Active Object Search

Jie Wu
Sun Yat-sen University
wujie10558@gmail.com

Tianshui Chen*
Sun Yat-sen University
DarkMatter AI
tianshuichen@gmail.com

Lishan Huang
Sun Yat-Sen University
huanglsh6@gmail.com

Hefeng Wu
Sun Yat-sen University
wuhefeng@gmail.com

Guanbin Li
Sun Yat-sen University
liguanbin@mail.sysu.edu.cn

Ling Tian
University of Electronic Science and Technology of China
lingtian@uestc.edu.cn

Liang Lin
Sun Yat-sen University
DarkMatter AI
linliang@ieee.org

ABSTRACT

In this work, we investigate an Active Object Search (AOS) task that is not explicitly addressed in the literature. It aims to actively perform as few action steps as possible to search and locate the target object in a 3D indoor scene. Different from classic object detection that passively receives visual information, this task encourages an intelligent agent to perform active search via reasonable action planning; thus it can better recall the target objects, especially for the challenging situations that the target is far from the agent, blocked by an obstacle and out of view. To handle this AOS task, we formulate a reinforcement learning framework that consists of a 3D object detector, a state controller and a cross-modal action planner to work cooperatively to find out the target object with minimal action steps. During training, we design a novel cost-sensitive active search reward that penalizes inaccurate object search and redundant action steps. To evaluate this novel task, we construct an Active Object Search (AOS) benchmark that contains 5,845 samples from 30 diverse indoor scenes. We conduct extensive qualitative and quantitative evaluations on this benchmark to demonstrate the effectiveness of the proposed approach and analyze the key factors that contribute more to address this task.

KEYWORDS

Active Object Search, Reinforcement learning, Cost-sensitive active search reward, Cross-modal action planner

*Tianshui Chen is the corresponding author.

MM '20, October 12–16, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3413571

ACM Reference Format:
Jie Wu, Tianshui Chen, Lishan Huang, Hefeng Wu, Guanbin Li, Ling Tian, and Liang Lin. 2020. Active Object Search. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.* ACM, Seattle, USA, 9 pages. https://doi.org/10.1145/3394171.3413571

1 INTRODUCTION

It is our long-standing goal to build intelligent agents that can actively process the multi-modal environment state and accomplish specified tasks, such as office service and life support. Even a small solid forward can make an energetic change and great convenience to our lives [11, 17]. For the object detection/recognition tasks [6, 8, 30, 36], however, most existing methods remain stuck in the passive vision way, although great success has been achieved due to the advance of deep neural networks. These methods assume that scene images are captured by preset cameras with the target objects somewhat properly displayed (e.g., the large-scale ImageNet [13] and COCO [23] datasets). Therefore, they can hardly locate the target objects in highly challenging real-world scenarios, e.g., the target objects are far from the camera, blocked by an obstacle, or out of view (see Figure 1). In contrast, human can effortlessly deal with such situations by actively performing a series of reasonable actions.

These observations motivate us to empower this human ability for intelligent agents. Hence we explore a new multimedia task, i,e, Active Object Search (AOS), where an agent is randomly placed and given a target query (object) in the scene, and it actively performs as few action steps as possible to locate the target object. An example of how the agent finds out an out-of-view target object based on our framework is presented in Figure 2. Compared with traditional object detection/recognition, this task poses three additional challenges: 1) the agent should establish a reliable cross-modal association that involves interactions between visual information and target query; 2) the agent requires an in-depth understanding about the environments, thus that it can explore and interact with the environments; 3) the agent needs to learn a policy that can adaptively plan a series of reasonable actions and should be able to generalize across different environments.

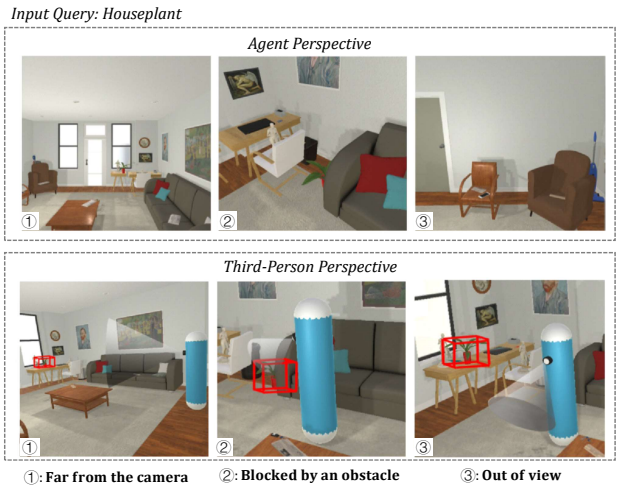

Figure 1: Some challenging cases that the target object is far from the agent, blocked by an obstacle and out of view. We present the scene images captured from the agent perspective (the first row) and from the third-person perspective (the second row).

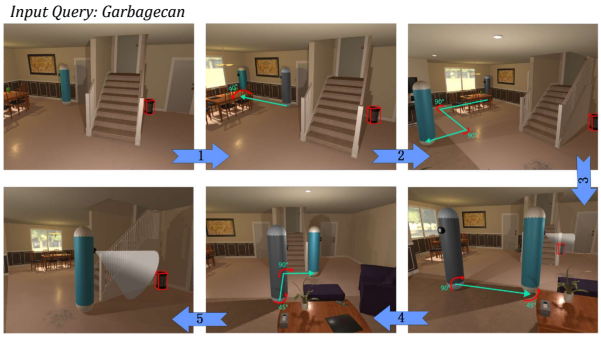

Figure 2: An example of how the agent finds the target object (i.e., GarbageCan) via active search.

