Abstract—Object finding in clutter is a skill that requires both perception of the environment and in many cases physical interaction. In robotics, interactive perception defines a set of algorithms that leverage actions to improve the perception of the environment, and vice versa use perception to guide the next action. Scene interactions are difficult to model, therefore, most of the current systems use predefined heuristics. This limits their ability to efficiently search for the target object in a complex environment. In order to remove heuristics and the need for explicit models of the interactions, in this work we propose a reinforcement learning based active and interactive perception system for scene exploration and object search. We evaluate our work both in simulated and in real world experiments using a robotic manipulator equipped with an RGB and a depth camera, and compared our system to two baselines. The results indicate that our approach, trained in simulation only, transfers smoothly to reality and can solve the object finding task efficiently and with more than 90% success rate.

Index Terms—active perception, interactive perception, clutter, interaction, manipulation, object finding, robotics

I. INTRODUCTION

ROBOTS are very efficient tools that have been used in industry for several decades. Their ability to perform specific, repetitive tasks is enabled by their very high precision and accuracy. However, such performance is only possible if the environment is known and not changing. Since nowadays numerous applications require robots to leave such environments and work in unknown spaces in collaboration with humans, new challenges arise. Some of these include maintaining a consistent representation of the world, interpreting the sensor readings even with dynamic and unknown objects in the unstructured and cluttered scenes, estimating the pose of the robot with respect to a map in all kinds of environments and conditions (indoor, outdoor, low-light, fog, rain, etc.), obstacle avoidance in human-shared spaces, etc.

Nevertheless, robots are slowly becoming more versatile tools that can tackle some of these challenges [1], [2]. However, they are still not able to cope with all of them at once. If the robots are to become more than just passive observers, the important first step is to let them interact with their surroundings. This enables a whole range of new applications such as completely autonomous object handling in stores and warehouse spaces, home assistance, elderly care, hospital assistance, disaster site inspection, etc. Such robots would save time, lower costs and improve the quality of a service in many of these industries. Furthermore, professionals would then be able to focus on tasks that are not physically demanding or dangerous.

One basic application where interaction is crucial is object search in unstructured environments. Some methods from computer vision focus on detecting objects in single RGB images. With recent progress in deep learning, such methods have shown impressive results, even surpassing human performance in the object detection task [3], [4], [5].

However, a single image is usually not sufficient to find an object if it is hidden in a clutter and occluded by other objects, and therefore not visible. Because of this, active perception, where the camera is actively moved, is required to observe the scene from different viewpoints. In 3D scene reconstruction [6], changing the camera viewpoint is required to reveal details of the scene that are obstructed by other objects [7] or to gain more information on the scene in order to grasp an object [8]. Selecting the “next best view” where the camera should move can be based on the current knowledge of the scene and the given task [9], or even selected based on a greedy heuristic [10] that does not account for long term planning reward.

Even with allowing camera motions and smart selections of the next views, objects can still be hidden in a pile or completely occluded by other objects. Therefore, interaction with the environment is necessary to remove these occluders. Interactive perception builds upon active perception and combines camera motion with environment interaction. It uses information from the interaction to get a better understanding of the scene, and then based on the current state of the scene, it plans the next best action. Such systems have been demonstrated for different applications such as segmentation, grasp planning or object recognition [11], even using Reinforcement Learning (RL) [12]. However, these usually use predefined...
actions [13] and do not consider exploration. As a result, such methods are constraining the robot to a limited set of interactions which are usually not efficient for object searching in cluttered environments.

Our approach to object finding in clutter, using interactive perception, is built upon an RL based control algorithm and a simple color detector. Since object detection in a single image is already a very well studied problem and to simplify our problem, we assume that the target object is of a specific color and use a color detector to provide us with the detections. In addition to these, we encode the scene state using a discretized Truncated Signed Distance Field (TSDF) volumetric representation [14]. Afterwards, the next action is determined by an RL algorithm based on the current encoded state and the knowledge obtained from past experiences. Since specifying which actions are good and which are bad is very hard and non-intuitive, supervised methods are not adapted to this task. Furthermore, it is more intuitive to define rewards for the agent in such a complex task than to manually engineer policies. With such a framework, we show that we can learn a policy that can effectively search an object in a cluttered scene. The main contributions of this paper are:

- an RL approach to active and interactive perception based object search in clutter,
- a compressed and interpretable volumetric representation of the environment suitable for RL,
- experimental evaluation of the framework on both a simulated and a real world robotic system, including a comparison to baselines.

