Explore, Approach, and Terminate: Evaluating Subtasks in Active Visual Object Search Based on Deep Reinforcement Learning

Jan Fabian Schmid, Mikko Lauri, Simone Frintrop

***Abstract*— Searching for objects and distinguishing task-relevant objects from others is a key requirement for service robots. We propose a reinforcement learning solution to the active visual object search problem. Our method successfully learns to explore the environment, to approach the target object, and to decide when to terminate the search as the target object has been found. We demonstrate the efficiency of our solution on a dataset of real-world images collected by a robot. Our approach outperforms state-space planning or other baseline search strategies, reaching a higher success rate in a shorter time. We also study individual subtasks of active visual object search. Although strong baselines exist for the subtasks, our RL solution outperforms them in the overall search task.**

I. INTRODUCTION

Consider a service robot executing a task such as "bring me a red cup". The robot must not only be able to visually distinguish the target object, but also to decide what to do if the cup is not in its visual field of view. As objects may move as a result of actions of humans or other robots, the robot cannot rely on remembering where objects are to solve this search task. The variety of environments in which the robot may be deployed excludes hard-coding a sequence of search locations to visit to find the object. Instead, a search strategy that allows the robot to find objects in any environment is required. As shown in Fig. 1, such a strategy encodes how the robot should explore the environment, approach a potential target object and decide if the target object has been found.

Reinforcement learning (RL) is a suitable method for designing autonomous agents which execute actions to reach a goal [1]. Recent advances in image understanding through deep convolutional neural networks allow high-dimensional image inputs in RL methods for visual object search [2], [3], [4]. Current state-of-the-art RL approaches to visual object search apply simplifications of the visual search problem that make them not directly applicable in a robotics scenario. For example in [2], [3], the search task is considered complete if a location close enough to the target object's location is reached. However, a robot executing a fetching task must not only get close to the object, but also detect the object, and take an action to grasp the object. In such cases, it is crucial to consider active termination of the search task, where the target object must be explicitly declared as found to move to the next phase of the fetch task. Simulated environments are commonly used to train and test RL agents for visual search [2], [4]. For robotics applications, it is important to verify if the methods work with real image data.

Department of Informatics, University of Hamburg, Hamburg, Germany, SchmidJanFabian@gmail.com, {lauri,frintrop}@informatik.uni-hamburg.de.

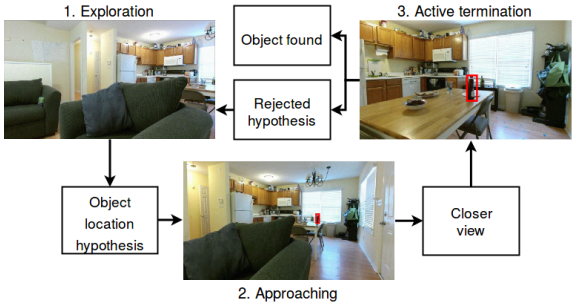

Fig. 1. Active visual object search has three subtasks [7]: exploration to form a target object hypothesis, approaching the hypothesis to collect evidence, and active termination to declare the object is found or to resume exploration. The target object hypothesis is shown by the red bounding box.

We address the aforementioned drawbacks of current RL methods applied to visual object search. Our method extends the deep recurrent Q-network (DRQN) [5] to solve the visual object search problem. Unlike prior work, we explicitly consider the active termination of the search task. We apply an object proposal method to create object hypotheses, learn a strategy for approaching and evidence accumulation, and for explicitly declaring the target object as found. We use real images collected by a robot from the active vision dataset [6].

Our contribution is two-fold. Firstly, we propose the first RL method that addresses all subtasks of visual object search including active termination, and demonstrate its performance with realistic image inputs. Our method generalizes to new object arrangements and search environments, while also outperforming state-space planning and heuristic baseline policies. Secondly, we give empirical evidence on the subtasks of exploration, approaching, and active termination. Although strong baselines exist for the individual subtasks, our integrated RL approach provides a superior solution.

The paper is structured as follows. Section II reviews related work. Section III gives a problem statement and Section IV introduces our RL method. Sections V–VII describe our experiments. Section VIII concludes the paper.

II. RELATED WORK

We review how prior information is applied in visual object search, and then focus on RL methods similar to ours.