Inspired by recent progress in active perception-based cross-modal tasks, e.g., vision-and-language navigation [3] and embodied question answering [11], we utilize reinforcement learning approaches to address the aforementioned challenges and develop a feasible solution to address the proposed AOS task. Specifically, our framework consists of three modules, i.e., a 3D detector to locate the target object, a state controller to decide whether to stop, and a cross-modal action planner to process cross-modal concepts and infer a series of actions. Compared with existing frameworks introduced by recent tasks that explore active vision [3, 11, 17, 18, 20, 40, 41], our framework features three differences: (i) We build a 3D object detector explicitly, which is more suitable for agents to interact with the environment; (ii) A state controller is specially designed, instead of adding the stop signal to the action space as existing works, and experiments show that our design can achieve impressive performance gain; (iii) We define a cost-sensitive active search reward that enables the agent to find out the target object with minimal action steps. Moreover, it is also worth noting that our AOS task is a rather fundamental one when compared with past similar works as mentioned above and can be used to assist in fulfilling these more high-level tasks.

To evaluate this task, we construct a new benchmark that allows an agent to interact with the environment. AI2-THOR [21] is a newly-raised simulation platform that contains various type of daily objects, complex and diverse rooms, and enables a virtual agent to interact with the environment. In this work, we increase the room diversity of AI2-THOR by providing more varied object instantiations, color types, surface patterns, and layouts, and build an Active Object Search (AOS) benchmark that contains 5,845 samples from 30 diverse indoor scenes. The benchmark is divided into three subsets, i.e., a training set (3,991 samples), a validation set (996 samples), and a test set (888 samples). Each sample contains a random seed that determines the room layout, object appearance and object position, which in fact is able to greatly enrich the number and diversity of the samples in the AOS benchmark.

The contributions of this work are summarized into four folds: 1) We present a new task, active object search, that mimics human to actively search a target query (object) via reasonable action planning. 2) We design a modulized framework that enables the agent to plan, detect, and control in the environment to address this task, and our framework introduces new characteristics in the design of action space, object detector, stop control and reward when compared with existing works. 3) A large-scale Active Object Search (AOS) dataset based on the AI2-THOR simulation platform is built for training and evaluation. We rebuild the AI2-THOR platform in the aspects of room layout and object randomness to increase its diversity. 4) In-depth analyses are conducted on the AOS benchmark to demonstrate the effectiveness of the proposed framework over some baseline and competitive methods and discuss the key factors that contribute more to handle this task.

2 RELATED WORK

Object Search. Conventional object localization [28, 34] and detection [9, 16, 24, 33] tasks have achieved substantial progress in recent years. Beyond heavy focus on 2D object detection, recent demand eagerly calls for promoting 3D object understanding in the realistic scenarios [10, 35, 39]. Qi *et al.* [29] took advantage of mature 2D object detector [30] and designed the frustum PointNets to achieve the state-of-the-art performance in 3D object detection tasks on both indoor [32] and outdoor [14] environment based datasets. However, the above-mentioned tasks are still performed in static images, where the target objects are generally in a suitable perspective and moderate size, and even situated in the middle of the image. Such setting is quite far from the real scenarios. Take service robots for example, the agent is commonly spawned with a random state in an indoor environment, and it often cannot immediately detect the target. Thus the agent should learn to reason about the next action based on its active perception of the environment.
Active Perception based Cross-modal Task. Research on active perception has been around for a long time, since the 1980s. The

essence of active perception is to set up a goal based on some current belief about the world and to put in motion the actions that may achieve it [1]. The idea of active perception is frequently incorporated in some cross-modal tasks for action planning [3, 5, 11, 17, 18, 20, 22, 25, 40, 41]. Abhishek *et al.* [11] presented the EmbodiedQA task, which requires the agent to actively navigate in the environment and gather crucial visual information to answer questions. Gordon *et al.* [17] proposed an interactive question answering (IQA) task that requires the agent to interact with the dynamic visual environment and actively plans a series of actions conditioned on the questions. Different from the above tasks, the proposed AOS task is a more fundamental task, which requires the agent to follow the instructions to infer action to locate the target with a suitable viewpoint. AOS can be incorporated as an essential sub-task for improving the performance of other high-level tasks, such as robot navigation and grabbing.

Reinforcement Learning. Recently, deep reinforcement learning (RL) methods have been employed to learn policies in an end-to-end manner to predict a series of actions [7, 38]. Zhu *et al.* [40] adopted the successor representation [12] to design a hybrid approach of model-based and model-free RL. They proposed to train the model with imitation learning and fine-tune with RL, which enables the agent to perform more realistic tasks and offer significant benefits for transfer learning to new tasks. Gordon [17] used hierarchical reinforcement learning based model to operate at multiple levels of temporal abstraction. Zhu *et al.* [41] proposed a deep siamese actor-critic model to learn to navigate. This method generalizes well to new target goals. RL technique enables the agent to learn the optimal action policy function through trial and error interactions with the environment, hence it has been widely used in sequence-level tasks.

3 ACTIVE OBJECT SEARCH (AOS) BENCHMARK

3.1 Environment Setup

To evaluate the AOS task, it requires to collect samples that allow an agent to interact with the environment. Due to the problems of costs, scalability, and research reproducibility, it is impractical to obtain samples in real-world settings. Thus, we resort to AI2-THOR [21], a simulation platform that provides various photo-realistic indoor environments. AI2-THOR contains 120 rooms from 4 environments, i.e., living room, kitchen, bedroom, bathroom. Although these rooms have diverse layouts, the objects are always similar in color, shape, surface, and position. In order to collect more diverse samples, we modify the AI2-THOR platform from the following aspects: 1) Increase the diversity of object color, shape and surface pattern to achieve richer **object appearance**; 2) Increase the randomness of object coordinates to increase the diversity of **object position**; 3) Randomly move some structured object (e.g., sofa, table) to increase the diversity of **room layout**. The agent in this platform is equipped with a camera at a fixed height, which can capture RGB-D data and can perform 8 actions, i.e., *move ahead 25cm*, *move back 25cm*, *move 25cm left*, *move 25cm right*, *rotate 45 degrees left*, *rotate 45 degrees right*, *look up 30 degrees*, *look down 30 degrees*.