II. RELATED WORK

Finding objects hidden in clutter requires robots to actively explore and manipulate their environment. In this section, we present an overview of related works from the fields of active and interactive perception, as well as their intersection with robotic manipulation.

In contrast to recent works in object detection and semantic segmentation that typically only operate on single fixed images [3], [4], active perception considers the problem of optimizing the sensor placement in order to perform a task [15], and has found applications in a variety of tasks such as mapping [16] and object reconstruction [17], [18], [6]. A common approach is to choose the next-best-view in order to maximize an information gain metric [19]. Velez et al. [20] presented a planner for improved object detection in an unknown scene taking into account the confidence of the object detector and uncertainty of the robot's pose. Choosing a good metric is often challenging and task specific [21], however measures such as Shannon entropy [22] or KL-divergence [23] are often used. Recently new information gain formulations have been proposed for volumetric reconstruction of unknown scenes [24], [10]. Acquiring more information of a scene is also beneficial for manipulation. Kahn et al. [7] model the probability of grasp locations in occluded regions as a mixture of Gaussians and optimize for sensor placements that minimize uncertainty of these regions, while Morrison et al. [8] choose informative viewpoints based on a distribution of grasp pose estimates.

However, in many cases robots can improve perception through physical interaction with their environment. The goal of interactive perception is to learn the relationship between actions and their sensory response [11]. Applications cover a wide range of problems, such as object recognition [25], object segmentation [26], [23], and inferring physical properties of objects [27], [28]. Dogar et al. [29] generate plans that minimize the expected time to find an occluded object using a visibility-accessibility graph, assuming that the target is the only hidden object in the scene. Li et al. [30] formulate object search as a POMDP and solve it with an approximate online solver. Their approach however relies on highly accurate segmentations. Xiao et al. [31] address some of these limitations, assuming knowledge of the exact number and the geometric properties of the objects in the scene. In a recent work, Danielczuk et al. use the output of a grasp planner [32] and different heuristics to choose between grasping, suction, and pushing actions to extract a target object from a heap [13]. Similarly, Breyer et al. [33] leverage RL to design simple actions in order to clear a table of objects, with a visual input mounted on the end-effector, similarly as in our work.

The majority of these works rely on predefined action primitives, such as pushing and grasping, reducing the set of possible actions to choose from. In contrast, we propose to use RL in order to learn a suitable policy and allow the agent to freely control the pose of its end-effector. Gualtieri et al. [34] also use RL to learn 6 Degrees of Freedom (DOF) movement of a robot arm. However, to simplify the problem, they constrain the agent to focus attention on sub-regions of the current observation, which requires the target to be visible from the beginning. Cheng et al. [12] train an Recurrent Neural Network (RNN) to predict gripper displacements that push an object to a target location, while being robust to occlusions.

A critical component of the latter work was to incorporate the prediction of an object detector in the observation encoding. This agrees with the findings of Sax et al. [35], who motivate that task-specific, mid-level features offer better learning efficiency and generalization compared to raw images. This work thus tries to focus on simple and interpretable inputs, allowing light and efficient learning of interactive tasks.

III. METHOD

In our method, we tackle the problem of finding specific objects in cluttered environments where interaction is required, using a robotic arm with wrist-mounted RGB and depth cameras. Our goal is to find a mapping from sensor measurements to end-effector displacements, i.e. a policy $\pi(a_t|s_t)$, using a model-free RL algorithm. This algorithm should, based on the current representation of the scene and past experience from the training process, predict the next action to perform.

