Reinforcement Learning for Active Search and Grasp in Clutter

Thomas Pitcher[1], Julian Förster[2], Jen Jen Chung[1]

***Abstract*— This paper presents an Active Search policy that balances between moving the camera and removing occluding objects to search for and retrieve a target object in clutter. While both types of action can reveal unobserved parts of a scene, they typically vary in execution complexity and time. Our proposed method explicitly reasons about the occluded spaces in the scene where the target object may be hidden, and uses reinforcement learning to compute the value of each action with the ultimate goal of finding and retrieving the target object in minimal time. Results in simulation and real-world experiments demonstrate a significant improvement in both task execution speed and success rate compared to baseline grasping strategies.**

I. Introduction

When humans need to find and retrieve hidden objects we are able to fluently reason between changing our viewpoint and moving objects within the space. Robots typically approach such a task as a *mechanical search* problem, where the goal is to locate a target object via physical interaction with the environment [1]. Recent work has explored the application of mechanical search in structured environments, however, human-centred spaces, such as the home or workplace, generally lack such fixed structure and require more adaptable solutions.

Prior work into making more informed interactions with objects in the environment has evolved from database-driven approaches such as the Columbia Grasp Database [2], [3] to learning-based approaches [4]–[6]. Such work has targeted a range of environments including top-down retrieval of objects in bins [1] and lateral access shelves [6], with grasping being performed by different end-effectors such as parallel jaw grippers [1], [5] or suction grippers [1], [6]. Furthermore, approaches that apply deep reinforcement learning (RL) have targeted generalisability by training the grasping and interaction policies solely as a function of raw sensor inputs such as optical or depth images [7]–[9].

While these existing methods variously consider the confidence of a grasp or the value of a new viewpoint, they lack a framework to explicitly combine and compare between these two action modalities, which is needed when the goal is to find and retrieve a hidden object in clutter. For example, if your keys are under a hat, no viewing action will reveal the keys. Applying mechanical search to environments in which a target object is hidden needs to involve the possibility of both changing the robot's viewpoint and removing blocking objects from the scene. In both instances, the goal is to expose occluded parts of the scene where the target object may be hiding, bearing in mind that if the size of the target object is known, we can already discard some occluded regions from the set of potential hiding places. In this case, the reduction in the number of possible target locations given an object removal action or a viewpoint change action can be cast as a form of information gain.

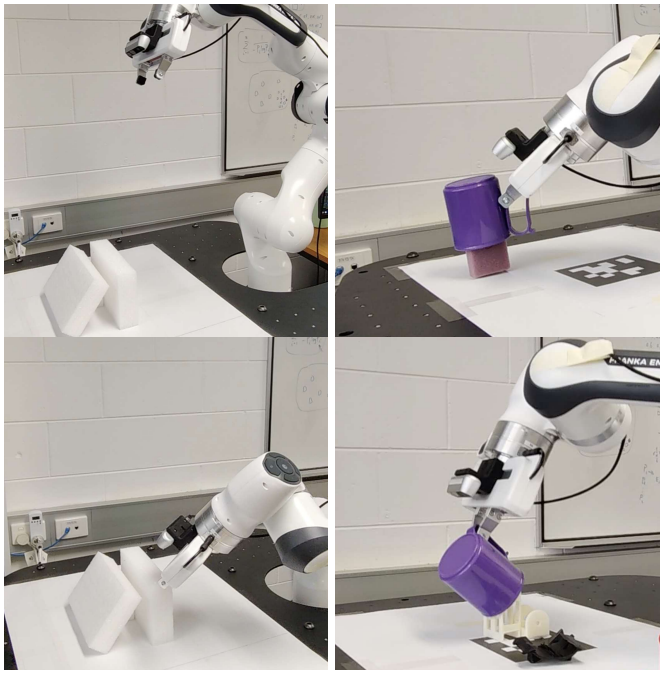


Fig. 1: Interactive search combines novel viewpoints and removal of occluding objects, such as the purple cup, to find and retrieve hidden objects in clutter.