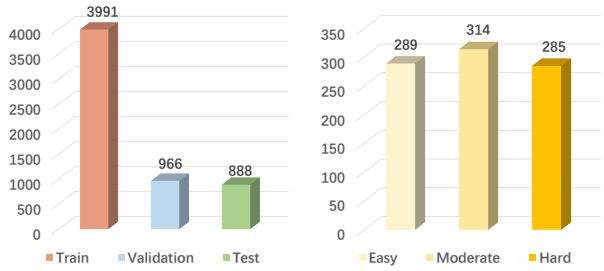

Figure 3: Sample distribution of the training, validation, and test subsets of the AOS dataset (left) and sample distribution of the easy, moderate, and hard on the test subset (right).

3.2 AOS Dataset

AOS dataset contains 5,845 samples from 30 living rooms of AI-THOR platform. Each sample contains a target object (query) and a random seed that determines the object appearance, object position, and room layout. The random seed also determines the ground truth 3D bounding box of the target objects. In this work, we focus on the living rooms as these rooms are large enough to perform searching. We consider 6 common object categories in the living rooms as the target query, i.e., *Box*, *Newspaper*, *GarbageCan*, *WateringCan*, *HousePlant* and *Statue*.

We divide the dataset into training, validation, and test subsets, in which the training sub-set contains 3,991 samples from 20 rooms, the validation sub-set is comprised of 966 samples from another 5 rooms, and the test sub-set consists 888 samples from the rest 5 rooms. Note that there are no room overlap in different sub-sets. Thus, the dataset can be used to evaluate the generalization performance to the unseen scenes. For each sample from the training set, the agent is spawned in the scene with a random position, rotation, and angle at each training iteration. Such a random strategy can produce a large number of exploratory situations and encourage the agent to learn a more robust policy. For each sample from the validation or test sets, the position, rotation, and angle of the agent are fixed. To provide more comprehensive evaluations, we further divide the test set into three parts according to their difficulty: 1) easy samples: The target object is within or near the field of view of the agent, and the agent can search the object successfully within a few simple actions. 2) moderate samples: The target object is out of view of the agent, and the agent needs to make a few actions to locate the object successfully. 3) Hard samples: The target object is far from the agent and the agent should make a series of correct actions to finish the task. The sample distribution over these three difficulty levels is presented in Figure 3.

4 LEARNING FRAMEWORK

4.1 Task Formulation

We formulate the proposed AOS task as a standard reinforcement learning task where an agent is required to actively locate the target accurately with the minimum number of actions. The agent is spawned at a random position in the environment and given a target object o at the beginning of each episode during training. At

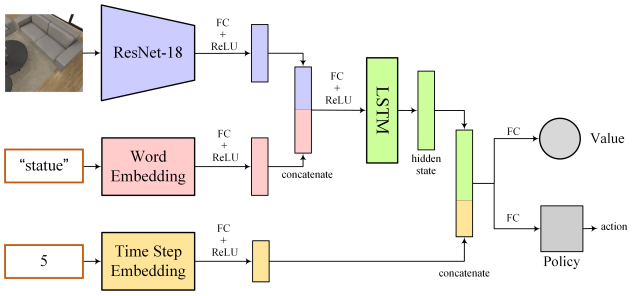

Figure 4: An overall pipeline of the cross-modal action planner. At each time step, the planner takes the image feature of the current scene, the embedded feature of the target object, and the current time step as input to predict the next action.

each time step t, the agent receives the RGB-D images (I_t, D_t) via a first-person egocentric vision sensor. Subsequently, an action a_t is drawn from the agent's policy function distribution.

In the AOS task, the agent aims to search the target and predict its 3D bounding box accurately. Specifically, the agent will predict a 3D box when the target is sensed in the current frame. When the 3D IoU between the predicted and ground-truth 3D bounding boxes is higher than 0.25, the task is considered to be successfully accomplished.

4.2 Model

To address this task, we present a reinforcement learning framework that enables the agent to plan, detect, and control. The proposed framework consists of three basic modules: an cross-modal action planner, a pre-trained 3D object detector and a state controller. The cross-modal planner jointly models the vision and language concepts to infer a reliable action. The 3D object detector predicts the 3D bounding box of the target object, and the state controller allows the agent to know when to stop.

3D Object Detector. Frustum PointNets [29] is utilized as the 3D object detector in our work, which takes the RGB-D images as input to predict the 3D bounding boxes. We first employ Faster-RCNN [30] as the 2D object detector to predict objects' 2D bounding boxes as well as their categories. The target candidates' predicted boxes are then used to generate frustum proposals and extract their frustum point clouds. The frustum point clouds are consecutively fed into 3D instance segmentation PointNet, T-Net and box estimation PointNet to perform the object segmentation, bounding box regression tasks and predict the corresponding 3D bounding boxes. We compare the predicted bounding boxes with the ground truth ones and compute the 3D intersection-over-union (IoU), which will be applied to provide the active search reward in the training stage and evaluate the search performance during testing.

State Controller. Previous related works [17, 41] assume that the agent is notified by the environment on whether it has finished the task. However, this setting hardly holds in actual scenarios. In the training stage, the agent can leverage the ground-truth information to decide whether the agent finishes the task, but the agent should determine when to stop by itself during testing. Hence, we design a state controller that enables the agent to judge the correct stop time. In the pre-trained 3D object detector, the box parameters are estimated by the global feature from the box estimation PointNet. Hence we cascade two fully-connected layers after the global feature to output a predicted IoU U^p_ϑ of the 3D bounding box. ϑ denotes the parameters of the controller. U^p_ϑ can be considered as the confidence of whether an agent successfully locates the target. When U^p_ϑ is higher than 0.25, the controller will emit the "*stop*" signal. Otherwise, the controller will emit the "*continue*" signal.