A. Prior information in visual object search

Spatial dependency priors are widely applied in visual object search. Garvey [8] introduced the indirect search approach that exploits spatial relations describing the likelihood of finding a target object close to some other objects, or in

a certain type of room. An indirect search agent looking for a cup might exploit a spatial prior that says cups are often found on tables, and search for a table before looking for a cup on the table. In [9], known landmark objects are used to score possible next camera poses. The score is increased if the corresponding view contains landmark objects that the target object likely co-occurs with. Aydemir et al. [10] assume knowledge of the type of room that the target object is likely to occur in. Rooms are categorized by the search agent using a pre-defined probability distribution.

Bayesian methods maintain a probability mass function of each region to contain the target object. Ye and Tsotsos [11] proposed the first complete Bayesian object search framework. Others [12], [13] extended the framework by adding an attention mechanism to further guide the exploration phase.

Some approaches assume complete knowledge of the search environment. In [14] an agent learns to avoid obstacles in a known office environment. The agent moves randomly until the target object is detected and then approaches the detection area. In [15] a map is required to plan a series of views to exhaustively explorate the environment. The target object is assumed to be visible from the initial pose in [7].

B. Reinforcement learning for visual object search

Oh et al. [2] evaluate memory extensions to the DRQN [5], using search environments within the Minecraft video game. The goal of the agent is to reach a target area indicated through task specific visual cues. Zhu et al. [4] employ end-to-end learning for target-driven visual navigation and extend the framework to visual object search in simulated environments [3]. The search agent exhaustively visits all containers which might include the target object. This strategy requires static environments with a fixed set of potential target object positions and is therefore not applicable to our test cases where arrangements of targets and furniture are changed between training and testing. In both approaches above, the target object is considered found if the agent is close to it, i.e., no active termination is required. Some methods target only single subtasks of visual object search. For example, [6], [16] learn RL policies for the approaching task.

We propose the first RL method that addresses all subtasks of visual object search including active termination. We avoid any spatial dependency priors, as argued by [7], such priors have limited usefulness in room-scale search spaces.

III. PROBLEM STATEMENT

Consider a robot searching for a target object o. The robot's pose at time step t is denoted by x_t. To remember information about its past actions and experiences, the robot maintains an internal state vector s_t. The robot can move left, right, forward, or backward; or rotate clockwise or counter-clockwise. Additionally, the action `decl` declares the target object found and terminates the task. After an action a_t, the robot's new pose is $x_{t+1} = f(x_t, a_t)$, where f is a deterministic state transition function.

When the robot enters a new pose x_t, it observes a camera image. A feature vector $y_t = g(x_t, o)$ is extracted, where g represents the imaging and feature extraction process. The internal state is updated via $s_t = h(s_{t-1}, y_t)$. We define g and h in our implementation in the next section.

The objective is to maximize the sum of rewards $R(x_t, a_t)$ over T time steps. All movement actions have a reward of -2. Invalid movement actions, e.g., a movement that would crash the robot into a wall, have a reward of -5. A reward of 50 or -50 is set for declaring the target object correctly or wrongly, respectively. A reward of 10 or -10 is set when the robot enters or leaves, respectively, a pose from which the target object is visible. We address the following problem.

Problem 1 (Active visual object search)**.** *Given the initial pose x_0, internal state vector s_0, target object o, and a budget of $T \geq 1$ actions, find a sequence of policies $\mu_{0:T-1}$, where for $0 \leq t \leq T-1$, $\mu_t : s_t \mapsto a_t$, that maximizes*

$$\begin{aligned} \max_{\mu_{0:T-1}} \quad & \mathbb{E}\left[\sum_{t=0}^{T-1} R(x_t, a_t)\right] \\ \textit{s.t.} \quad & a_t = \mu_t(s_t),\ t = 0, \ldots, T-1 \\ & x_t = f(x_{t-1}, a_{t-1}),\ t = 1, \ldots, T-1 \\ & y_t = g(x_t, o),\ t = 0, \ldots, T-1 \\ & s_t = h(s_{t-1}, y_t),\ t = 1, \ldots, T. \end{aligned}$$

IV. METHOD

We apply reinforcement learning to find a stationary policy that approximates Problem 1. We describe our image preprocessing process in Subsection IV-A. In Subsection IV-B we present our extension of DRQN [5] to object search.

A. Image feature extraction

From the camera view at pose x_t, we extract a feature vector $y_t = g(x_t, o)$. The process is illustrated in Fig. 2 (red part), and its motivation is to extract relevant information about objects and the environment by pre-trained networks. The feature vector $y_t = \begin{bmatrix} y_t^1 & y_t^2 & y_t^3 \end{bmatrix}$ is obtained as follows.