In this work, we present an RL framework for active search and grasp in cluttered environments. The proposed framework comprises four main components: target object location estimation, a state encoding method for efficient depth sensor data processing, action generation to give the robot a set of possible viewing and grasping actions, and an RL agent for information gain optimisation over the generated action set. Our method continuously integrates depth images from a wrist-mounted camera into a truncated signed distance field (TSDF) representation, which we directly exploit to identify potential target locations given a known target object size. To maintain tractability, the volumetric scene information is first compressed through an encoder to generate a lower-dimensional representation that is then passed as part of the state to the RL agent. Action generation is handled by the Active Grasp [4] and Volumetric Grasping Network

[1] School of Electrical Engineering and Computer Science, The University of Queensland, Brisbane 4072, Australia, t.pitcher@uq.net.au, jenjen.chung@uq.edu.au

[2] Autonomous Systems Lab, ETH Zürich, Zürich 8092, Switzerland {fjulian}@ethz.ch

(VGN) [5] implementations from our prior work. We trained the RL policy in simulated PyBullet [10] scenarios where, when interacting with the environment, the robot reasons whether to move occluding objects or change its wrist-mounted camera viewpoint. These actions have trade-offs that are formulated via the RL reward function.

We evaluated our active search pipeline in randomly generated PyBullet scenes where the target object is initially hidden from view. This includes a mix of scenes where the target may be behind other objects or completely covered by another object such as a mug (see Figure 1). Our results show that our approach is able to retrieve the target in all scenarios and on average is 22% faster than state-of-the-art baselines. We also validated the approach using a Franka Emika Research 3 arm to demonstrate sim-to-real transfer of the learned policy.

II. Related Work

Retrieving fully occluded objects in clutter requires robots to explore and interact with their environment [11]. The proposed method involves synthesizing an overarching decision framework to select between next-best-view planning, mechanical search, and grasp planning.

A. Information Gain

Many prior works select actions based on some quantified information gain metric. This has been applied to tasks like next-best-view decision making [4], [12], as well as object segmentation [13]–[15]. In particular, [14] uses an online implementation that continuously samples the environment such that if a better action arises during the motion sequence it can transition seamlessly. This is similar to our proposed method where actions are continuously queried as more of the environment is revealed. However, unlike previous work, our method offers a way to compare the potential benefits of moving the camera or removing an occluding object from the scene in order to find and retrieve a hidden target object.

B. Action Planners

Action planning can be framed as selecting between a set of skills, for example moving the camera, pushing an object or grasping an object. [16] proposed a method for finding hidden objects using either push or grasp actions where action values were proposed by two separate pushing and grasping networks. In [17], the pushing action was formulated from a depth map used to recognize whether objects may be stacked, thereby occluding the target below. Recent work has also looked into revealing objects on shelves where an overhead view is not possible [6]. Similar to the previous two papers, [6] trains separate neural networks for the different actions, however, here the actions are whether to destack objects or stack them in an effort to reveal a target object behind. This work assumes a fixed-size environment in the form of lateral access shelves with limited object variability as all cuboid objects must be positioned with a face aligned normal to that of the rear wall.

C. Learning-based Methods for Mechanical Search

There have been recent extensions to the large body of work in learning-based approaches for grasping and manipulation towards solving mechanical search problems [1], [18]. These works propose approaches that combine visual perception and robotic manipulation to recover a target object. The visual perception elements reason about which occluding objects need to be removed in cases where the target object is initially seen [1] or unseen [18], while Dex-Net 2.0 [19] is queried to synthesise grasps on different parts of the scene. In contrast to our approach, these works have a fixed camera setup and use the image segmentation and object recognition confidence scores to estimate the location of the target. In our work, we exploit the viewpoint changes available with a wrist-mounted camera to extend the search capabilities into full 3D.

III. Active Search

Our proposed RL framework for learning an active search policy is shown in Fig. 2. The state information primarily consists of an encoding of the observed 3D scene alongside the robot arm's configuration. The action space is populated by a predefined set of camera viewpoints as well as the set of grasps output from our grasp synthesis network (VGN [5]). The interactive search policy is then trained to evaluate each possible action based on the potential information gain and the time required for execution.

