Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A deep Q-learning network based active object detection model with a novel training algorithm for service robots*

Shaopeng LIU†, Guohui TIAN†‡, Yongcheng CUI, Xuyang SHAO

School of Control Science and Engineering, Shandong University, Jinan 250061, China

†E-mail: shaopeng.liu66@mail.sdu.edu.cn; g.h.tian@sdu.edu.cn

Received Mar. 20, 2022; Revision accepted July 29, 2022; Crosschecked Aug. 16, 2022; Published online Sept. 24, 2022

Abstract: This paper focuses on the problem of active object detection (AOD). AOD is important for service robots to complete tasks in the family environment, and leads robots to approach the target object by taking appropriate moving actions. Most of the current AOD methods are based on reinforcement learning with low training efficiency and testing accuracy. Therefore, an AOD model based on a deep Q-learning network (DQN) with a novel training algorithm is proposed in this paper. The DQN model is designed to fit the Q-values of various actions, and includes state space, feature extraction, and a multilayer perceptron. In contrast to existing research, a novel training algorithm based on memory is designed for the proposed DQN model to improve training efficiency and testing accuracy. In addition, a method of generating the end state is presented to judge when to stop the AOD task during the training process. Sufficient comparison experiments and ablation studies are performed based on an AOD dataset, proving that the presented method has better performance than the comparable methods and that the proposed training algorithm is more effective than the raw training algorithm.

Key words: Active object detection; Deep Q-learning network; Training method; Service robots
https://doi.org/10.1631/FITEE.2200109 **CLC number:** TP242

1 Introduction

The family service robot's main function is to provide various services for its users (Shuai and Chen, 2019), and in recent years, it has received much attention from the research community. When performing service tasks in a family environment, for example, finding a service-related object (Wang et al., 2019; Liu et al., 2022b), the service robot needs to not only perceive the environment but also detect the target object. The purpose of object detection is to recognize and locate objects.

With the development of deep learning, the accuracy of object detection has been greatly improved by state-of-the-art object detection methods (Pu et al., 2021). In recent years, anchor-free networks like ExtremeNet (Zhou et al., 2019) and CenterNet (Duan et al., 2019) are popular due to their advantages of concerning small object detection. The development and success of object detection in the field of computer vision have provided new solutions for detection tasks in robotics. A vision system using the YOLO algorithm has been proposed to detect objects that can be obstacles in the path of a mobile robot (Dos Reis et al., 2019). Wan and Goudos (2020) employed Faster R-CNN for multi-class fruit detection for a harvesting robot in the field of smart farming.

In robotic applications, the detection task is executed over multiple images using active motion control of the robot to obtain an optimal viewpoint and approach the target object for further manipulation, which is called robot active object detection (AOD) (Ammirato et al., 2017). With the guidance

‡ Corresponding author
* Project supported by the National Natural Science Foundation of China (Nos. U1813215 and 62273203)
ORCID: Shaopeng LIU, https://orcid.org/0000-0002-9624-846X; Guohui TIAN, https://orcid.org/0000-0001-8332-3064

of AOD, the robot can make appropriate movements (e.g., forward and turning left) to get close to the target object for better detection. It is worth noting that AOD is based on object detection, because the detection result of each image is the necessary basis for AOD to take the next action.

In Zhang et al. (2017), viewpoint control strategies have been presented for active object recognition that is quite similar to an AOD task. With the development of reinforcement learning (RL) techniques, RL-based AOD methods have become popular. Paletta and Pinz (2000) first introduced RL in the context of AOD using Q-learning. Recent works about AOD have focused on deep RL (DRL) approaches (Ammirato et al., 2017; Han et al., 2019). Ammirato et al. (2018) used a DRL method to train a network to take an action at each step and built an AOD dataset. Han et al. (2019) first introduced the deep Q-learning network (DQN) with a dueling architecture for robot AOD to predict multiple actions at each step. Nevertheless, the mentioned works trained DQN models based on the basic training algorithm, which can lead to low training efficiency and testing accuracy. At the same time, designing an appropriate state space and model structure can enhance the performance of robot AOD.