To find the target we use a color detector to provide us with object detection in 3D. These 3D detections together with a volumetric scene representation are used to define the state of our system. Our action space has 5 DOF and one additional value to let the algorithm decide to terminate the searching procedure. The overview of the system is shown in Figure 2.
A. Problem Formulation

We model the problem as a discrete, finite horizon Markov Decision Process (MDP), with the set of all states $S; A(s)$ is the set of all valid actions for a given state $s$, $r : S \times A \times S \rightarrow R \subset \mathbb{R}$ a real-valued reward function, $\gamma$ the discount factor, and $p : S \times R \times S \times A \rightarrow [0, 1]$ a deterministic function that defines the dynamics of the MDP.

At each time step $t$, the RL agent observes the current state $s_t \in S$. Based on the parameterized policy $\pi_{\theta}(a_t | s_t)$, it decides which action $a_t \in A$ to take next. Upon completion of this action the system transitions into a new state $s_{t+1} \in S$ based on the system dynamics $p$ and receives a reward $r_{t+1}(s_t, a_t, s_{t+1})$. For each episode (one run of the algorithm) this is repeated until the maximum number of time steps $T$ is reached or the agent reaches the terminal state. The final goal of the on-policy RL is to find the optimal actions for the given states that maximize the expected return, i.e. the total discounted reward.

B. Agent Model

We first define how the agent converts the sensor measurements into a state representation that is useful for its RL control module. The control module, based on the current state, plans the next action for the robot. In order to plan meaningful actions, the module is trained in simulation where a reward is given for each episode. The reward is shaped in such a way that the algorithm is guided to successfully complete the given task.

1) State Representation: The choice of a state representation is a crucial component for learning good policies. The representation has to be general enough to capture all the relevant information for the successful completion of the task. In our object finding task, the representation has to allow the agent to distinguish the boundaries of the objects, to reason about the spatial relationships between the objects, to distinguish the target object, and to provide information about unobserved areas. However, the representation should also be compact enough to keep the network size reasonable and informative enough to allow the convergence of the RL control module.

TSDF is a volumetric representation mainly used in 3D reconstruction [14], which discretizes the scene into 3D voxels and assigns a value to each of them. These values range from -1 to 1, where negative values are used to define distances behind the surface, and positive values are used in front of the surface. 1 also indicates observed free-space in front of the surface, and -1 indicates unobserved space. At all times, the 0 crossing in the field defines the surface. The voxel values are averaged over all integrated depth measurements. Additionally, each voxel stores a weight value which, in our approach, is incremented by 1 each time a new measurement in this voxel is observed. It represents the measurement uncertainty of each voxel. The main advantage of this representation is that it has a notion of free-space and unobserved space, where both unexplored areas and regions inside the objects are considered unobserved.

The volumetric TSDF representation provides us with spatial information about the scene. However, since we are interested in specific objects, additional information is required. Therefore, we have extended this voxel grid with an additional voxel value that indicates if the voxel is part of the target or not. This information is provided by the object detector which works on the same RGB-D frame. Hence, these detections can be integrated into the grid the same way as depth measurements. Since the focus of our work is on interactive perception, we have simplified the detection problem by using an object of specific color as a target. Therefore, a color detector in CIELAB color space is used as a target detection method.

Depending on the resolution of the voxels, such a representation can contain a very large amount of information. Since it would be infeasible to use this information directly in the RL algorithm, we need to encode it into a smaller sized vector that can be efficiently used for learning. To do this, we split the space into two parts, one above the fingers of the gripper and one below. Then we sum the values of the TSDF and the 3D detections separately along the $z$ coordinate to get 4 flattened maps. These maps are then discretized into a $3 \times 3$ grid, where the middle cell is further separated into a $3 \times 3$ grid. As a result, we obtain 4 maps encoded with 17 values, which are then concatenated into a 68 dimensional vector that represents the state, depicted in Figure 3.

The global representation of the environment is stored in the robot’s base frame. However, prior to generating the encoded vector, the TSDF and the 3D detection grids are transformed to the end-effector frame projected onto the xy-plane. This is done by translating the two grids horizontally so that the end-effector’s position is at the center of the grids. Because of this, there is no need to additionally append the end-effector pose to the robot state since all the measurements are relative to it.

Since we also allow interaction with the scene, to be able to update the representation in case something moves, we modified the depth image integration to limit the maximum weight in the TSDF grid to 2. This means that all past measurements have a small weight (high uncertainty) and when new measurements are obtained, old ones are quickly forgotten.
Such a strategy allows us to update the representation very quickly by only taking a few additional measurements.

