentation datasets that can detect objects with high accuracy [9, 10, 11, 12, 13]. The models self-supervisedly learned to output confidence scores along with their detection results. The output confidence scores do a good job of reflecting the models' detection accuracy, which dynamically changes depending on the input images. We build on the model developed by Hu et al. [11] for our object detector because it is trained to handle referring expressions—complex noun phrases to describe objects.

III. 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 (where η is the normalizing constant): $b'(s') = \eta O(s', a, z) \sum_{s \in S} T(s, a, s')b(s)$.

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: $V^{\pi}(b_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \middle| a_t = \pi(b_t)\right]$ where the discount factor γ determines the impact of future rewards on current decision making.

While many problems can be modeled by POMDPs, they are typically computationally intractable for exact planning in large domains [23]. To address the planning complexity, we use the sampling-based planner PO-UCT [24], which uses Monte-Carlo tree search with upper confidence bounds to

select an action by estimating the best state-action values using rollouts conditioned on states sampled from the current belief state, and then performs an exact belief update each time step based on the incoming observation and performed action. PO-UCT has successfully been used in robotic object-search settings [7, 8]. However, we note that our contribution is not dependent on the specific method of planning and state estimation, and LCOMs would be useful in any approach that requires a model of the observation probability distribution.

IV. Object Search Formulation

We model object search as a POMDP with an observation model corresponding to a deep-learned object detector.

A. Planning Framework

To model the object search 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 IV-B. Formally, we define the object search POMDP problem as a 10 tuple: $< o_d, L, S, A, T, R, \gamma, \Omega, h_L, O >$

1) o_d: is a desired object that exists in the environment (not including the robot). The desired object o_d has a 2D position attribute $(x_{o_d}, y_{o_d}) = o_d$, representing its discrete position in the occupancy-grid map M. The desired object is used to define the reward function.
2) L: is a string of words representing the natural language command given by the human, such as "The white cup on the table." L is only used to condition the visual observation model and transform raw images into our fan-shaped sensor model. We defer more details to Section IV-B. In this work we assume L to be given at the start 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 a 2 dimensional vector $(o_d, r) = s$, where $r = (r_x, r_y, r_o)$ is a 2D position and discrete orientation (NORTH, EAST, SOUTH, WEST) for the robot in M. We assume r is fully observable and o_d is 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 [25] and localized itself within that map, but does not know where the desired object is currently located.
4) A: is a set of actions the robot can execute to move around the map, observe object locations, and declare the desired object as found. Specifically, we have three types of parameterized actions:
   a) $Move$(*DIR*): points the robot in direction *DIR* and moves it one grid in that direction, with *DIR* being either NORTH, EAST, SOUTH, 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 each, $Look$ receives -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.
9) h_L: $\Omega \rightarrow z^s, c^s$ is a language-conditioned observation-mapping function that transforms raw images into observations from the same fan-shaped sensor model described in Wandzel et al. [7] and confidence scores for each object detection. If the desired object o_d is not detected by the sensor, $z^s = NULL$. Otherwise, z^s is the location (X, Y) where o_d is detected in the discretized fan-shaped region V. c^s represents the object-specific observation confidence score. We describe how h_L is used for LCOMs in Section IV-B, and our particular instantiation of h_L for our experiments in Section IV-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 always produces the $NULL$ observation except when $(X, Y) == (x_{o_d}, y_{o_d})$. We discuss the observation model in more detail in the following subsection.

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 IV-A, 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.

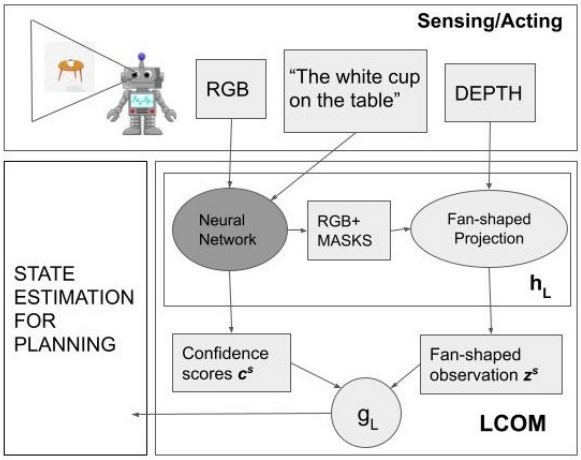


Fig. 2: **LCOM Overview**: The robot receives an RGB-D image and language description of the object. RGB-D and language go into h_L, which produces language-conditioned confidence scores c^s for our fan-shaped detected observations z^s. The confidence scores are then transformed by g_L into a noise model 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.