In this study, a DQN-based AOD model with a novel training algorithm is proposed for service robots. The DQN-based model is designed, including state space, feature extraction, and a multilayer perceptron (MLP). In the state space, the RGB image and the bounding box of the target object are selected as the state information to reflect the environmental changes that occur after the robot moves. Feature extraction contains two types of extractors for different types of state information. The convolutional neural network (CNN) feature extractor is composed mainly of 16 residual blocks (He et al., 2016) to generate a feature vector of the RGB image. The location feature extractor normalizes the values of the bounding box to produce a location feature vector. Two kinds of feature vectors are combined as a fusion feature to be input into the MLP. The three-layer perceptron fits the Q-values of different actions. Meanwhile, a novel training algorithm based on memory (TAM) is designed to speed up the exploration with high efficiency during the training and to increase the testing accuracy. The proposed training algorithm can generate more training data with positive rewards and prevent the model from repeatedly learning the data with negative rewards. In addition, when to stop exploration during training is a key problem. To solve this problem, the end state is used to judge whether the robot has arrived at the end point, and the generation method of the end state is presented. Extensive comparison experiments and ablation studies prove that our proposed method is superior to other methods with better training efficiency and higher testing accuracy in the robot AOD tasks.

The main contributions of this paper can be summarized as follows:

1. We propose a DQN-based AOD model with a novel training algorithm for service robots.

2. The designed TAM with the generation method of the end state improves the training efficiency and testing accuracy of the DQN-based model for the robot AOD.

3. The comparative experiments and ablation studies based on an AOD dataset show the competitive performance of our method and provide analysis that explains the superiority of the presented training algorithm.

2 Problem formulation

The robot AOD can be regarded as an action decision problem (Han et al., 2019). When detecting an object, the robot should move to an appropriate observation location where it can have a better detection view of the target. This motion process is modeled by $(S, A, f_{\mathrm{r}}, T, \pi)$. The description of each element is provided as follows:

1. $S = \{s_t, s_{t+1}, ..., s_{t+n}\}$ denotes a space of robot states at different time points. The content of each s is the robot observation of its environment.

2. $A = \{a_1, a_2, ..., a_6\}$ is an action space including six motion directions of the robot, which are clockwise rotation (rotate_cw), counterclockwise rotation (rotate_ccw), left, right, forward, and backward.

3. f_{r} expresses a reward function

$$r_t = f_{\mathrm{r}}(s_t, a_t), \tag{1}$$

aiming to give a score of the robot action a_t in the current state s_t.

4. T as a state transition function

$$s_{t+1} = T(s_t, a) \tag{2}$$

brings the robot from the current state s_t into the next state s_{t+1} after the robot executes an action a.

5. π, the action selection strategy, is the map from the state space S to the action space A,

$$a_t = \pi(s_t), \tag{3}$$

which guides the robot to execute an action a_t according to the current state s_t.

As the formulation is established, the workflow of the AOD is demonstrated in Fig. 1. In the initial state s_t, a robot starts to detect the target object appearing in the current observation o_t. In the next step, $t+1$, the robot will choose an action a_t based on the action selection strategy π to get into the next state s_{t+1} with a new observation o_{t+1}. Meanwhile, the robot will obtain a reward r_t from the environmental feedback. The reward function (1) is used to guide the robot to choose the correct action in each state; its principle is that a good action receives a positive reward while a bad action receives a negative reward. Combining functions (2) and (3), this state transition process is expressed by

$$s_{t+1} = T(s_t, a) = T(s_t, \pi(s_t)), \tag{4}$$

where s_{t+1} is closely related to the last state s_t and the last action a_t.

An appropriate action can lead the robot to obtain a better observation view than that of the last state. For example, in Fig. 1, when the bounding box of the target object lies on the right side of o_t in s_t, the rotate_cw action will be executed to make the robot have a better view where the target object is in the center of o_{t+1}. If the robot acts incorrectly in s_t, such as rotate_ccw, it will lose the target object in the observation of the next state. Thus, the action selection strategy is crucial. The problem solved in this study is to generate an optimal action selection strategy π^* based on DQN (Mnih et al., 2015) for the robot to have a better detection view. This issue can be optimized by the action-value function:

$$Q^{\pi}(s, a) = \mathbb{E}[R_t | s = s_t, a = a_t, \pi], \tag{5}$$

which represents the sum of all the robot's rewards from start to end in an AOD task by performing an action a_t based on strategy π in each state s_t. The purpose is to find an optimal action-value function

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a), \tag{6}$$

for solving the robot AOD problem. In this work, the action-value function is represented approximately by a designed deep neural network:

$$Q(s, a|\theta) \longrightarrow Q^*(s, a), \tag{7}$$

where θ denotes the parameter of the deep neural network.