The top branch produces y_t^1 by first applying the region proposal network (RPN) from [17] to extract 300 bounding boxes of object proposals from the input image. The bounding boxes are then fed into the instance classifier of [6] which calculates for each proposal a confidence score for each of the 33 possible classes. Non-maximum suppression is applied to keep only one highest-confidence proposal for each class. We remark that one proposal may be the highest-confidence proposal for multiple classes. A vector of length 5 is provided for each of the 33 object classes: one value representing whether this object class is currently searched for, 3 values representing the position and size of the object proposal it is currently most likely corresponding to, and one element for the confidence value. Finally, the stacked vectors are used as the input for the network and are processed by a fully connected (FC) layer with output size 512 to form y_t^1.

To obtain y_t^2, we extract the highest confidence bounding box detected for the current target object o. The input image is cropped to the contents of this bounding box. The cropped segment is processed by Inception-v3 [18] trained on ILSVRC [19], the output of which is fed into a FC layer with

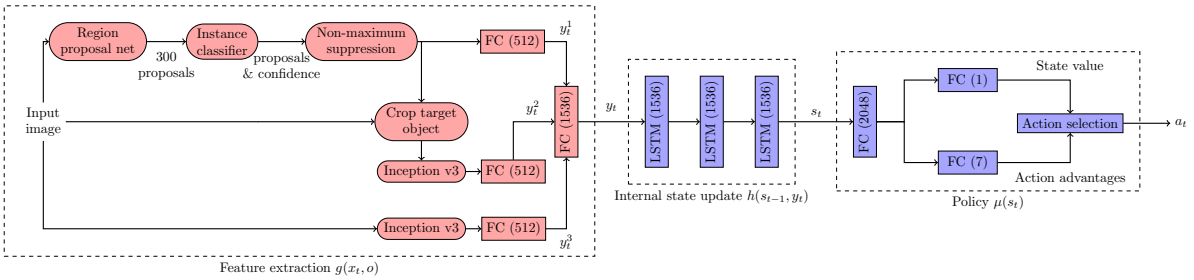

Fig. 2. Overview of the architecture of our RL agent. In red: processing of the current camera view at pose x_t to extract the feature vector $y_t = g(x_t, o)$. In blue: DRQN model with update of internal state $s_t = h(s_{t-1}, y_t)$ and the policy module $a_t = \mu(s_t)$. Rectangular nodes represent the network layers of the core network that are affected by learning. Nodes with rounded corners are fixed to pre-trained weights, or do not have any trainable parameters.

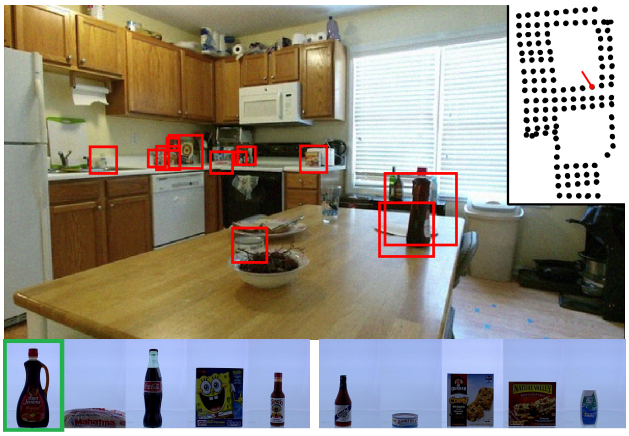

Fig. 3. Snapshot of a search scenario. Object proposals are highlighted with red bounding boxes. In the upper right corner, reachable camera poses are shown with the current pose indicated in red. Possible target objects are shown at the bottom, with the current target object highlighted in green.

512 outputs to form y_t^2. The part y_t^3 is obtained by processing the entire input image by Inception-v3, and processing it by a FC layer with 512 outputs. The outputs of the three initial processing layers are merged and processed by a FC layer to form y_t with size 1536.

B. Extending DRQN to active visual object search

The deep recurrent Q-network (DRQN) [5] is an RL method that extends the deep Q-network [20] with a long-short term memory (LSTM). Through its recurrent structure the DRQN is equipped to deal with spatio-temporal contexts [2], which is an important ability in a partially observable environment such as ours. We apply DRQN to active visual object search as shown in Fig. 2 (blue part).