2) Actions: To perform the object finding task successfully, the robot is required to move the camera in 3D, as well as to interact with the objects in the scene. To generalize well to both tasks of active and interactive perception, we decided to control relative displacements of the gripper in the end-effector frame. Therefore, we have 3 variables for the translations in \(x\), \(y\), and \(z\) direction. Since exploring the unobserved areas can be much faster by tilting the end-effector, we additionally added roll (\(\phi\)) and yaw (\(\psi\)) angles. The final degree of freedom, pitch (\(\theta\)), is omitted here to reduce the complexity of the RL agent. To handle scenarios where no object is present, we use an additional binary variable that allows the agent to terminate the episode. This allows the agent to finish successfully even if the target object is not in the scene.

3) Reward Function: Given the complexity of the task, the reward should be shaped such that it guides the agent towards the optimal solution. It requires balancing between exploration and exploitation, i.e., discovering unobserved space and evaluating if any potential 3D detection is actually part of the target or not. Therefore, the reward needs to be carefully crafted to avoid getting stuck at a local optimum that prevents the algorithm from converging.

Our reward has the following components:

- Detection reward - the agent receives a reward proportional to the number of pixels detected that belong to the target object if the number of detected pixels is less than a defined threshold. This encourages the agent to observe more pixels from the target by moving towards it. However, in order to discourage the agent from obtaining this reward repeatedly at each step without terminating, we apply a weight to each detected pixel of the target that decreases when the pixel has already been observed multiple times.

- Exploration reward - the agent is rewarded for exploring the scene, proportionally to the number of newly observed voxels (the number of voxels with value -1 is compared before and after the action). In order to balance the importance of the detection relative to the exploration, this reward is clipped to a maximum value.

- Time penalty - to encourage the agent to finish the task as fast as possible, a constant negative reward is given at each step.

- Final detection reward - if the agent observes enough of the target object and decides to terminate, it gets a final detection reward.

- Final exploration reward - if there is no target object in the scene and the agent terminates, it gets a final exploration reward equal to the final detection reward.

- Final failure penalty - if the agent terminates before achieving the task it gets a failure penalty.

At each iteration, if the agent does not terminate or find the object, the best that it can achieve is a 0 reward since the exploration and the detection rewards summed together are equal to the time penalty reward. Getting a total positive reward means that the agent decided to terminate after having seen at least part of the target or when there was no target in the scene.

4) Policy: To find a successful policy for our task, we used a fully connected network to map the current state to the next action and update its weights with Proximal Policy Optimization (PPO) [36], a state-of-the-art policy gradient method. The output layer uses \(\tanh\) activations such that the mean of predicted actions is in the \(-1\) to 1 range.

C. Simulation

In order to train the RL model, we use a simulation environment based on the Bullet physics engine [37]. In the simulation, we added a disentangled robot hand whose pose is controlled by a force constraint. Furthermore, we attached a virtual RGB-D camera to the end-effector, mimicking the real robot setup. Depth and RGB images are generated using bullet’s camera renderer. The simulated scene is surrounded by walls, and initial positions and number of objects are randomized in each episode.

D. Transfer to Reality

Transfer to reality was done without any fine-tuning of the model on the real robot. Since robot poses are provided in the base frame of the robot, and the depth measurements are in the camera frame, the transformation between the two frames is required to properly integrate the measurements into the TSDF grid. The transformation was obtained using our hand-eye-calibration toolbox [38]. The fact that we use a TSDF representation, which averages depth images during integration time, allows us to have some noise in the depth image and small errors in the pose estimates of the camera. The discretization of the volume used to represent the state, additionally, allows small scene reconstruction errors. Such a choice of state makes the precise details of the scene less important, and as a result, allows for a better abstraction and generalization of real data.
TABLE I: Maximum values for each component of the reward function for each step of the episode.

| Reward                      | Value |
|-----------------------------|-------|
| Maximum detection reward    | 0.8   |
| Maximum exploration reward  | 1.2   |
| Time penalty                | -2    |
| Final detection reward      | 500.0 |
| Final exploration reward    | 500.0 |
| Final failure penalty       | -300.0|

IV. Evaluation

In our experiments we evaluate the performance of our approach both in simulation and in the real world, and compare it to two baselines, one for the active perception task and one for the interactive perception task. As a metric for our evaluation we use success rate, number of steps per episode, and total time per episode.