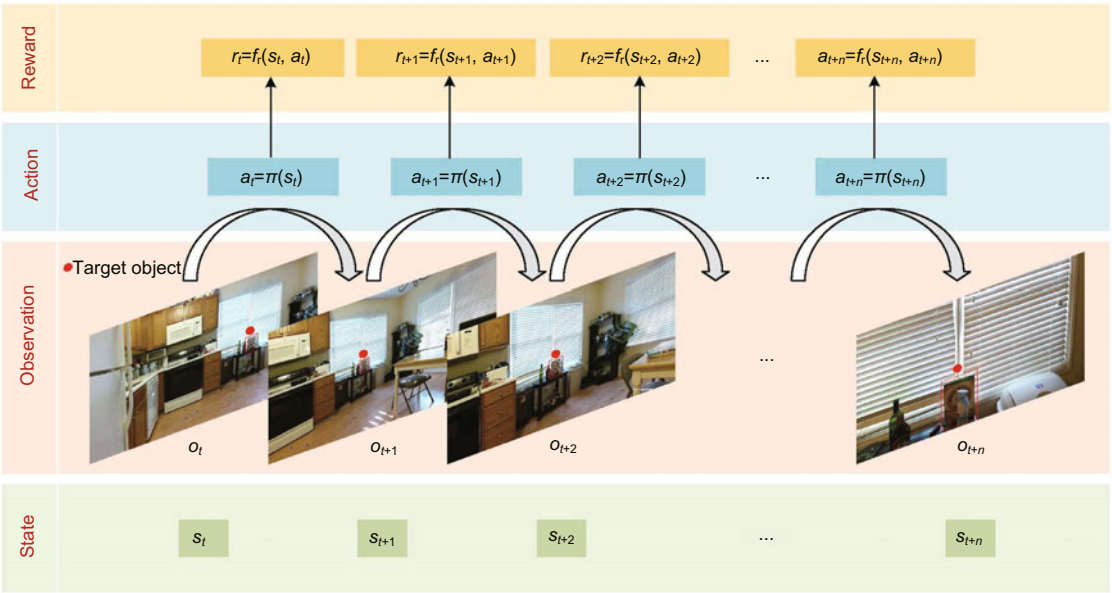

Fig. 1 Workflow of active object detection (AOD) expressed in four parts: state, observation, action, and reward (Red point and bounding box represent the target object in the observation part. References to color refer to the online version of this figure)

3 Proposed method

3.1 State space and model design

The DQN state space represents the environmental information perceived by the robot and the changes caused by the robot's movement. The state space is the basis for the robot to make decisions and evaluate its long-term benefits. The quality of the state space design directly determines whether the DQN algorithm can converge or not.

In the AOD task, the state space is mostly represented by robot's visual information. As shown in Fig. 2, the state space contains two parts: the captured scene image and the bounding box. The scene image captured by an RGB camera displays the current environmental state information, which can reflect the space location of the robot in the current state. Meanwhile, the bounding box provides the location of the target object. The values of the bounding box are normalized to prevent the gradient from vanishing. The environmental information combined with the bounding box of the target object is beneficial in leading the robot to approach the target object. Because the emphasis on study of this thesis is the action selection strategy for robot AOD rather than the object detection model, the bounding box in the state is assumed to be available using an object detection model, such as Faster R-CNN (Ren et al., 2017).

The model of the proposed DQN for the robot AOD is shown in Fig. 2, including the state information as the input, the feature extraction module with two extractors (CNN feature extractor and location feature extractor), and the three-layer MLP.

The model is designed as a double-input network because the state contains two types of state information. To extract the features of the RGB image, the CNN feature extractor is designed based on the residual blocks. As shown in Fig. 3, the CNN feature extractor contains a convolution and pooling layer, 16 residual blocks, a pooling layer, and a flatten layer. In each residual block, there are batch normalization (BN) layers, activation layers using rectified linear unit (ReLU), and two kinds of convolution layers (1×1 conv and 3×3 conv). The functions of the BN layers are to speed up the network training and avoid gradient explosion and gradient disappearance. The 1×1 conv adds non-linear activation to the last layer. The 3×3 conv increases the layer number. In the final layer of the CNN feature extractor, the flatten layer is used to process the features to form a one-dimensional (1D) feature vector. For the location feature extractor, the coordinate values of the bounding box are normalized to generate a 1D feature vector, which is combined with the CNN feature to create a fusion feature using the concat operation.