A. Encoded State Representation

We integrate the stream of depth images from the wrist-mounted camera into a TSDF where each element of the resulting $40 \times 40 \times 40$ matrix represents the distance to the nearest observed surface. As the target object must be specified as part of the task, we assume that a volumetric bounding box is directly available or that one can be readily derived e.g. from a point cloud or mesh of the object. We can therefore combine this knowledge with the current TSDF to generate an accurate map of all the hidden voxels the target could occupy, i.e. in contiguous occluded (negative) parts of the TSDF that are large enough to contain the object. By exploiting the parallelism of a graphical processing unit (GPU), we are able to conduct this search extremely efficiently. An example of this evaluation can be seen in Fig. 3. The resultant mask of potential target locations is then appended to the TSDF as a second channel.

To enable tractable learning, we compress the scene representation through the encoder head of an autoencoder that was pretrained offline on randomised scenes. The encoder reduces the size of the volumetric scene information from $2 \times (40 \times 40 \times 40) = 128000$ to a vector of 512. The robot arm's 7 joint angles are then appended to this vector as well as the action's pose, defined in section III-B, to form the full state representation used for learning.

B. Action Generation

Two types of actions are available to the robot, *view* actions define a camera pose, while *grasp* actions define

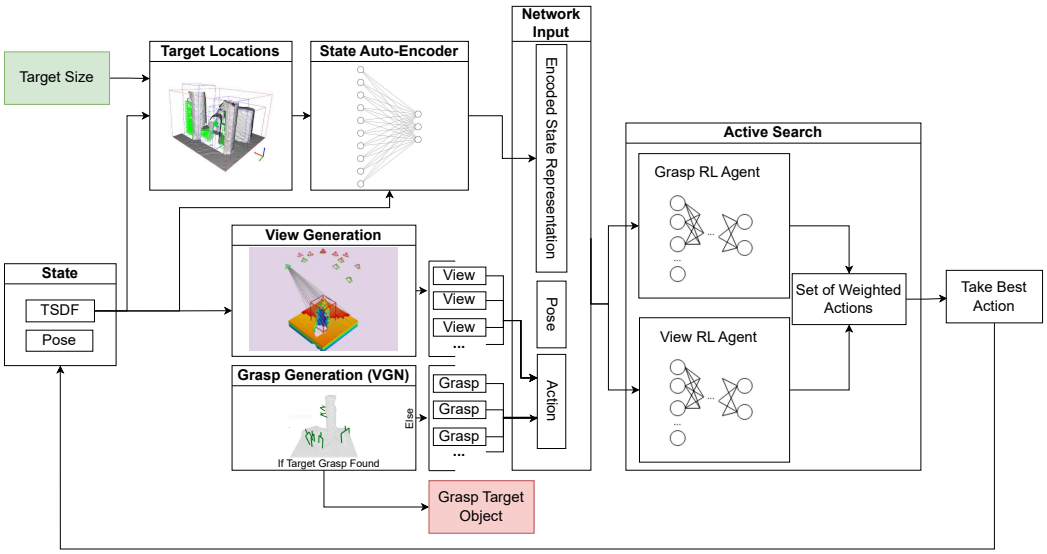


Fig. 2: Our active search policy encodes the volumetric scene information, robot pose, and each view or grasp action to evaluate the best action to execute . The network is structured as a Branching Dueling Q-Network [20] where separate view and grasp agents are trained to output the value of each type of action.

an end-effector pose. Both actions are represented by a 7-dimensional vector of position and orientation (quaternion).

1) View Generation: Similar to the method used in [4], the available view actions are pre-defined as a set of 10 poses spawned above the scene and positioned on a hemisphere facing the centre of the task space. An example of the spawned views can be seen in the "View Generation" inset in Fig. 2. The volumetric scene information is updated using all incoming depth images as the robot moves. In contrast to [4], selected view actions are executed to completion, that is, the robot will move the camera to the commanded viewpoint before querying for the next action. In our experience, this resulted in more stable learning compared to training the policy at the TSDF update rate, which typically lead to much noisier reward signals.