Cross-modal Action Planner. The cross-modal action planner is designed to reason the next action based on the RGB image I_t, the target query o. The architecture details are depicted in Figure 4. The backbone of the visual branch is ResNet-18 [19] pre-trained on ImageNet. The visual representation, word embedding and time step embedding is mapped into a 512-d, 512-d and 32-d feature respectively by a fully-connected layer and the ReLU activation function. The time step embedding is a simple lookup table that stores embeddings of a fixed dictionary. This module is often used to store time step embeddings and retrieve them using indices. The image features and word embedding are concatenated to create a joint representation. The LSTM layer is designed specifically to enable the agent to incorporate the memory information about the previous state (seen scene), which is beneficial for the agent to develop high-level abstractions and lead to a more generalizable model. The hidden state of LSTM is concatenated to the embedding representation of the time step to estimate the policy function $\pi_\theta(a_t|s_t)$ and the value approximator $V_\theta(s_t)$. θ denotes the model parameters. a_t and s_t denote the predicted action and the state respectively. $\pi_\theta(a_t|s_t)$ computes the probability distribution over possible movements in action space and $V_\theta(s_t)$ computes a scalar estimate of the agent value function for optimization. The representation of the time step is introduced to stabilize value prediction.

An example of how the agent locates the target in the inference time is shown in Figure 5. When the controller emits the "*continue*" signal, the framework will switch to the planner. When the controller emits the "*stop*" signal, the searching processing will be considered to be finished. The whole active object searching process in the inference time is described by Algorithm 1.

4.3 Cost-sensitive Active Reward

The planner is trained by reinforcement learning algorithm and thus the reward setting is crucial for optimization. Considering that AOS task is highly related to the searching results and the episode cost, we define the reward function from two aspects to encourage the agent to find out the target actively and quickly. The first reward is active search reward, which is essential to encourage the agent to learn to locate the target actively. Concretely, active search reward is closely related to the 3D IOU, i.e., U^G_t obtained from the pre-trained 3D object detector module. When U^G_t is higher than 0.25, the target object is considered to be found out and the planner will receive an active search reward greater than zero:

$$R^a_t = \begin{cases} (U^G_t/0.5) + 0.5 & U^G_t > 0.25 \\ 0 & U^G_t \leq 0.25; t < T_{max} \\ -1 & U^G_t \leq 0.25; t = T_{max} \end{cases}, \quad (1)$$

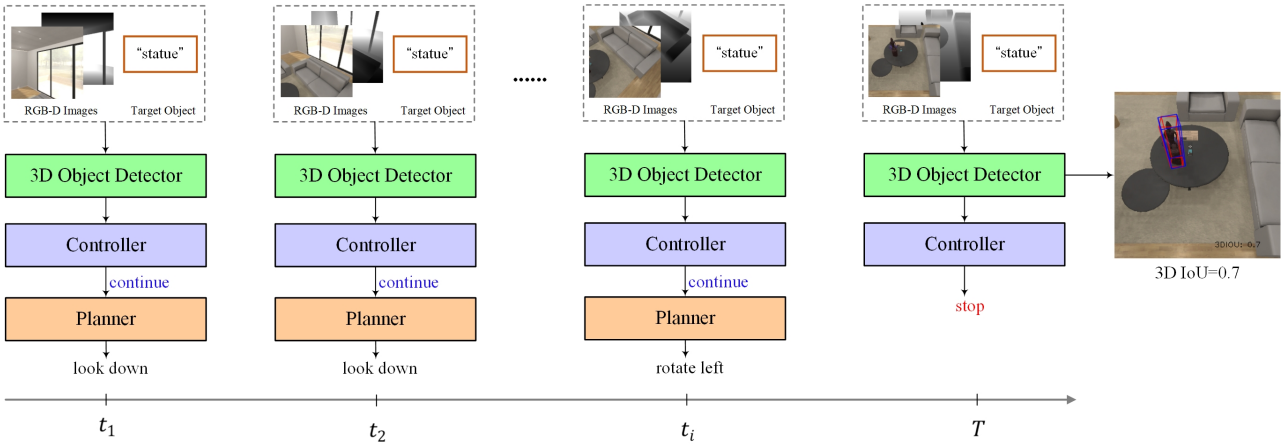

Figure 5: The pipeline of the proposed framework. The agent actively explores the environment to search the target in an iterative manner. During each iteration, a 3D detector first detects the target based on the current observation, and a controller then predicts whether the target is successfully located. If the controller outputs a *continue* signal, the planner reasons an action to continue searching. If the controller outputs a *stop* signal, the agent stops searching and outputs the final result.

Algorithm 1 Active Object Searching Process in the Inference Time

Input: target query o
for each $t \in [1, T_{max}]$ **do**
  Obtain I_t, D_t;
  Predict the 3D bounding box and get the controller output U_{ϑ}^{p};
  if $U_{\vartheta}^{p} > 0.25$ **then**
    The controller emits the "*stop*" signal;
    Evaluate the performance of this episode;
    break;
  else
    The controller emits the "*continue*" signal;
    The planner proposes a action: $a_t \sim \pi_\theta(s_t)$;
    Update the state of agent;
  end if
end for

In this way, the action sequences that can search the target successfully will be explicitly encouraged, and the planner learns to actively choose the corresponding action when it encounters similar scenes. Furthermore, the agent is penalized a terminated constant (-1) when it does not find the target and has reached the maximum episode length T_{max}. In this paper, T_{max} is fixed to 100.

Suppose that the planner only employs the active search reward, the agent is likely to search the target by walking through the whole room with unrestrained actions. It contradicts the design objectives of the AOS task. This task also requires the agent to be sensitive to the episode cost and learn to locate the target quickly. So we provide the cost-sensitive reward R_c as a small punishment for each action to constrain the agent to be sensitive to the movement costs. R_c encourages the agent to search the target with fewer actions.