After the feature extraction mentioned above, the MLP is used to fit the Q-value functions of different actions. Considering the complexity and training difficulty of the network, the MLP in the DQN includes only three layers: the input layer (IL), hidden layer (HL), and output layer (OL). The input

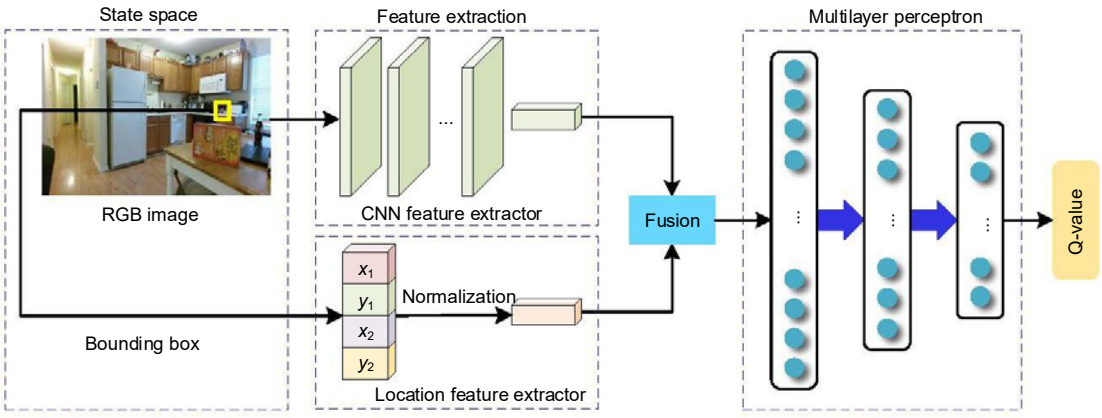

Fig. 2 Model of the proposed deep Q-learning network (DQN) for the robot active object detection (AOD), including the state information as the input, the feature extraction module with two extractors (convolutional neural network (CNN) feature extractor and location feature extractor), and the three-layer multilayer perceptron (MLP)

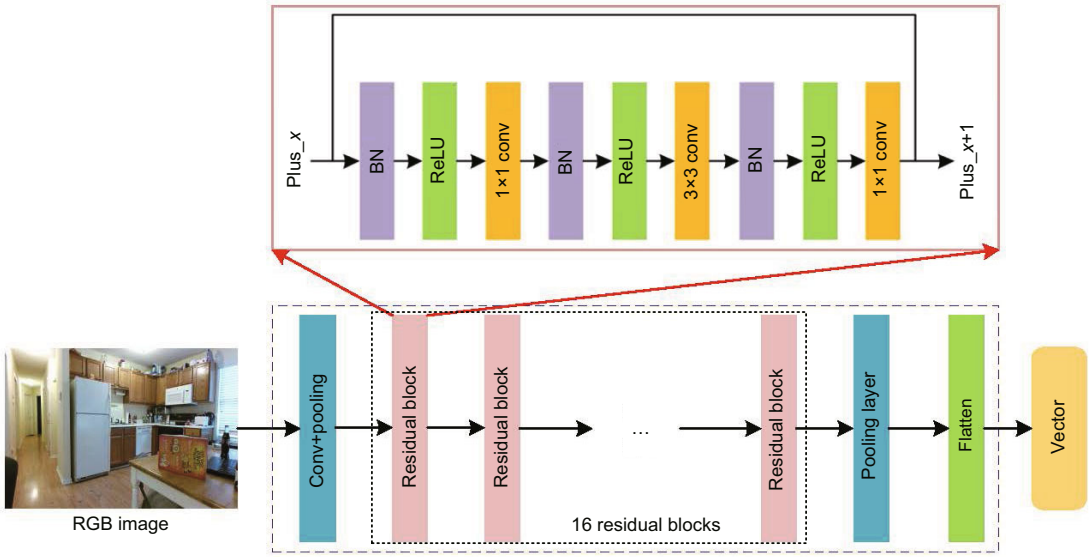

Fig. 3 Network structure of the designed convolutional neural network (CNN) feature extractor, including a convolution and pooling layer, 16 residual blocks, a pooling layer, and a flatten layer (The output of the CNN feature extractor is a one-dimensional feature vector)

of the IL is the fusion feature and the HL contains 64 neurons. There are six neurons in the OL, because there are six categories of action in the robot AOD. The activation function used in the MLP is ReLU. The function of the DQN model is designed as a non-linear approximation of the optimal action-value $Q^*(s,a)$ in function (7). During the training, all parameters θ's in the DQN are optimized.