A. Experimental Setup

We evaluate our method with a position controlled 7-DoF arm of an ABB Yumi. An RGB Chameleon 3 camera, coupled with a CamBoard pico flexx providing depth images, is mounted on the wrist of the robot. We use a similar model in the physics engine Pybullet\(^1\) to perform the task realistically in simulation. The position of the gripper is randomly initialized between each run.

The scene is composed of cubes, either scattered on the ground in the active task or forming piles in the interactive one. The target, when there is one, is always the single red object in the scene and the other objects are either all green (in simulation) or with diverse colors (real world). The different setups are visualized in Figure 4. The TSDF is integrated with a modified version of the library Open3D [39] that can integrate the detections of the target as well. The target detection is performed by first converting the RGB image into the CIELAB color space and then labelling the red pixels as belonging to the target.

Both tasks are trained in simulation using the PPO implementation of the OpenAI Baselines framework [40] for 5 epochs, with a batch size of 3000, a minibatch size of 300, a discount factor of 0.99, an entropy bonus of 0.01 and the reward values shown in Table I. The network used for inference is a Multi Layer Perceptron with 2 hidden layers, 300, a discount factor of 0.99, an entropy bonus of 0.01 and 5 epochs, with a batch size of 3000, a minibatch size of 3000.

B. Baselines

In order to evaluate the performance of our RL based algorithm for object search, we have implemented two baselines, one for the active perception task and one for the interactive perception task.

1) Active Perception: For the active perception task, where the target object can be revealed just by moving the camera, we have implemented a one-step Greedy Next-Best-View agent [19]. This agent samples \(n\) poses within the proximity of the current pose and evaluates each of them based on how many voxels would be observed if the agent would have moved to that location, assuming that there are no objects in the unobserved area. The sample that obtains the highest score is taken as the next action. If one part of the red object was detected, then the samples are evaluated based on which one brings the target object closer to the camera image center. Once the agent explores more than 97% of the voxels, or it sees more than 20% of the red object, it terminates the search. In our experiments, we used \(n = 10\) pose samples in simulation and \(n = 30\) pose samples in the real world experiments. Each sample is obtained by sampling actions uniformly in the allowed range and applying them to the current pose of the

| Simulation | Real World |
|------------|------------|
| Min num. objects | 5 | 15 |
| Max num. objects | 25 | 15 |
| Num. of piles | 0 | 4 |
| No target prob. | 0.1 | 0.0 |
| Exploration ratio | x | x |
| TSDF grid resolution | 100 | 100 |
| SDF truncation dist. [m] | 0.04 | 0.04 |
| Workspace length [m] | 0.6 | 0.6 |
| Max. translation [m] | 0.06 | 0.06 |
| Max. roll rotation | 0.15 | 0.15 |
| Max. yaw rotation | 0.15 | 0.15 |
| Time horizon | 150 | 50 |
| Num. experiments | 1000 | 20 |

TABLE II: Simulation and real world experiment parameters for both small and large (L) active perception (AP), and interactive perception (IP) tasks. Exploration ratio specifies a percentage of scenes for which there are no piles, therefore it is only used in the interactive task.

\(^1\)https://pybullet.org
agent.

2) Interactive Perception: For the task where the target object can be hidden in a pile, a Grid Exhaustive Search agent has been implemented. The agent traverses the whole workspace at a fixed low height (just above the lowest cube) pushing any cube along the way that is placed on top of another cube, thus revealing any hidden cubes below. To achieve this, the agent divides the workspace into a regular grid which defines a graph whose nodes are intersections of the grid. The nodes are one maximum translation allowed apart from each other such that the agent can move from one node to another in one step. Since the initial position of the agent is randomized, the problem is equivalent to the Traveling Salesman Problem (TSP) and is NP-complete. Therefore, we used a 2-opt algorithm [41] to efficiently compute a route that is close to optimal. The robot then traverses the route until it detects the object or reaches the maximum number of steps. This efficient strategy shows a high success rate to solve the task (97.3% in normal scenes and 96% in large ones).