The total reward r_t received at the time step t is obtained by:

$$r_t = R_t^a + \lambda R_c. \tag{2}$$

λ is introduced to achieve a better trade-off between detection results and movement costs, ensuring two types of reward can collaborate with each other mutually for the proposed task.

4.4 Optimization

Optimization for Controller. In order to obtain an accurate "*stop*" signal from the controller, the optimization process for the controller is designed as a regression task. Namely, the controller outputs a predicted value $U_{\vartheta m}^{p}$ for 3D IOU instead of a binary signal. In our work, we use the smooth-L1 loss [15] for training the controller:

$$\mathcal{L}(\vartheta) = \frac{1}{M}\sum_{m=1}^{M} \text{smooth}_{\text{L1}}(U_m^G, U_{\vartheta m}^{p}), \tag{3}$$

M is the number of training samples.

Optimization for Planner. We employ the advantage actor-critic (A2C) policy gradient algorithm to maximize the objective of the policy network $\mathcal{J}_\pi(\theta)$, formulated as:

$$\mathcal{J}_\pi(\theta) = \mathbb{E}_\pi[\sum_{t=0}^{T} A^\pi(s_t, a_t)\log\pi_\theta(a_t|s_t)], \tag{4}$$

where $A^\pi(s_t, a_t)$ denotes the advantage function. In this paper, we use the Generalized Advantage Estimation (GAE) [31] to achieve a bias-variance tradeoff in the setting of policy gradient. The advantage function is estimated by the k-step returns with function approximation:

$$A^\pi(s_t, a_t) = \sum_{l=0}^{k-1} \gamma^l r_{t+l} + \gamma^k V_\theta(s_{t+k}) - V_\theta(s_t), \tag{5}$$

where γ denotes the discount factor. Furthermore, we add the negative entropy of the policy for improving exploration as described

Table 1: The performance results of naive detector, random policy and our proposed CSASR method.

Method	Easy		Moderate		Hard	
	Recall	MIoU	Recall	MIoU	Recall	MIoU
ND	2.77	1.71	0	0	0	0
RA	60.90	35.72	35.67	22.46	22.46	14.42
CSASR	**79.58**	**48.01**	**66.56**	**42.09**	**50.53**	**32.50**

Table 2: The performance comparison of random policy, two active-related methods and our proposed approach.

Method	R@10	MIoU@10	MEL@10	R@20	MIoU@20	MEL@20	R@30	MIoU@30	MEL@30
RA	21.29	12.70	8.79	31.87	19.20	16.96	39.64	24.20	22.59
Nav [26]	22.36	15.87	8.23	46.72	28.77	15.43	63.22	39.32	18.26
DQN[4]	24.89	17.31	7.97	48.49	29.33	15.17	63.92	39.18	18.01
TD-RL [41]	26.62	18.03	7.85	50.27	30.14	14.38	64.88	40.11	17.93
DNN-RL [2]	29.71	19.16	7.62	52.39	31.04	13.89	65.69	**41.33**	17.65
CMP [18]	32.68	20.37	7.41	54.03	**32.45**	13.07	**65.79**	40.82	17.49
CSASR	**34.12**	**21.43**	**7.23**	**54.17**	32.22	**12.81**	65.65	40.94	**17.40**

in [27]. The overall policy loss function is defined as:

$$\mathcal{L}(\theta) = -\mathcal{J}_{\pi}(\theta) - \alpha H(\pi_{\theta}(a_t|s_t)), \tag{6}$$

where $H(\pi_{\theta}(a_t|s_t))$ is the entropy of π_{θ} and the parameter α controls the strength of entropy regularization term.

The value function $V_{\theta}(s_t)$ provides an estimation of the expected sum of rewards: $V_{\theta}(s_t) = \mathbb{E}_{\pi}[r_t]$. In order to train the value network, we perform temporal difference updates [27] to minimize the squared difference between $V_{\theta}(s_t)$ and the bootstrapped k-step returns $\hat{r_t} = \sum_{l=0}^{k-1} \gamma^l r_{t+l}$, and minimize the loss:

$$\mathcal{L}_{\mathcal{V}}(\theta) = \beta||\hat{r_t} - V_{\theta}(s_t)||^2, \tag{7}$$

where β is a constant coefficient.

5 EXPERIMENTS

5.1 Experiment setting

Implementation details. We implement the 3D object detector with a Faster-RCNN [30] to predict 2D location and Frustum PointNets [29] to further estimate 3D bounding box, and we train this module similar to [29]. The controller is implemented by two fully-connected layers that take global feature of Frustum PointNets as input to predict whether to stop. During training, we compute the Intersection over Union (IoU) score between the predicted 3D bounding box and ground truth one. If the IoU is higher than 0.25, the sample is used as a positive sample. If the IoU is lower than 0.25, and it is used as a negative sample otherwise. Furthermore, the resampling strategy is leveraged to balance the positive and negative samples. The cost-sensitive reward R_c is set to -1. The discount factor γ is 0.99. The hyperparameters α and β are empirically set to 0.01 and 0.5, receptively.

Evaluation metrics. We conduct experiments with the maximum episode lengths T of 10, 20 and 30, respectively, and design the following metrics for evaluation. 1) **Recall@T** (shorted as R@T (in %)) is the proportion of the successfully searched samples when a "stop" is predicted or the maximum episode lengths T is reached. 2) **Mean IoU@T** (shorted as MIoU@T (in %)) denotes the mean of the IoUs between the estimated 3D bounding box and corresponding ground truth ones with the maximum episode lengths of T. The IoU is set to 0 if the target object is not successfully detected. 3) **Mean Episode Length@T** (shorted as MEL@T) denotes the mean of episode lengths for all samples with the maximum episode lengths of T. The length is set to T if the target object is not successfully detected.