2) Grasp Generation: Grasps are synthesised using the volumetric grasping network (VGN) from our previous work [5]. VGN takes as input the current TSDF of the scene and outputs a set of grasps computed over the full volume of the workspace. Due to its fast inference speed, we are able to synthesise grasps in real-time at the TSDF update rate. However, unlike the view generation pipeline, the set of synthesised grasps from VGN is not guaranteed to be a fixed size. Indeed, there may sometimes be no predicted grasps for a particular scene. We handle this variability through the design of our RL network architecture, which we describe in Section III-D.

Grasps are filtered based on whether they are reachable, determined using Trac-IK. To execute grasps, trajectories are planned using MoveIt's RRT-Connect motion planner. During grasp execution, depth image integration is paused to avoid introducing errors in the TSDF, which would arise when the robot moves the objects in the scene. To account for the scene change after a grasp is successfully executed, we reinitialise the TSDF voxels that correspond to the volume of the extracted object.

C. Information Gain Reward Function

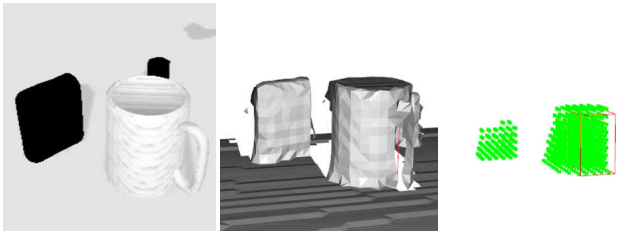
The goal is to train an active search policy that allows the robot to find and retrieve the target object in minimum time. Therefore, the RL reward function (1) trades off between the execution time and the amount of newly revealed space generated by each action. In particular, we are only interested in revealing unobserved voxels where the target object may be hidden (see Fig. 3), and so we compute the information gain component of the reward function as the relative change in possible target locations:

$$R = \omega \times \frac{\Delta\text{Possible target locations}}{\text{Possible target locations}} - t, \qquad (1)$$

where t is the time taken to perform the chosen action. The first term is multiplied by a scaling factor ω to weight the relative importance of information gain versus execution time. In this work we use a scaling factor of $\omega = 10$. In general, grasp actions take longer to complete, however, they can uncover large parts of the scene that are inaccessible to viewing actions especially later in task execution. It's worth noting that the reward function (1) doesn't explicitly reward retrieving the target object. Target grasp success would only offer a very sparse learning signal compared to the information gain, and by default the downstream grasp execution is instructed to attempt any valid grasp that is found on the target. Thus, the primary role of the active search policy is to sufficiently reveal the target object such that a successful grasp can be found.

D. RL Architecture

As described in Section III-B, there are two classes of actions available to the robot, *views* or *grasps*, both of which are represented by 7D vectors of position and orientation in quaternions. Since the number of available grasp actions is not fixed over the states, we structure our learning architecture to evaluate each available action individually.



(a) (b) (c)

Fig. 3: (a) The target object is hidden under the white mug. (b) The TSDF surface reconstruction from the first viewpoint. A small part of the true target object bounding box (red) is also visible in. (c) Initial evaluation of the TSDF to reduces the search volume to the green voxel mask (the true bounding box is also shown). This mask is included as an additional channel to the TSDF encoded in the RL state-space.

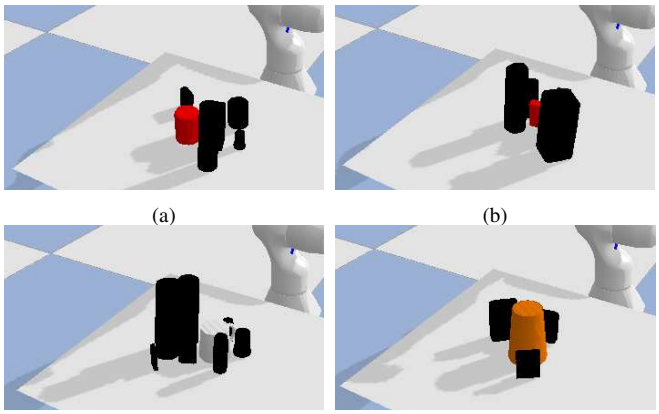
In addition, we learn a separate value network for each class of action. The main motivation for this design choice was to manage the data imbalance between the view and grasp actions, where in general there are fewer grasp actions available, and at the start of learning, view actions tend to be prioritised as they are faster to execute. This architecture can be seen as a Branching Dueling Q-Network [20]. The two value networks share the same lightweight three-hidden-layer MLP architecture, $526 \rightarrow 256 \rightarrow 128 \rightarrow 1$ neuron/s, and take as input the encoded state representation, the robots pose, and the relevant action pose. Each set of actions is batch-processed through their respective network and are first scaled by softmax against their intra-class actions before being compared across the full set of inter-class actions.