While our approach seems overall on par with the greedy baseline in terms of success rates (Table III), it solves the task (97.3% in normal scenes and 96% in large ones). As the real experiment is a time-consuming and resource demanding task, the evaluation on the real setup for both active

C. Simulation Results

The active task in simulation is evaluated on 2 scenes of different size: a normal and large (L) one. Both tasks are evaluated with 1000 episodes. The same number of episodes was used to evaluate the interactive task. The simulation parameters are shown in Table II and the results are shown in Figure 5.

1) Active Perception: The agent trained for active perception learned to quickly go up in order to get a good overview of the scene and then to roll to observe the remaining unobserved parts. This efficient strategy shows a high success rate to solve the task (97.3% in normal scenes and 96% in large ones). While our approach seems overall on par with the greedy baseline in terms of success rates (Table III), it solves the task significantly faster than the baseline, with an average time per episode of 8.45s for the Active 3D Exploration against 22.38s for the Greedy Next-Best-View (reduction of 62%) and 17.58s for the Active 3D Exploration on large scene against 126.07s for the Greedy Next-Best-View on the same sized scene (reduction of 86%). This is due to the sampling nature of the Next-Best-View algorithm and to the fact that views are evaluated based on the number of newly unobserved voxels in the scene which can be costly to obtain. It is important to note here that the Greedy Next-Best-View baseline has the knowledge of the whole map and uses this knowledge to compute the number of unobserved voxels, while our approach uses only the local map to obtain the state. Nevertheless, even though the baseline is in a favourable position, we achieve comparable results for the success rate.

Learning to terminate is challenging for the agent since it only has a compressed information about a local map, and if it is not big enough to cover the whole TSDF volume the agent has no way to know if it explored the whole area. Furthermore, the discretized state can be either normalized or not. In the first case, we lose the absolute number of unobserved voxels in the scene and the number of point detections, however, we are better at generalizing to different scenes with more or less objects and it forces the agent to explore even the small unobserved areas. Additionally, the normalized state makes it very difficult for the agent to determine if it should terminate or not when there is no object in the scene. In the second case, we know the exact amount of unobserved voxels in the local map, however, we are very scene dependent. To force the agent to explore more, normalized state was used in the experiments.

2) Interactive Perception: Results show that our learned approach for interactive perception is more efficient than the Grid Exhaustive Search baseline (both in terms of number of steps and timing: 39.60s for ours against 63.89s for the baseline), while keeping a slightly lower success rate (Table III). To achieve this outcome, our agent has learned to stay down and to push any clutter of objects that is sufficiently big to appear as unobserved in the TSDF.

The exhaustive baseline can still fail the task when it accidentally pushes the target away without seeing it, so that it needs a second exploration of the whole scene to find the target. Since the target is also not present in the scene in 5% of the scenarios, it always fails in these particular scenarios as it has no means to know that there is no target.

Our learned method has the advantage of being significantly faster, especially as the grid search does not scale well to an increase in the size of the workspace (quadratic in terms of the side of the scene).

D. Real-World Results

As the real experiment is a time-consuming and resource demanding task, the evaluation on the real setup for both active
and interactive task is performed over 20 runs. The results are shown in Table IV and visualized in Figure 6. The scene parameters used are shown in Table II.

1) Active Perception: The comparison between the learned Active 3D Exploration and Greedy Next-Best-View baseline shows comparable results to the ones obtained in simulation, both in terms of number of steps per episode and success rates. Nevertheless, this shows that the active task based on RL can be successfully transferred to reality with a high success rate.

2) Interactive Perception: The Interactive 3D Exploration method performs well on the real world experiment with a success rate of 100% (over 20 episodes), similarly as the Grid Exhaustive Search baseline. Furthermore, the learned interactive method outperforms the baseline in terms of efficiency (number of steps per episode). These results confirm the outcome of the evaluation in simulation and show that a transfer to reality can be performed without any fine-tuning on real data.