5.2 Experiment Results

As there exists limited literature on this problem, we first design two baseline methods to demonstrate the effectiveness of the active search mechanism: 1) *Naive detector (ND)*. We directly use the 3D object detector to estimate the 3D bounding box statically according to the current scene; 2) *Random policy (RA)*. We replace the learned policy with a randomly selected mechanism, with others remained. We present the result in Table 1. For our methods (CSASR) and *Random policy*, we use the maximum episode lengths of 30 for comparisons. As can be seen, the *naive detector* baseline cannot find out any target object in the middle & hard setting. Even in the easy setting, it can hardly locate the target object successfully, i.e., a recall rate of 2.77%. The *Random policy* baseline can better find the target object via random exploration about the environment, which achieves recalls of 60.90%, 35.67%, and 22.46% in the easy, middle and hard settings, respectively. By learning policy to actively explore the environment, our methods boost the recalls to 79.58%, 66.56%, and 50.53%, with a relative improvement of 30.8%, 86.6%, and 125.0% compared with the *Random policy* baseline in three settings. This experiment well proves the significance of active search mechanism.

In order to further demonstrate the effectiveness of CSASR, we implement several competitive active perception based methods to address the proposed AOS task. 1) DQN [4] proposes a dynamic attention action strategy joint with the Deep-Q-Network based reinforcement learning algorithm to localize an object. 2) Nav [26] designs the reward that corresponds to whether the agent can reach a goal from a random start location and orientation. 3) TD-RL [41] fuses the current observation and target scene into the generic

Table 3: The performance comparison of different reward settings in terms of R@T, MIoU@T and MEL@T.

Method	R@10	MIoU@10	MEL@10	R@20	MIoU@20	MEL@20	R@30	MIoU@30	MEL@30
ASR	31.67	19.71	8.40	48.46	29.82	13.97	60.97	38.41	18.36
CSASR (λ=1×10^{-4})	32.36	20.19	8.07	52.07	30.64	13.38	63.01	39.32	18.46
CSASR (λ=1×10^{-3})	**34.12**	**21.43**	**7.23**	**54.17**	**32.22**	**12.81**	**65.65**	**40.94**	**17.40**
CSASR (λ=1×10^{-2})	33.86	21.05	7.93	51.53	31.28	13.06	64.23	40.56	18.11
CSASR (λ=1×10^{-1})	28.80	18.88	9.14	40.88	26.94	14.86	50.79	32.28	20.33

Table 4: The performance comparison of different learning methods to predict the termination time.

Method	R@30	MIoU@30	MEL@30	Error Rate (%)
NC-CSASR	7.27	4.96	28.79	46.41
CSASR	65.65	40.94	17.40	10.09
GT-CSASR	68.76	44.23	15.89	0

siamese layers to reason the action. 4) DNN-RL [2] establishs a baseline for object instance detection and adopts a deep learning-based system for next best view selection. 5) CMP [18] designs a unified joint architecture for mapping and planning, which constructs a top-down map of the environment and applies a differentiable neural net planner to infer the next action.

The performance results in the proposed AOS dataset are shown in Table 2. As shown, the previous leading methods are DNN-RL [2] and CMP [18], which obtain R@10 of 29.71 and 32.68, respectively. Our approach outperforms these competitors on almost all metrics, e.g., improving the R@10 score to 34.12. For the proposed task, Nav and TD-RL manage to encourage the agent to navigate to the target object, achieving better observation effect at close range. Furthermore, the above algorithms rely heavily on the episode lengths to achieve better detection results, while CSASR can obtain better results with fewer steps.

5.3 Ablative Study

Analysis of Cost-Sensitive Reward. As discussed above, we introduce a cost-sensitive reward to penalize redundant action steps. To validate the effectiveness of the cost-sensitive reward item, we design a baseline method (ASR) that simply removes the cost-sensitive reward and re-trains the framework. As shown in Table 3, we find removing this reward leads to an obvious performance drop. Specifically, for the setting of T=20, it reduces the recall and MIoU by 5.71% and 2.40%, and increase MEL by 1.16.

To further analyze the cost-sensitive reward, we conduct experiments by setting the trade-off factor λ to 1×10^{-1}, 1×10^{-2}, 1×10^{-3}, and 1×10^{-4}, respectively. The results are presented in Table 3. We find that setting λ to a low value (i.e., 1×10^{-4}) leads to better performance than that is without this reward term, and increasing λ to a large value (i.e., 1×10^{-3}) can promote the performance. However, when further increasing λ, the performance suffers from a severe drop. For example, the R@20, MIoU@20 drop from 54.17% and 32.22% to 40.88% and 26.94% if increasing λ from 1×10^{-3} to 1×10^{-1}. The possible reasons are two aspects: 1) using a low reward cannot well penalize redundant step; 2) using a high reward may suppress the active search reward. Thus, we set λ as 1×10^{-3}.

Analysis of controller. The agent should stop searching if it has accurately located the target objects during the test stage. Since the ground truth stop signal are not provided in the testing procedure, we need to predict a signal that indicates whether the agent should stop. Our method introduces a controller for this end, while some previous works [11, 37] include *stop* signal as an additional action (NC-CSASR). As shown in Table 4, the agent cannot well predict whether to stop if including *stop* signal as an additional action, and thus it is of less impressive performance. By introducing the controller, our method can accurately predict the *stop* signal, and significantly boost the performance, e.g., improving the recall@30 from 7.27% to 65.65%, MIoU from 4.96% to 40.94%, and reducing the MEL@30 from 28.79 to 17.40. Furthermore, the error rate of the stop signal decreases by a sizeable margin, i.e., from 46.41% to 10.09%.