E. Training

We trained our Active Search policy in randomised Pybullet [10] simulation scenes using objects from the ShapeNet dataset [21]. We generate two types of scenes for training and testing, *under* and *behind*, which refers to the main type of occlusion in the scene. For the *under* scenes, the simulation spawns a target object at some uniformly random (x,y) location in the robot's workspace with some uniformly random bounded scale to ensure that it is graspable with the robot's end-effector. The occluding object is then randomly sampled from the "mug" or "cup" class of ShapeNet objects and dropped on top of the target object. Occlusion is checked by ensuring the bounding box of the target object is within the bounding box of the occluding object. After this, a random number of decoy objects are spawned around the scene at random locations and with random scale. The *behind* scenes spawn the target object in the same fashion, however, we then spawn a larger object in front of the target to occlude it in the initial camera view followed by some random number of randomly placed *decoy* objects.

Training is completed over 10000 learning epochs with each epoch sampling 5 experiences from a replay memory buffer of the last 100 epochs. Over 1200 unique scenes are used during training, where (similar to [5]) each scene is automatically terminated if:

- The target object is retrieved.
- The robot takes greater than 30 actions.



(a) (b) (c) (d)

Fig. 4: Examples of scenes that are used to test the performance of each implementation. (a)-(b) The target object is hidden behind occluding objects in the initial camera view. (c)-(d) The target object is fully covered by another object, which must be removed before the target can be retrieved.

- There are greater than 4 failed grasp attempts in total.

IV. Experimental Setup

A. Simulation Experiments

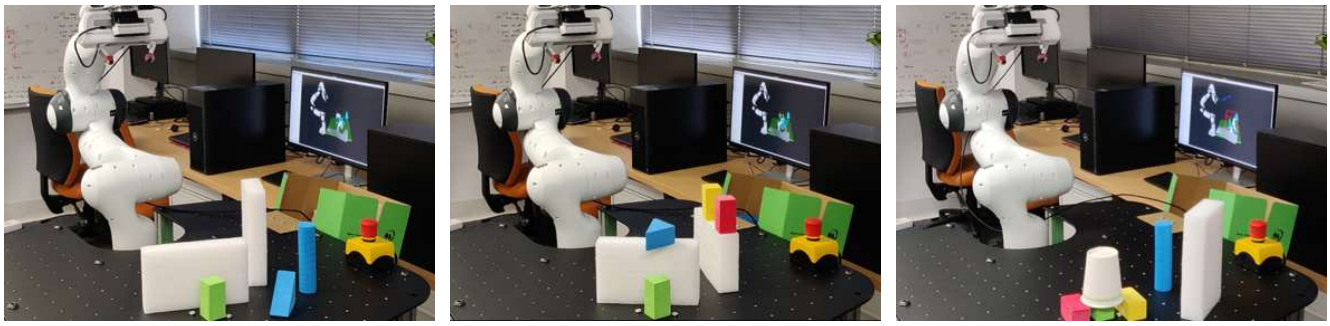
We evaluated our Active Search policy in 22 simulation test scenes and compared its performance to running pure VGN and a baseline heuristic search method. VGN is setup to execute the highest quality grasp, part of the network output, and returns to a fixed top-down camera view between each grasp. The top-down viewpoint is selected to ensure that the full workspace is within the camera frame. The baseline heuristic, which we call Baseline 50/50, computes the information gain of each potential view or grasp action according to the same voxel-based calculation we use in (1). However, it has an equal probability of selecting either the highest value view or grasp action to execute. Compared to VGN and Active Search, Baseline 50/50 is a stochastic policy[1], therefore, in the following experiments, we conducted 15 statistical trials of Baseline 50/50 for each test scene and recorded the best-performing run.