### E. Limitations

One limitation of our system is that it is not aware of the size of the workspace and its current pose within that workspace. This can cause the agent to get stuck by trying to repeat the action to go up, however not being able to execute it.

In the real world experiments, because of the small size of the workspace, the robot was not always able to plan the direct paths to the target location. Additionally, the agent was trained with a detached gripper in simulation, therefore allowing transition to any pose, which is unfeasible on the real robot.

The current state of the robot encodes information about the unobserved voxels and object detections from a single object. However, it would need to be extended if we were to allow multiple object hypotheses in our detector.

### V. Conclusion

In this work, we have shown that an RL approach is feasible to learn a successful policy for an object finding task that requires both active and interactive perception. With the encoded volumetric representation of the environment and reward shaping, we are able to achieve a high success rate. By comparing our approach to two baselines, we have shown that it is more time efficient than the Greedy Next-Best-View, since it does not require expensive sampling in each iteration, and more step efficient than the Grid Exhaustive Search, as it does not do naive brute force search, but instead interprets the encoded state and acts upon it. Furthermore, our agent learned to decide when to terminate the sequence. Results obtained in the real robot experiments validate the findings provided by the simulations.

In future work, we plan to integrate a more sophisticated object detection pipeline such that we do not rely on the fact that the target object has a specific color. Furthermore, we plan to extend our volumetric representation to also encode dynamic objects and avoid the need to “forget” the volume where objects moved.

### Acknowledgements

This work was supported by the Amazon Research Awards program, the Swiss National Science Foundation (SNF) through the National Centre of Competence in Research (NCCR) on Digital Fabrication, the Luxembourg National Research Fund (FNR) 12571953, and ABB Corporate Research Center.

### References

[1] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman, “Analysis and observations from the first amazon picking challenge,” IEEE Transactions on Automation Science and Engineering, 2016.

[2] G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield et al., “The grand challenges of science robotics,” Science Robotics, vol. 3, no. 14, p. eaae7650, 2018.

[3] K. He, G. Gkioxiari, P. Dollar, and R. Girshick, “Mask R-CNN,” in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2961–2969.

[4] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7263–7271.

[5] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, “Light-head R-CNN: in defense of two-stage object detector,” CoRR, vol. abs/1711.07264, 2017.

[6] S. Kriegel, C. Rink, T. Bodenmüller, and M. Suppa, “Efficient next-best-scan planning for autonomous 3D surface reconstruction of unknown objects,” J Real-Time Image Proc, vol. 10, no. 4, pp. 611–631, Dec. 2015.

[7] G. Kahn, P. Sujan, S. Patil, S. Bopardikar, J. Ryde, K. Goldberg, and P. Abbeel, “Active exploration using trajectory optimization for robotic grasping in the presence of occlusions,” in 2015 IEEE International Conference on Robotics and Automation (ICRA), May 2015, pp. 4783–4790.

[8] D. Morrison, P. Corke, and J. Leitner, “Multi-View Picking: Next-best-view Reaching for Improved Grasping in Clutter,” in 2019 International Conference on Robotics and Automation (ICRA), May 2019, pp. 8762–8768.

[9] E. Dunn and J.-M. Frahm, “Next best view planning for active model improvement,” in BMVC, 2009.

[10] J. Delmerico, S. Isler, R. Sabzevari, and D. Scaramuzza, “A comparison of volumetric information gain metrics for active 3D object reconstruction,” Auton Robot, vol. 42, no. 2, pp. 197–208, Feb. 2018.
[11] J. Boltj, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. S. Sukhatme, “Interactive Action: Leveraging Action in Perception and Perception in Action,” in IEEE Transactions on Robotics, vol. 33, no. 6, pp. 1273–1291, Dec. 2017.

[12] R. Cheng, A. Agarwal, and K. Fragkiadaki, “Reinforcement Learning of Active Vision for Manipulating Objects under Occlusions,” in Conference on Robot Learning, Oct. 2018, pp. 422–431.