Given y_t and the old internal state s_{t-1}, a three-layer LSTM module with 1536 cells per layer computes a new internal state s_t. The new internal state is processed by a FC layer with output size 2048. For the policy module $\mu(s_t)$, we employ the dueling DQN approach [21], predicting the state values and action advantages. Finally, the action a_t with the highest predicted value is selected.

V. EXPERIMENTAL SETUP

We test the generalization capability of our RL agent in apartments with up to 3 rooms, considering two scenarios:

1) Known apartment, new object arrangement. The apartment has been used in training, but the arrangement of objects is different at test time.
2) New apartment. Apartment was not used in training.

In both scenarios, the agent can move on a known graph of discretized locations in the apartment, see Fig. 3 (top right).

A. Search environments

We use the active vision dataset (AVD) [6] with scans of 14 real apartments recorded using a robot equipped with a camera, allowing virtual movement through the apartments. We classify each apartment in the AVD as simple, medium, or hard for the visual search task. Simple apartments consist of a single small room. Medium difficulty apartments consist of a large room and sometimes an additional small room, e.g., a bathroom. Hard apartments have multiple large rooms. We use two simple, five medium and five hard apartments for training. Evaluation of Scenario 1 is done with one unseen object arrangement for one apartment of each difficulty. Scenario 2 is tested with one simple and one hard apartment.

B. Target objects and object proposals

Each search environment contains a subset of the 33 BigBIRD objects [22] that we use as target objects. Object instances in each of the categories "hand soap bottle", "snack bar box", and "cereal box" are visually similar. Therefore, we group all instances in these categories, and accept finding any instance as a correct solution to a search task.

The instance classifier is trained with images of target objects which are visually different from how the objects appear in the actual search environment. An example of the training images is shown at the bottom of Fig. 3. The `decl` action proposes the object proposal bounding box with the largest instance classifier certainty as the target object.

To evaluate the effect of object detection, we also replace the RPN object proposals with the ground truth proposals.

C. Comparison methods

As we consider a search task that must be actively terminated, methods such as [2], [4], [3] are not applicable for direct comparison. Instead, our main comparison method is based on state-space planning where a path planning algorithm is guided by the instance classifier. This method modifies the Bayesian reasoning framework [11] following

an approach similar to [10]. The environment is partitioned into a voxel grid. At each voxel, the probability that the target object is located there is maintained. Voxel probabilities are updated after each observation according to the object proposals and their likelihood of corresponding to the target object according to the instance classifier. The target object is declared found if at least 40% of the probability mass has been accumulated in the voxels corresponding to the location of a single object proposal. If the target object is not declared, a movement action is taken. Voxels visible from the adjacent poses are estimated using raytracing and the current reconstruction of the environment. The next pose is the one maximizing the probability of the target object being visible.

We apply two additional baseline methods. The *random* baseline selects a movement or declaration action uniformly at random. The *random + threshold* baseline selects movement actions uniformly at random, but declares the target object found when an object proposal occurs with instance classifier certainty above a threshold of 0.3.

D. Training and implementation details

Our learning procedure is similar to DRQN [5] with a modified replay memory implementation. As the last step of a search episode is the most informative, we sample tail ends of sequences for training. The sequences vary in length, allowing to learn both how to find the target object when it is already close and when the search requires a long sequences of actions. Training batches have a balanced number of successful and unsuccessful episodes. We set a budget of $T = 125$ time steps. During training we add Gaussian noise to the feature vector y_t, and use dropout of entire input sources (y_t^i) for regularization. We use the Adam optimizer with learning rate 10^{-5}. We train three networks each for 72 hours and choose the one with greatest training success rate for evaluation. Our implementation is available online[1].

VI. RESULTS

Table I shows our main result, the search success rate and the average length of successful search episodes. Results are shown for both of the two generalization scenarios described in Section V, either using RPN object proposals or ground truth (GT) object proposals. Results are averaged over 1500 search tasks for the *new object arrangement* scenario and 400 search tasks for the *new apartment* scenario. For the random baselines, each search task is repeated 10 times.