The test scenes are generated using the same procedure as the training environments but include objects and configurations not seen during training. Of the 22 test scenes, 7 are configured with the target object *behind* occluding objects and 15 where the target object is positioned *under* an occluding object (see Fig. 4 for two examples of each type of test scene). All three policies have the same termination behaviour whereby if a valid grasp is found on the target object, the robot will be commanded to execute that grasp. An episode terminates under the same three conditions as described in Section III-E.

B. Hardware Experiments

We demonstrate the sim-to-real transfer of Active Search by executing the learned policy on a Franka Emika Research 3 robot arm in the three real-world test scenes shown in

[1]The only source of stochasticity in VGN and Active Search arises from the MoveIt RRT connect planner used to execute the commanded grasps.



(a) (b) (c)

Fig. 5: Real-world test scenes. The robot is shown in its initial pose and the target object in all scenes is the green prism.

Fig. 5. The three scenes were designed to vary in complexity. In scenes (a) and (b), the target object (the green prism) is occluded from the initial camera view by a large white block. In addition, in scene (b), there is a blue triangular block that partially occludes the target from top-down views and prevents the robot from grasping the target directly from above. For both of these scenes, it is possible for the robot to grasp the target object without removing any other objects from the scene. However, it is considerably more challenging to do so in scene (b) as the robot would first need to reconstruct the far side of the object for VGN to return a reachable and collision-free grasp. In scene (c), the target is covered by a cup, which must first be removed before the green block can be retrieved. Other distractor objects of varying size are also placed in the scene.

When an object is removed from the scene, the TSDF voxels corresponding to the object must be reset. In simulation we use the known object bounding box to identify the relevant voxels that need updating; however, in the real world, we generally do not have known object bounding boxes. Thus, we applied a conservative estimate and reset all voxels within a 20×20 column around the grasp pose, where the top of the column starts 10 voxels above the grasp point and extends all the way down to the workspace surface. In future, we plan to integrate a multi-object volumetric segmentation pipeline such as [22] to automate this procedure.

V. Results and Analysis

A. Simulation Experiments

Performance of VGN, Baseline 50/50 and Active Search over the 22 simulation test scenes was measured using:

- **Success rate:** Where success is defined as the robot retrieving the target object without exceeding 30 actions or 4 total grasp failures.
- **Average completion time:** Time taken to retrieve the target object averaged over all scenes.

Table I shows the results of these experiments. Active Search achieves the highest success rate at 100% across both scene types, along with this it also achieves the fastest object retrieval performance. The Baseline 50/50 and VGN implementations are able to solve all of the *behind* scenes, however, the performance of both methods drops in the *under* scenarios, with Baseline 50/50 completing 13 of the 15 tasks while VGN is only able to retrieve the hidden object in 9 of the cases. Interestingly, VGN is faster than Baseline 50/50 at retrieving the object in *behind* cases, but this result is reversed for the *under* scenarios. Overall, however, Active Search is faster than both other methods in all cases, achieving task completion times that are up to 33.6% (*behind*), 24.3% (*under*) and 22.0% (combined) faster than VGN or Baseline 50/50.

Across all tested scenarios it was observed that Active Search was able to solve complex cluttered scenes in fewer moves than its counterparts, thus allowing it to retrieve the target object faster. A prime example of this was test case 7 shown in Fig. 4c. Active Search's policy for this particular scene was 5 consecutive grasps allowing it to complete the scene in 86.75s as opposed to 135.8s and 163.6s from the Baseline 50/50 and VGN, respectively. Baseline 50/50 performed the action sequence V-G-G-V-G-V-G-G-V-G, where V is a view and G is a grasp, as this scene is highly cluttered there are many grasps needed to de-clutter. The solution found by Active Search uses only 5 total grasps instead of the 6 used by the Baseline. As VGN has no concept of information gain it results in removing all 8 decoy objects from the scene before finally revealing and retrieving the target, resulting in 9 grasps.

B. Hardware Experiments

Experiments in the three real-world test cases were repeated five times for each method to account for slight variations in the scene setup and stochasticity in the tested policy. We report the average task completion time and the number of grasped objects in Table II.