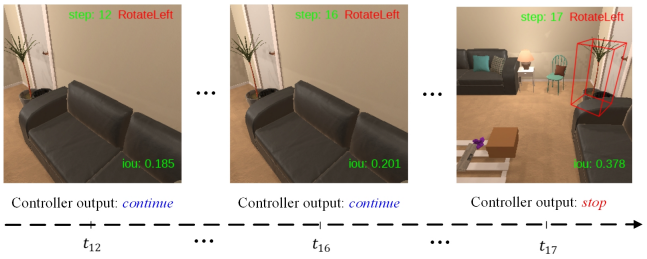

Figure 6: The visualization of how the agent determines a correct termination time by the controller.

We also present an example to illustrate the situation when the agent stops searching in Figure 6. As shown, although the target object is in view at timestep t_{12} and t_{16}, the controller still predicts *continue*, since the agent has low confidence for accurately locating the target object. At timestep t_{17}, the agent locates the target object accurately with a high degree of confidence, and it outputs a *stop* signal to terminate the searching process.

Some other works [17, 41] assume that the agent receives a *stop* signal from the environment if locating the target object accurately. To further analyze the effect of the controller, we replace the *stop* signal predicted by the controller with this ground truth *stop* signal (GT-CSASR), and present the results in Table 4. Although it can improve the performance by a certain level, it is impractical as

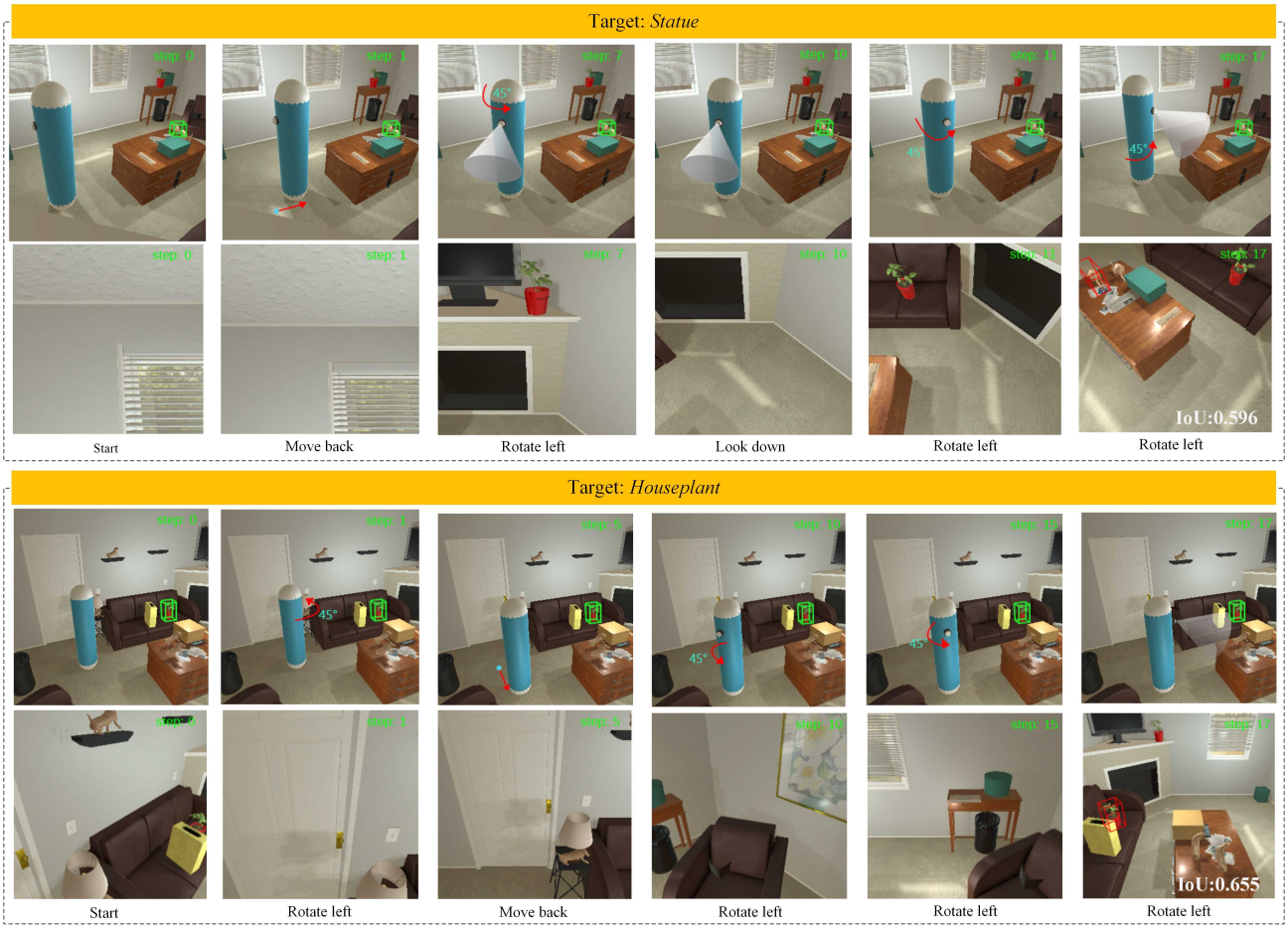

Figure 7: Visualization of the searching process. At each step, we visualize the performed action from the three-person perspective and perceived scene from the agent perspective. For a simple illustration, we only visualize some key steps.

the agent does not know the ground truth information in the real-world setting. By simultaneously considering performance and practicability, we introduce the controller to automatically predict when to stop searching.

5.4 Visualization of Active Object Search

We visualize some examples of the searching process in Figure 7. As shown, the agent performs actions that change its position and pose to search the target object when the objects are out of view (the first example) or blocked by obstacles (the second example). For the second example, as most of the target object is blocked by the obstacles, it cannot be found by simply rotation. Thus, the agent first steps to a proper position and then rotate to an appropriate angle, and finally locate the object successfully.