For the more realistic scenario using RPN proposals, our RL agent outperforms all baseline approaches. Compared to using GT proposals, using the more noisy RPN proposals only reduces the performance of our method from 0.696 to 0.609. The success rate of the Bayesian method decreases from 0.475 to 0.172 when using RPN proposals. The difference is even greater for the *random + threshold* baseline. Recall that this baseline declares the target object found when an object proposal with instance classifier certainty above a threshold of 0.3 is encountered. With more false object proposals in the RPN case, the performance degrades from 0.620 to 0.223 compared to using GT proposals. All baseline methods strongly depend on reliable object proposals, whereas our RL solution learns to cope with noisy proposals.

The results for the two scenarios of *new object arrangement* and *new apartment* indicate that our RL agent can generalize both to new object configurations in known environments, as well as to searching for objects in novel environments. Even in novel environments, our RL method consistently outperforms the baselines when using RPN proposals. When using GT object proposals, *random + threshold* has similar success rate as our RL agent. However, the RL agent requires significantly less actions on average to find the target object. This suggests that the *random + threshold* baseline takes many unnecessary steps before encountering a reliable GT proposal that matches the target object. The random method takes about 5 steps on average to find the target, indicating that it is only successful when the target object is close to the robot's starting location.

Table II shows the average success rate and length of successful episodes for the *new object arrangement* scenario, split according to the difficulty of the search environment. With both types of object proposals, and in all difficulties of search environments, our method outperforms the baseline approaches. Generally, the length of successful episodes tends to increase for more difficult environments. When using GT proposals, the success rate of all methods decreases as the environment difficulty increases. However, for RPN proposals the success rates are particularly low for the simple environment. This indicates problems of the RPN specific to this environment. While on average for about 15% of GT proposals corresponding RPN proposals are generated (having an IoU greater than 0.5), this is only for about 10% of GT proposals the case on the simple environment used to evaluate the *new object arrangement* scenario.

VII. WHY DO OBJECT SEARCH METHODS FAIL?

To analyze why active visual object search methods fail, we study the subtasks of exploration, approaching, and active termination. We describe the experimental setup and results, and discuss which subtasks are critical for object search.

A. Experimental setup

Experiments are run in the *new object arrangement* scenario. For exploration and approaching, we use the GT object proposals to exclude the effect of imperfect object proposals.

a) Exploration: The agent starts at a random pose. The exploration subtask is successfully completed when the agent reaches a pose from which the target object is visible. We evaluate this subtask on the same 1500 search tasks from the general search task for the *new object arrangement* scenario. The *random* baseline randomly selects a movement action that leads to a pose that is still unvisited. The *rotating* baseline performs a 360 degree rotation at the starting pose.

b) Approaching: The agent starts at a pose where the target object is visible. Over ten actions, we record if the agent finds the target object or not, and how the instance

[1] https://github.com/JanFabianSchmid/RL_for_AVOS

TABLE I

OBJECT SEARCH SUCCESS RATE AND LENGTH OF SUCCESSFUL EPISODES. BEST SUCCESS IN BOLD.

Scenario	Method	RPN object proposals		Ground truth object proposals	
		Success rate	Length of successful episodes	Success rate	Length of successful episodes
New object arrangement	Ours	**.609**	19.2	**.696**	16.4
	Bayesian	.172	46.2	.475	40.3
	Random	.078	5.2	.133	5.3
	Random + threshold	.223	15.5	.620	26.3
New apartment	Ours	**.555**	36.5	.448	16.9
	Bayesian	.122	51.2	.358	43.3
	Random	.081	4.4	.134	5.0
	Random + threshold	.189	7.9	**.471**	27.4

TABLE II

SEARCH SUCCESS RATE (SR) AND LENGTH OF SUCCESSFUL EPISODES (NEW OBJECT ARRANGEMENT SCENARIO). BEST SUCCESS RATES IN BOLD.

	RPN object proposals						Ground truth object proposals					
	Simple		Medium		Hard		Simple		Medium		Hard	
	SR	Length	SR	Length	SR	Length	SR	Length	SR	Length	SR	Length
Ours	**.540**	12.9	**.722**	20.6	**.564**	23.5	**.774**	12.2	**.726**	16.2	**.588**	22.1
Bayesian	.104	35.2	.212	50.8	.200	46.9	.616	39.5	.484	46.1	.326	33.4
Random	.074	5.6	.091	4.8	.070	5.1	.165	5.5	.136	5.5	.098	4.9
Random + threshold	.195	14.6	.281	17.7	.194	13.3	.682	24.7	.650	26.3	.528	28.4

classifier certainty changes. Our hypothesis is that the higher the classifier certainty, the better the agent can approach the target object. We average over 750 task configurations.We apply a *random* baseline, and a *forward* baseline that approaches the visible object by only executing the action `fwd`. The *active instance classifier* (AIC) baseline [6] is an RL agent trained to approach an area visible in the current image. The area to approach is the object proposal that is currently the most likely one corresponding to the target object.