[13] M. Danielczuk, A. Kurenkov, A. Balakrishna, M. Matl, D. Wang, R. Martin-Martin, A. Garg, S. Savarese, and K. Goldberg, “Mechanical Search: Multi-Step Retrieval of a Target Object Ocluded by Clutter,” in 2019 International Conference on Robotics and Automation (ICRA), May 2019, pp. 1614–1621.

[14] B. Curless and M. Levoy, “A volumetric method for building complex models from range images,” in Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, ser. SIGGRAPH ’96. New York, NY, USA: ACM, pp. 303–312. [Online]. Available: http://doi.acm.org/10.1145/237170.237269

[15] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, “Revisiting active perception,” Auton Robot, vol. 42, no. 2, pp. 177–196, Feb. 2018.

[16] H. Carrillo, I. Reid, and J. A. Castellanos, “On the comparison of uncertainty criteria for active SLAM,” in 2012 IEEE International Conference on Robotics and Automation, May 2012, pp. 2080–2087.

[17] R. Pito, “A solution to the next best view problem for automated surface acquisition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 10, pp. 1016–1030, Oct. 1999.

[18] S. Wenhardt, B. Deutsch, E. Angelopoulou, and H. Niemann, “Active Visual Object Reconstruction using D-, E-, and T-Optimal Next Best Views,” in 2007 IEEE Conference on Computer Vision and Pattern Recognition, Jun. 2007, pp. 1–7.

[19] C. Connolly, “The determination of next best views,” in 1985 IEEE International Conference on Robotics and Automation Proceedings, vol. 2, Mar. 1985, pp. 432–435.

[20] J. Velez, G. Hemann, A. S. Huang, J. Posner, and N. Roy, “Planning to Perceive: Exploiting Mobility for Robust Object Detection,” in Twenty-First International Conference on Automated Planning and Scheduling, Mar. 2011.

[21] S. Chen, Y. Li, and N. M. Kwok, “Active vision in robotic systems: A survey of recent developments,” The International Journal of Robotics Research, vol. 21, no. 10, pp. 1343–1377, Sep. 2011.

[22] P.-P. Vazquez, M. Feixas, M. Sbert, and W. Heidrich, “Viewpoint Selection Using Viewpoint Entropy,” in Proceedings of the Vision Modeling and Visualization Conference 2001, ser. VMV ’01. Aka GmbH, 2001, pp. 273–280.

[23] H. van Hoof, O. Kroemer, H. B. Amor, and J. Peters, “Maximally informative interaction learning for scene exploration,” in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct. 2012, pp. 5152–5158.

[24] S. Iser, R. Sabzevari, J. Delmerico, and D. Scaramuzza, “An information gain formulation for active volumetric 3D reconstruction,” in 2016 IEEE International Conference on Robotics and Automation (ICRA), May 2016, pp. 3477–3484.

[25] J. Sinapov, C. Schenk, and A. Stoytchev, “Learning relational object categories using behavioral exploration and multimodal perception,” in 2014 IEEE International Conference on Robotics and Automation (ICRA), May 2014, pp. 5691–5698.

[26] P. M. Fitzpatrick and G. Metta, “Towards manipulation-driven vision,” in IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1, Sep. 2002, pp. 43–48 vol.1.

[27] Z. Xu, J. Wu, A. Zheng, J. Tenenbaum, and S. Song, “DensePhysNet: Learning Dense Physical Object Representations Via Multi-Step Dynamic Interactions,” in Robotics: Science and Systems XV, vol. 15, Jun. 2019.

[28] R. Martin-Martin and O. Brock, “Building Kinematic and Dynamic Models of Articulated Objects with Multi-Modal Interactive Perception,” in 2017 AAAI Spring Symposium Series, Mar. 2017.

[29] R. Pito, “A solution to the next best view problem for automated surface acquisition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 10, pp. 1016–1030, Oct. 1999.

[30] J. K. Li, D. Hsu, and W. S. Lee, “Act to See and See to Act: POMDP planning for objects search in clutter,” in 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2016, pp. 5701–5707.

[31] Y. Xiao, S. Katt, A. t Pas, S. Chen, and C. Amato, “Online Planning for Target Object Search in Clutter under Partial Observability,” in 2019 International Conference on Robotics and Automation (ICRA), May 2019, pp. 8241–8247.