We observe that Active Search is on average 11.8s slower to retrieve the target object in the least cluttered scene (a) where it consistently uses two grasps, the first to remove the large white occluding block and the second to retrieve the object. In 3 of the 5 runs, Baseline 50/50 was able to complete this scene in a single grasp by first selecting an advantageous view, which it is able to compute from its ray-casting method adapted from [4]. Each action taken by VGN returns to a top-down view of the scene, which is also an advantageous position for this specific scene. Thus, although it also takes executes two grasps, it is able to retrieve the target on average 6.6s faster than Active Search.

The following two scenes (b) and (c) present substantially more clutter, and in the case of (c), requires removing another object before the target object can be reached.

TABLE I: AGGREGATED RESULTS FROM SIMULATION EXPERIMENTS OVER 22 SCENES. THE TIMING RESULTS SHOW MEAN AND ONE STANDARD DEVIATION OVER THE SUCCESSFUL RUNS.

Method	**Behind** (7)		**Under** (15)		**All** (22)	
	Success rate	Time (s)	Success rate	Time (s)	Success rate	Time (s)
VGN [5]	100%	59.30 ± 50.93	60.0%	100.04 ± 35.96	72.7%	84.31 ± 47.40
Baseline 50/50	100%	66.80 ± 66.40	86.7%	80.63 ± 37.57	90.9%	75.79 ± 48.26
Active search	100%	44.33 ± 28.30	100%	75.78 ± 31.40	100%	65.77 ± 33.33

TABLE II: RESULTS FROM REAL-WORLD CASE STUDIES.

Scene	**Method**	**Time (s)**	**Number of grasped objects (min/mode/max)**
(a)	VGN [5]	38.0 ± 4.24	2/2/2
	Baseline 50/50	32.8 ± 15.06	1/1/2
	Active Search	44.6 ± 7.40	2/2/2
(b)	VGN [5]	76.6 ± 25.63	3/3/5
	Baseline 50/50	81.6 ± 20.97	3/4/4
	Active Search	69.8 ± 18.09	2/4/4
(c)	VGN [5]	82.6 ± 19.11	2/2/3
	Baseline 50/50	81.8 ± 23.78	2/3/4
	Active Search	54.2 ± 18.70	2/2/2

Active Search is able to solve both of these scenes faster and with fewer grasps compared to VGN ((b) 6.8s and (c) 28.4s faster) and Baseline 50/50 ((b) 11.8s and (c) 27.6s faster). This is particularly evident in scene (c) where Active Search consistently solves the scene with just two grasps, demonstrating that the learned policy transfers well to real-world search and retrieval tasks in cluttered, complex scenes.

VI. CONCLUSIONS AND FUTURE WORK

In this work, we present Active Search, a Reinforcement Learning agent for reasoning and acting in cluttered environments to retrieve a hidden target object. We showed that it is possible to learn the trade-off between the potential information gain of removing an occluding object or moving the camera viewpoint given the current knowledge of the scene and the robot pose. This work shows considerable gains over conventional algorithmic methods, namely a heuristic baseline approach based only on information gain and a pure object-agnostic grasping approach. We demonstrated that Active Search is directly transferable to real-world scenarios remaining performant and effective on complex search tasks. In future we would like to add a volumetric segmentation pipeline to perform bounding box and pose estimation to improve real-world performance.

REFERENCES

[1] 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 *2019 Int. Conf. on Robot. and Autom.*, 2019, pp. 1614–1621.

[2] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "The Columbia grasp database," in *2009 IEEE Int. Conf. on Robot. and Autom.*, 2009, pp. 1710–1716.

[3] C. Goldfeder, M. Ciocarlie, J. Peretzman, H. Dang, and P. K. Allen, "Data-driven grasping with partial sensor data," in *2009 IEEE/RSJ Int. Conf. on Intell. Robot. and Sys.*, 2009, pp. 1278–1283.

[4] M. Breyer, L. Ott, R. Siegwart, and J. J. Chung, "Closed-loop next-best-view planning for target-driven grasping," in *2022 IEEE/RSJ Int. Conf. on Intell. Robot. and Sys.*, 2022, pp. 1411–1416.