c) Active termination: The agent starts at a random pose where an object is visible. In 50% of cases the visible object is the target object. In the other 50% of cases the visible object is a non-target object. We execute movement actions that approach the object. While approaching, if the agent's policy module outputs a declaration action, the task terminates. If the agent uses the declaration action when approaching a target object, or if it does not use the declaration action when approaching a non-target object, the task is considered successful. In case of false positive declaration or not declaring the target object, the task is considered unsuccessful. We employ the *certainty threshold* baseline which declares the target object found when an object proposal occurs with instance classifier certainty above a threshold of 0.3. This subtask is evaluated on 750 task configurations.

B. Results

Average success rate in exploration as a function of the number of available steps is shown in Fig. 4. In the first 5 steps our RL agent has similar performance as the rotating policy. We verified by examining the movement patterns that our method starts exploration by looking around. Random navigation reaches a higher success rate than our method, but only after 75 steps of exploration. This partly explains why the *random + threshold* baseline performs well in the overall task when using GT proposals.

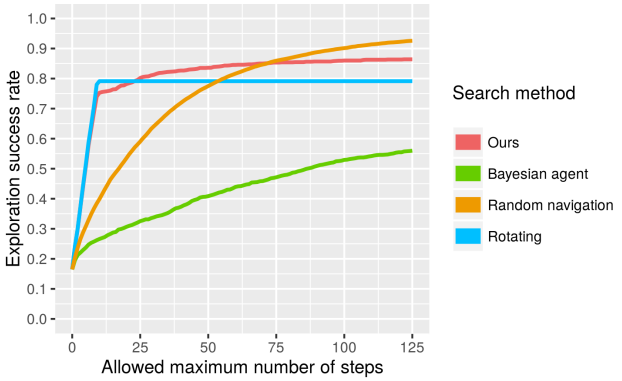

Fig. 4. Exploration success rate as function of the number of actions.

TABLE III

EXPLORATION SUCCESS RATE W.R.T. DIFFICULTY.

Method	Simple	Medium	Hard
Ours	.938	.952	.702
Bayesian	.668	.618	.394
Random nav.	**.990**	**.988**	**.799**
Rotating	.924	.886	.564

Table III shows the exploration success rate after 125 steps grouped by search environment difficulty. In environments with simple and medium difficulty, exploration is not the main challenge of the overall search task. In these cases, about 90% of the target objects are observable by rotating at the initial pose. In hard scenes, success rate in the overall search task is limited by the difficulty of exploration, i.e., getting the target object into the field of view.

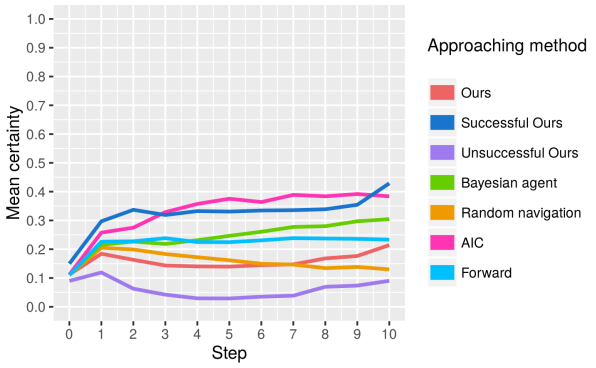

Fig. 5. Instance classifier certainty as function of the number of actions.

TABLE IV
SUCCESS RATE IN ACTIVE TERMINATION.

Object proposals	Ours	Bayesian	Certainty threshold	Random
RPN	**0.741**	0.499	0.585	0.500
Ground truth	0.819	0.708	**0.837**	0.500

Next, we evaluate the approaching subtask. Fig. 5 shows the average instance classifier certainty as a function of number of steps in cases where the initial certainty was below 0.5[2]. We analyze separately the cases where our method successfully declared the correct object hypothesis during the approaching task (Successful ours, 38% of cases), or did not (Unsuccessful ours, 62% of cases). In successful cases, the performance of our method is similar to AIC. In unsuccessful cases, the certainty decreases, and the agent incorrectly decides that the visible object is not the target object and searches elsewhere. Overall, the AIC [6], which is specifically trained for this subtask and does not consider exploring, rather than approaching, performs the best.