We treat z^s as having a probability of being drawn from three mutually exclusive and exhaustive events: a true positive (A), a false positive (B), or a true or false negative (C). More formally, let A be when z^s is from the desired object o_d and $z^s \in V$, B be when $z^s \in V$ but z^s comes from other sources besides o_d, and C be when $z^s = NULL$. We assume the $Find(X, Y)$ action always 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 events and the noise model for the observation conditioned on that event based on the associated confidence score c^s, and decompose the observation model $p(z^s|s, a)$ into:

$$p(z^s|s, a, c^s) = \sum_{e \in \{A,B,C\}} p(z^s|e, s, a, c^s)p(e|s, a, c^s) \quad (1)$$

If event A occurs, the observation is distributed normally with μ being the true object position: $p(z^s|A, s, a, c^s) = \eta' Norm(z^s|\mu, \Sigma)$. η' is the normalization factor, and the covariance matrix is defined by $\Sigma = \sigma \mathbf{I}^{2\times 2}$. If event B occurs, the observation is distributed uniformly within the sensor region: $p(z^s|B, s, a, c^s) = \frac{1}{|V|}$. If event C occurs, the null observation has nearly 1 probability while any other observation has nearly 0 probability, which we implement with additive smoothing.

Similar to Wandzel et al. [7], we define the probability of the events as $p(A|s, a) = \alpha$, $p(B|s, a) = \beta$, $p(C|s, a) = \gamma$, where $\alpha + \beta + \gamma = 1$. The probability of these events are conditioned on whether or not the desired object o_d is in the fan-shaped sensing region V, which is defined as:

$$(\alpha, \beta, \gamma) = \begin{cases} (\epsilon_{TPR}, 0, 1 - \epsilon_{TPR}) & \text{if } o_d \text{ is in } V \\ (0, 1 - \epsilon_{TNR}, \epsilon_{TNR}) & \text{if } o_d \text{ is not in } V \end{cases} \quad (2)$$

where ϵ_{TPR} represents the sensor's true positive rate, and ϵ_{TNR} represents its true negative rate.

Together σ, ϵ_{TPR}, and ϵ_{TNR} define the sensor's overall accuracy. To implement the function g_L, which transforms the confidence scores to the sensor noise in the observation model, we map the continuous value of c^s to a discrete range of hyper-parameter values that represent high-confidence and low-confidence for each setting, respectively. In our experiments, we map ϵ_{TPR} to 0.7 and σ to 0.6 when $c^s \geq 1$, and ϵ_{TPR} to 0.5 and σ to 1 otherwise. These numbers reflect that when the detector's confidence is high, the true positive rate should be high and the uncertainty over the observed position of the object should be low.

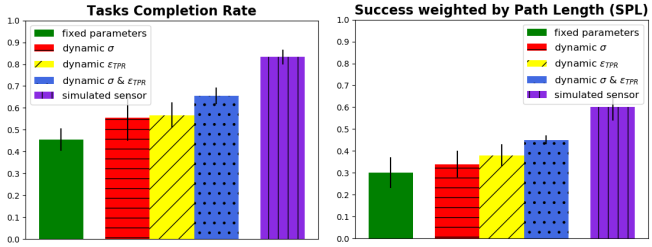
We note that LCOMs depend on visual input to detect potential objects in the image and report confidence scores that are used to define the sensor noise in the observation model. During state estimation with real-robot hardware, acquiring visual input is straightforwardly done by capturing images with the robot's camera. 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 computational efficiency, 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. [7] during planning based on the 2D geometries of the known occupancy-grid map M. Integrating different 3D scene representations into the planning module is straightforward but orthogonal to our contribution, so we defer this investigation to future work.

C. Object Detector

We build upon the model developed by Hu et al. [11] for our object detector as it can handle complex noun phrases to describe objects. The model takes in a referring expression and RGB image and outputs scores for every pixel in the image, which are then binarized and returned as the predicted segmentation mask for the image region described by the language. The loss function used for training is the average pixelwise loss: $Loss = \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} L(v_{ij}, M_{ij})$. W and H are image width and height, v_{ij} is the pixel's score, and M_{ij} is the binary ground-truth label at pixel (i, j).

The original model by Hu et al. [11] was trained on the ReferIt dataset [26] which mostly contains outdoor images, whereas we are interested in detecting indoor household objects. We, therefore, additionally trained the model on the RefCOCO dataset [26] which contains referring expression annotations for segmented objects from the COCO dataset of common objects in context [27]. Furthermore, the original model was primarily trained on positive examples such that most images contained the target object, and the model only had to learn to identify where the object was in the image. In contrast, when using a model like this for object search,

Fig. 3: **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)*.