[5] M. Breyer, J. J. Chung, L. Ott, R. Siegwart, and J. Nieto, "Volumetric Grasping Network: Real-time 6 DOF grasp detection in clutter," in *2020 Conf. on Robot Learning*, 2020.

[6] H. Huang, L. Fu, M. Danielczuk, C. M. Kim, Z. Tam, J. Ichnowski, A. Angelova, B. Ichter, and K. Goldberg, "Mechanical search on shelves with efficient stacking and destacking of objects," in *The Int. Symp. of Robot. Res.*, 2022, pp. 205–221.

[7] M. Breyer, F. Furrer, T. Novkovic, R. Siegwart, and J. Nieto, "Comparing task simplifications to learn closed-loop object picking using deep reinforcement learning," *IEEE Robot. and Autom. Lett.*, vol. 4, no. 2, pp. 1549–1556, 2019.

[8] S. Joshi, S. Kumra, and F. Sahin, "Robotic grasping using deep reinforcement learning," in *2020 IEEE 16th Int. Conf. on Autom. Sci. and Eng.*, 2020, pp. 1461–1466.

[9] M. Q. Mohammed, K. L. Chung, and C. S. Chyi, "Review of deep reinforcement learning-based object grasping: Techniques, open challenges, and recommendations," *IEEE Access*, vol. 8, pp. 178 450–178 481, 2020.

[10] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2022.

[11] L. L. Wong, L. P. Kaelbling, and T. Lozano-Pérez, "Manipulation-based active search for occluded objects," in *2013 IEEE Int. Conf. on Robot. and Autom.*, 2013, pp. 2814–2819.

[12] W. Bejjani, W. C. Agboh, M. R. Dogar, and M. Leonetti, "Occlusion-aware search for object retrieval in clutter," in *2021 IEEE/RSJ Int. Conf. on Intell. Robot. and Sys.*, 2021, pp. 4678–4685.

[13] H. Van Hoof, O. Kroemer, and J. Peters, "Probabilistic segmentation and targeted exploration of objects in cluttered environments," *IEEE Transactions on Robot.*, vol. 30, no. 5, pp. 1198–1209, 2014.

[14] T. Patten, M. Zillich, and M. Vincze, "Action selection for interactive object segmentation in clutter," in *2018 IEEE/RSJ Int. Conf. on Intell. Robot. and Sys.*, 2018, pp. 6297–6304.

[15] B. Serhan, H. Pandya, A. Kucukyilmaz, and G. Neumann, "Push-to-see: Learning non-prehensile manipulation to enhance instance segmentation via deep Q-learning," in *2022 Int. Conf. on Robot. and Autom.*, 2022, pp. 1513–1519.

[16] Y. Yang, H. Liang, and C. Choi, "A deep learning approach to grasping the invisible," *IEEE Robot. and Autom. Lett.*, vol. 5, no. 2, pp. 2232–2239, 2020.

[17] T. Novkovic, R. Pautrat, F. Furrer, M. Breyer, R. Siegwart, and J. Nieto, "Object finding in cluttered scenes using interactive perception," in *2020 IEEE Int. Conf. on Robot. and Autom.*, 2020, pp. 8338–8344.

[18] M. Danielczuk, A. Angelova, V. Vanhoucke, and K. Goldberg, "X-Ray: Mechanical search for an occluded object by minimizing support of learned occupancy distributions," in *2020 IEEE/RSJ Int. Conf. on Intell. Robot. and Sys. (IROS)*, 2020, pp. 9577–9584.

[19] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," 2017.

[20] A. Tavakoli, F. Pardo, and P. Kormushev, "Action branching architectures for deep reinforcement learning," in *AAAI Conf. on Artificial Intelligence*, 2018, pp. 4131–4138.

[21] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University, Princeton University, Toyota Technological Institute at Chicago, Tech. Rep., 2015.

[22] M. Grinvald, F. Tombari, R. Siegwart, and J. Nieto, "TSDF++: A multi-object formulation for dynamic object tracking and reconstruction," in *2021 IEEE Int. Conf. on Robot. and Autom.*, 2021, pp. 14 192–14 198.