Table IV shows the average active termination success rates. Due to our experimental setup, a random policy has expected success rate 0.5 in this subtask. With GT proposals, our method performs similarly as the certainty threshold baseline. This shows that the instance classifier certainty is a suitable indicator for active termination. Our method is robust to noisy RPN object proposals, maintaining a success rate of 0.741 compared to 0.585 for the certainty threshold, and 0.499 for the Bayesian method.

C. Discussion

Our analysis of the subtasks indicates that in hard scenes, exploration is difficult and the most limiting factor for the success rate in object search. We found the instance classifier certainty an appropriate measure for success in the approaching subtask. Active termination of the search task is important in robotics applications. Noisy object proposals strongly decrease the performance of the baseline policies in active termination. Our RL method has learned to cope with noisy proposals and succeeds significantly more often.

[2]With high initial instance classifier certainty, approaching is not necessary, and the task can immediately be terminated. If initial certainty is above 0.5, our RL agent terminates the task immediately (66.1% of the episodes).

VIII. CONCLUSION

Our RL method for active visual object search solves all subtasks of object search: exploration, approach, and active termination. This makes our method well suited for robotics applications where the target object must be explicitly declared found to execute subsequent tasks such as grasping. We showed that our method generalizes both to new arrangements of objects in known search environments, as well as to novel search environments. Our method's ability to dynamically switch between behaviors relevant for the subtasks of search helps it to outperform baselines.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] J. Oh, V. Chockalingam, S. Singh, and H. Lee, "Control of Memory, Active Perception, and Action in Minecraft," in *ICML*, 2016.

[3] Y. Zhu, D. Gordon, E. Kolve, D. Fox, L. Fei-Fei, A. Gupta, R. Mottaghi, and A. Farhadi, "Visual Semantic Planning Using Deep Successor Representations," in *ICCV*, 2017.

[4] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning," in *ICRA*, 2017.

[5] M. Hausknecht and P. Stone, "Deep Recurrent Q-Learning for Partially Observable MDPs," in *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*, 2015.

[6] P. Ammirato, P. Poirson, E. Park, J. Koeck, and A. C. Berg, "A dataset for developing and benchmarking active vision," in *ICRA*, 2017.

[7] S. Mittal, M. S. Karthik, S. Kumar, and K. M. Krishna, "Small Object Discovery and Recognition Using Actively Guided Robot," in *ICPR*, 2014.

[8] T. D. Garvey, "Perceptual Strategies for Purposive Vision," Ph.D. dissertation, Stanford, CA, USA, 1976.

[9] L. Kunze, K. K. Doreswamy, and N. Hawes, "Using Qualitative Spatial Relations for indirect object search," in *ICRA*, 2014.

[10] A. Aydemir, A. Pronobis, M. Gbelbecker, and P. Jensfelt, "Active Visual Object Search in Unknown Environments Using Uncertain Semantics," *IEEE T-RO*, vol. 29, no. 4, pp. 986–1002, 2013.

[11] Y. Ye and J. K. Tsotsos, "Sensor Planning for 3D Object Search," *CVIU*, vol. 73, no. 2, pp. 145 – 168, 1999.

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

[13] A. Rasouli and J. K. Tsotsos, "Visual Saliency Improves Autonomous Visual Search," in *Canadian Conf. on Comp. and Robot Vision*, 2014.

[14] V. A. Shim, M. Yuan, and B. H. Tan, "Automatic object searching by a mobile robot with single RGB-D camera," in *APSIPA ASC*, 2017.

[15] D. G. Lopez, K. Sjo, C. Paul, and P. Jensfelt, "Hybrid laser and vision based object search and localization," in *ICRA*, 2008.

[16] X. Han, H. Liu, F. Sun, and X. Zhang, "Active Object Detection with Multi-Step Action Prediction Using Deep Q-Network," *IEEE Transactions on Industrial Informatics*, 2019.

[17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *NIPS*, 2015.

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *CVPR*, 2016.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *NIPS*, 2012.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari With Deep Reinforcement Learning," in *NIPS Deep Learning Workshop*, 2013.

[21] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling Network Architectures for Deep Reinforcement Learning," in *ICML*, 2016.

[22] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "BigBIRD: A large-scale 3D database of object instances," in *ICRA*, 2014.