(a) (b)

Fig. 4: **Simulation Results**: Task completion rates and success weighted by path lengths for a simulated sensor and deep-learned sensor with static/dynamic observation models.

most images fed to the model will not contain the referenced object. Thus filtering a large number of true negatives without missing the rare 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. Our model, trained on the augmented data with a learning rate of 0.01, achieved a true negative rate of **0.918**, a significant improvement over the original model's true negative rate of **0.124**.

We now describe our instantiation of h_L for our experiments based on the deep learning 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 $z^s = NULL$. Otherwise, we take the average of the depth value at each pixel in the mask as well as 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^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^s for the detection. We note that the scores were not specifically trained for this task.

V. Experiments and Results

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

A. Simulation Results

We use scenes from the AI2-THOR simulator [28] to conduct our experiments. AI2-THOR consists of 120 near photo-realistic 3D scenes spanning four different categories: kitchen, living room, bedroom, bathroom. We select a subset of 15 scenes with 30 target objects (for an average of 2 objects/scene) for our experiments. Figure 3 shows images of the scenes used in our experiments. 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 build upon the POMDP implementation by Zheng and Tellex [29] in the pomdp_py library. We modeled the 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, planning time of 10 seconds for each action, and gave the agent a maximum time of 5 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.

Results appear in Figure 4. 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) [30]. 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. The perfect sensor was able to find all 30 objects at an average of 7.8 actions per object search task.

Each different version of our model was tested 3 times and we report the average and standard error in their performance. We present results for fixed optimal values of the sensor parameters computed from the scenes in our dataset. Our deep learning model achieved a true positive rate (TPR) of 0.581, a true negative rate (TNR) of 0.918, and a covariance of 0.827 for the normal distribution over the desired object's position. We then show results with σ, ϵ_{TPR}, and both σ & ϵ_{TPR} values set dynamically based on the deep-learned detector's output confidence score As the sensor's TNR is already high, we decide to keep ϵ_{TNR} fixed. Lastly, we show the performance with a simulated sensor whose noise model perfectly matches the model used for planning by the POMDP.

As expected, the performance for the simulated sensor is

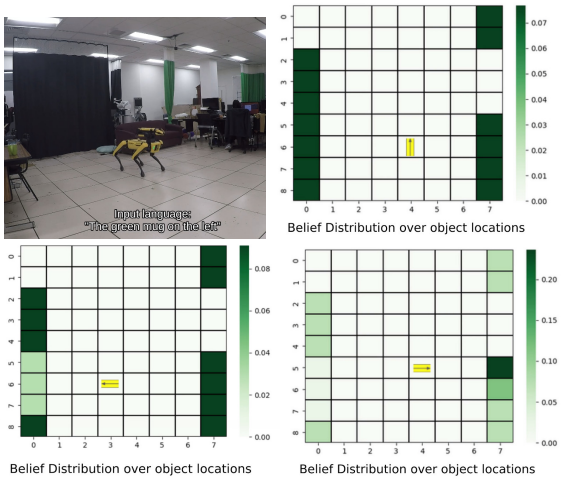
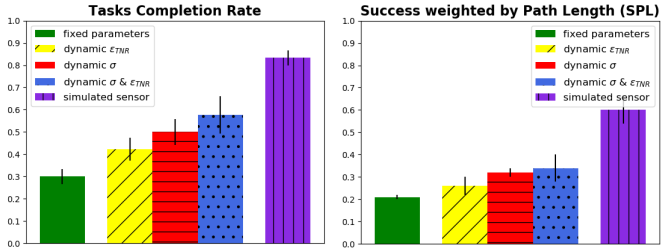


Fig. 5: **Real Robot Demonstration**: sample images from our real robot experiments with the Spot using LCOMs to find an object. *top left:* the robot is turned on and tasked with finding "the green mug on the left." *top right:* the robot's uniform initial belief about the target object's location. *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.