6 CONCLUSIONS

Human can actively collect multi-modal information and explore the environment to perceive the scene in-depth, and it is a long-term goal to mimic this ability for intelligent robots. As an early attempt toward this goal, this work explores active object search, a new task that requires an intelligent agent to perform as few actions as possible to find out a target query (object). To handle this task, we propose a reinforcement learning framework that actively explores the environment to find out the target object with minimum actions. To achieve this, we design a reward that simultaneously penalizes inaccurate object search and redundant action steps to train the framework. To evaluate this task, we construct an active object search benchmark based on the AI2-THOR environment and conduct extensive experiments and analyses on this benchmark to analyze the key factors that contribute more to this task.

ACKNOWLEDGMENTS

This work was supported in part by National Natural Science Foundation of China (NSFC) under Grant No. 61836012 and 61876045, in part by National High Level Talents Special Support Plan (Ten Thousand Talents Program), in part by the Natural Science Foundation of Guangdong Province under Grant No. 2017A030312006, and in part by Zhujiang Science and Technology New Star Project of Guangzhou under Grant No. 201906010057.

REFERENCES

[1] John Aloimonos. 1990. Purposive and qualitative active vision. In *[1990] Proceedings. 10th International Conference on Pattern Recognition*, Vol. 1. IEEE, 346–360.

[2] Phil Ammirato, Patrick Poirson, Eunbyung Park, Jana Košecká, and Alexander C Berg. 2017. A dataset for developing and benchmarking active vision. In *2017 IEEE International Conference on Robotics and Automation*. IEEE, 1378–1385.

[3] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2.

[4] Juan C Caicedo and Svetlana Lazebnik. 2015. Active object localization with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*. 2488–2496.

[5] Tianshui Chen, Riquan Chen, Lin Nie, Xiaonan Luo, Xiaobai Liu, and Liang Lin. 2018. Neural task planning with and–or graph representations. *IEEE Transactions on Multimedia* 21, 4 (2018), 1022–1034.

[6] Tianshui Chen, Liang Lin, Xian Wu, Nong Xiao, and Xiaonan Luo. 2018. Learning to segment object candidates via recursive neural networks. *IEEE Transactions on Image Processing* 27, 12 (2018), 5827–5839.

[7] Tianshui Chen, Zhouxia Wang, Guanbin Li, and Liang Lin. 2018. Recurrent attentional reinforcement learning for multi-label image recognition. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[8] Tianshui Chen, Muxin Xu, Xiaolu Hui, Hefeng Wu, and Liang Lin. 2019. Learning semantic-specific graph representation for multi-label image recognition. In *Proceedings of the IEEE International Conference on Computer Vision*. 522–531.

[9] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. 2017. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1907–1915.

[10] Zhenfang Chen, Zhanghui Kuang, Wayne Zhang, and Kwan-Yee K Wong. 2019. Learning Local Similarity with Spatial Relations for Object Retrieval. In *Proceedings of the 27th ACM International Conference on Multimedia*. 1703–1711.

[11] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 5. 6.

[12] Peter Dayan. 1993. Improving generalization for temporal difference learning: The successor representation. *Neural Computation* 5, 4 (1993), 613–624.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 248–255.

[14] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 3354–3361.

[15] Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 1440–1448.

[16] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 580–587.

[17] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2018. IQA: Visual question answering in interactive environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4089–4098.

[18] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. 2017. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2616–2625.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[20] Felix Hill, Karl Moritz Hermann, Phil Blunsom, and Stephen Clark. 2017. Understanding grounded language learning agents. *arXiv preprint arXiv:1710.09867* (2017).

[21] Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. 2017. AI2-THOR: An interactive 3d environment for visual AI. *arXiv preprint arXiv:1712.05474* (2017).

[22] Liang Lin, Lili Huang, Tianshui Chen, Yukang Gan, and Hui Cheng. 2017. Knowledge-guided recurrent neural network learning for task-oriented action prediction. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 625–630.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision*. Springer, 21–37.

[25] Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. Listen, Attend, and Walk: Neural Mapping of Navigational Instructions to Action Sequences.. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 1. 2.

[26] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. 2016. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673* (2016).

[27] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. 1928–1937.

[28] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. 2015. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 685–694.

[29] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. 2018. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 918–927.

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.

[31] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).

[32] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. 2015. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 567–576.

[33] Shuran Song and Jianxiong Xiao. 2016. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 808–816.

[34] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. 2015. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 648–656.

[35] Yizhou Wang, Yen-Ting Huang, and Jenq-Neng Hwang. 2019. Monocular Visual Object 3D Localization in Road Scenes. In *Proceedings of the 27th ACM International Conference on Multimedia*. 917–925.

[36] Zhouxia Wang, Tianshui Chen, Guanbin Li, Ruijia Xu, and Liang Lin. 2017. Multi-label image recognition by recurrently discovering attentional regions. In *Proceedings of the IEEE international conference on computer vision*. 464–472.

[37] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. 2018. Visual semantic navigation using scene priors. *arXiv preprint arXiv:1810.06543* (2018).

[38] Tong Yu, Yilin Shen, Ruiyi Zhang, Xiangyu Zeng, and Hongxia Jin. 2019. Vision-language recommendation via attribute augmented multimodal reinforcement learning. In *Proceedings of the 27th ACM International Conference on Multimedia*. 39–47.

[39] Heyu Zhou, An-An Liu, and Weizhi Nie. 2019. Dual-level Embedding Alignment Network for 2D Image-Based 3D Object Retrieval. In *Proceedings of the 27th ACM International Conference on Multimedia*. 1667–1675.

[40] Yuke Zhu, Daniel Gordon, Eric Kolve, Dieter Fox, Li Fei-Fei, Abhinav Gupta, Roozbeh Mottaghi, and Ali Farhadi. 2017. Visual semantic planning using deep successor representations. In *Proceedings of the IEEE International Conference on Computer Vision*. 483–492.

[41] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. 2017. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation*. IEEE, 3357–3364.