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 simulated 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 version. In addition, the version with dynamical σ & ϵ_{TPR} achieved a significantly higher average task completion rate and SPL than the static version (from **0.46** to **0.66**, and from **0.30** to **0.45**, respectively). On average, this version took **10.1** actions and **104** seconds to find each object, compared to the **11.5** actions and **118** seconds taken by the static version. 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. The robot takes as input an occupancy-grid 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 both cameras' intrinsic and extrinsic matrices. Spot moves through the environment by taking steps that are 0.6 meters (one grid cell) in length, and all decisions are driven by the POMDP until it finds the object. Scenes



(a) (b)

Fig. 6: **ViLD Simulation Results**: Task completion rates and success weighted by path lengths for a simulated sensor and ViLD with static/dynamic observation models.

from our demonstration and the corresponding belief updates from using LCOMs with real robot hardware are shown in Figure 5. Full video footage of the robot executing the task, the incoming sensor data, and the LCOM outputs is available at https://youtu.be/3Z4XQUQXCsY. The robot was asked to find "the green mug on the left" and successfully did so in 8 actions, where the planning and execution of each action took 10 seconds. 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 robot to efficiently search for and find objects.

C. ViLD Experiments

Given the recent success in open-vocabulary image classification/object detection powered by CLIP [31], we swapped out our trained object detector with ViLD [12], an object detector trained via vision and language knowledge distillation, for our object search experiments. ViLD takes in natural language expressions and an RGB image, proposes regions of interest within the image, computes visual embeddings for the regions, and calculates the dot product (score) between the visual embeddings and text embeddings generated by CLIP. It then returns the highest scoring image regions that correspond to the input natural language.

For each object in our experiment, we pass a simple natural language description of the object into ViLD along with the RGB image taken by the robot, and take as output the segmentation mask associated with the highest scoring image region. The mask has the same size as the input RGB image, and value 1 for every pixel within the proposed region and 0 otherwise. If ViLD does not find an image region corresponding to the input language, the segmentation mask is empty and the observation $z^s = NULL$. Otherwise, we take the average of the depth value at each pixel in the mask as well as the coordinates of the mask's center point and project it into a location (X, Y) in the robot's fan-shaped sensing region and return (X, Y) as z^s. We also retain the image region's score as the confidence score c^s.

Without fine-tuning, ViLD achieved a true positive rate (TPR) of 0.976, a true negative rate (TNR) of 0.118, and a covariance of 1.825 for the normal distribution over the desired object's position on our AI2-THOR dataset. Similar to other object segmentation methods, ViLD tends to return a non-empty segmentation mask even when the queried object

is not in the input image.

Experiment results are shown in Figure 6. The settings are the same as those described in Section V-A. We present results for fixed values of the ViLD sensor parameters. Next, we show results with σ, ϵ_{TNR}, and both σ & ϵ_{TNR} values set dynamically based on the output confidence score c^s. We map ϵ_{TNR} to 0.1 and σ to 1.0 when $c^s \geq 0.25$, and ϵ_{TNR} to 0.3 and σ to 2.0 otherwise. As ViLD's TPR is already high, we decide to keep ϵ_{TPR} fixed. Each different version was tested 3 times and we report the average and standard error in their performance.

The simulated sensor's performance is still the upper bound on the object search task. ViLD's performance trails behind our object detector which is fine-tuned for the task. However, all versions of our system with a dynamic observation model still significantly outperform the static version. The version with dynamical σ & ϵ_{TNR} achieved an average task completion rate of **0.578** and SPL of **0.34**, compared to the **0.3** and **0.21** achieved by the static version. These results demonstrate that our system works seamlessly with different object detectors and using dynamic observation models improves object search performance.

VI. Conclusion

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

Our model only considers 2D space. In future work, we plan to extend to 3D models, building on Zheng et al. [32] and Fang et al. [33] to model the 3D structure of objects. This extension will enable the robot to reason about different 3D viewpoints and predict the structure of a partially observed object to gather more views to identify and localize it. We also plan to specifically train the detector's output confidence scores to represent its detection accuracy.

Additionally, our model cannot reason about the likelihood of different views of the same object to improve its detection/localization of that object. Our current observation model assumes that each observation is independent, so 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 an 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.

Furthermore, we focus on language descriptions of the desired object to generate the object detector and observations. More complex language instructions that provide information about the location of the object such as "look in the kitchen" or "the object is to your right" can be incorporated by directly updating the agent's belief about the object's pose.

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.

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.

References

[1] T. Kollar and N. Roy, "Utilizing object-object and object-scene context when planning to find things," in *Proceedings of the International Conference on Robotics and Automation*. IEEE, 2009, pp. 2168–2173.

[2] N. Atanasov, B. Sankaran, J. L. Ny, G. J. Pappas, and K. Daniilidis, "Nonmyopic View Planning for Active Object Detection," *arXiv preprint arXiv:1309.5401*, 2013.

[3] 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.

[4] J. K. Li, D. Hsu, and W. S. 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*. IEEE, 2016, pp. 5701–5707.

[5] Y. Xiao, S. Katt, A. ten Pas, S. Chen, and C. Amato, "Online Planning for Target Object Search in Clutter under Partial Observability," in *Proceedings of the International Conference on Robotics and Automation*. IEEE, 2019, pp. 8241–8247.

[6] M. Danielczuk, A. Kurenkov, A. Balakrishna, M. Matl, D. Wang, R. Martín-Martín, A. Garg, S. Savarese, and K. Goldberg, "Mechanical Search: Multi-Step Retrieval of a Target Object Occluded by Clutter," in *Proceedings of the International Conference on Robotics and Automation*. IEEE, 2019, pp. 1614–1621.

[7] A. Wandzel, Y. Oh, M. Fishman, N. Kumar, W. L. LS, and S. Tellex, "Multi-Object Search using Object-Oriented POMDPs," in *Proceedings of the International Conference on Robotics and Automation*. IEEE, 2019, pp. 7194–7200.

[8] K. Zheng, Y. Sung, G. Konidaris, and S. Tellex, "Multi-Resolution POMDP Planning for Multi-Object Search in 3D," *arXiv preprint arXiv:2005.02878*, 2020.

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[11] R. Hu, M. Rohrbach, and T. Darrell, "Segmentation from Natural Language Expressions," in *Proceedings of the European Conference on Computer Vision*. Springer, 2016, pp. 108–124.

[12] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui, "Open-vocabulary Object Detection via Vision and Language Knowledge Distillation," *arXiv preprint arXiv:2104.13921*, 2021.

[13] W. Kuo, F. Bertsch, W. Li, A. Piergiovanni, M. Saffar, and A. Angelova, "FindIt: Generalized Localization with Natural Language Queries," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVI*. Springer, 2022, pp. 502–520.

[14] A. R. Cassandra, "A survey of POMDP applications," in *Working notes of AAAI 1998 fall symposium on planning with partially observable Markov decision processes*, vol. 1724, 1998.

[15] "Spot® - The Agile Mobile Robot." [Online]. Available: https://www.bostondynamics.com/products/spot

[16] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. 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*. IEEE, 2018, pp. 5113–5120.

[17] M. Wortsman, K. Ehsani, M. Rastegari, A. Farhadi, and R. 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*, 2019, pp. 6750–6759.

[18] F. Niroui, K. Zhang, Z. Kashino, and G. Nejat, "Deep Reinforcement Learning Robot for Search and Rescue Applications: Exploration in Unknown Cluttered Environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 610–617, 2019.

[19] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song, "CLIP on Wheels: Zero-Shot Object Navigation as Object Localization and Exploration," *arXiv preprint arXiv:2203.10421*, 2022.

[20] D. S. Chaplot, D. P. Gandhi, A. Gupta, and R. R. Salakhutdinov, "Object Goal Navigation using Goal-Oriented Semantic Exploration," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4247–4258, 2020.

[21] A. Majumdar, G. Aggarwal, B. Devnani, J. Hoffman, and D. Batra, "ZSON: Zero-Shot Object-Goal Navigation using Multimodal Goal Embeddings," *arXiv preprint arXiv:2206.12403*, 2022.

[22] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[23] O. Madani, S. Hanks, and A. Condon, "On the Undecidability of Probabilistic Planning and Infinite-Horizon Partially Observable Markov Decision Problems," in *AAAI/IAAI*, 1999, pp. 541–548.

[24] D. Silver and J. Veness, "Monte-Carlo Planning in Large POMDPs," in *Advances in Neural Information Processing Systems*, 2010, pp. 2164–2172.

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

[26] S. Kazemzadeh, V. Ordonez, M. Matten, and T. Berg, "ReferItGame: Referring to Objects in Photographs of Natural Scenes," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 787–798.

[27] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Proceedings of the European Conference on Computer Vision*. Springer, 2014, pp. 740–755.

[28] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "AI2-THOR: An Interactive 3D Environment for Visual AI," *arXiv*, 2017.

[29] K. Zheng and S. Tellex, "pomdp_py: A Framework to Build and Solve POMDP Problems," *arXiv preprint arXiv:2004.10099*, 2020.

[30] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva *et al.*, "On Evaluation of Embodied Navigation Agents," *arXiv preprint arXiv:1807.06757*, 2018.

[31] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning Transferable Visual Models From Natural Language Supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[32] K. Zheng, Y. Sung, G. Konidaris, and S. 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.

[33] Z. Fang, A. Jain, G. Sarch, A. W. Harley, and K. Fragkiadaki, "Move to See Better: Self-Improving Embodied Object Detection," *arXiv preprint arXiv:2012.00057*, 2020.