3.2 Training algorithm based on memory

To improve the training performance of the DQN model for robot AOD, a novel TAM is proposed in this subsection. In the raw DQN training algorithm, the robot takes a random action with a certain probability, or an action generated by the trained model in each state. The action taken may lead to a failed exploration, which receives a negative reward. For the same or similar states, the robot may take the same incorrect actions repeatedly, which leads to a lot of training data with negative rewards. However, too much training data with negative rewards has an adverse effect on the training efficiency and testing accuracy of the DQN model. Consequently, an exploration method with memory is added to TAM to avoid the repetitive error actions in the same state, which reduces the amount of training data with negative rewards. The TAM is summarized in Algorithm 1. When taking an action, the flag continue_search is set to avoid repeatedly making wrong actions (lines 16–18 and lines 36–38 in Algorithm 1).

The reward function used in the TAM is based on the work in Liu et al. (2022a):

$$f_{\mathrm{r}}(s_t,a_t)=\begin{cases}1, & \text{if } a_t \text{ is } a_{\max},\\ -1, & \text{otherwise},\end{cases} \tag{8}$$

where $a_{\max}$ is the subscript corresponding to the maximum value of $S_{a_i}^{s_t}$ $(i=1,2,...,6)$ in

$$L=\{S_{a_1}^{s_t},S_{a_2}^{s_t},...,S_{a_6}^{s_t}\}. \tag{9}$$

Each score $S_{a_i}^{s_t}$ of the adjacent state is computed by

$$S_{a_i}^{s_t}=\frac{S_{\mathrm{b}}^{s_{t+1}}-S_{\mathrm{b}}^{s_t}}{S_{\mathrm{b}}^{s_t}}+\frac{D^{s_t}-D^{s_{t+1}}}{D^{s_t}}, \tag{10}$$

where S_{b} is the area of the bounding box and D is the center distance between the bounding box and the observation image.

Each episode must end when the robot has arrived near the target object location and obtained a good observation point. Therefore, judging whether the robot has arrived at the appropriate place is a key problem during training, which is the same as generating the end state s_{e} (line 6 in Algorithm 1) of the episode. Therefore, the generation method of the end state (GMES) is proposed based on the position and acreage of the target object bounding

box, which is summarized in Algorithm 2. In the final state, the robot can achieve a good view of the target object. In some cases, the two actions in the adjacent states are inverse (e.g., a_t = forward and a_{t+1} = backward), which results in the reciprocating motion of the robot. To overcome this issue, the last action a_l is recorded to avoid choosing the inverse action in the current state and then select the action that has the second value in the action-score list as the best action (lines 7–10 in Algorithm 2). All the generated end states will be checked and saved for the TAM. An example of the end state generation is demonstrated in Fig. 4.

Algorithm 1 TAM for robot AOD

Require: initialized training data $\mathcal{D}$
Ensure: well-trained DQN model
1: Define a storage $\mathbb{S}$ and initialize the evaluation DQN $Q_\mathrm{e}(s,a|\theta_\mathrm{e})$ and the target DQN $Q_\mathrm{t}(s,a|\theta_\mathrm{t})$ with random parameters θ_e and θ_t
2: $N_\mathrm{s}=0$ /* N_s is the number of steps */
3: **for** episode=1:M **do**
4: $N_\mathrm{s}=N_\mathrm{s}+1$
5: Initialize a scene image I_i with a target object o_id as the start state s_t from $\mathcal{D}$
6: Obtain s_e /* s_e is the end state of this episode */
7: done=False
8: continue_search = False
9: action_list = $\{a_1,a_2,a_3,a_4,a_5,a_6\}$
10: **while** not done **do**
11: **if** $N_\mathrm{s}>200$ **then**
12: Select a random action a_t with probability ϵ; otherwise, $a_t=\max Q_\mathrm{e}(s_t,a|\theta_\mathrm{e})$
13: **else**
14: Select a random action a_t
15: **end if**
16: **if** continue_search and action_list are not null **then**
17: $a_t=a_\mathrm{s}$ /* sample an action from action_list */
18: **end if**
19: $s_{t+1}=T(s_t,a_t)$
20: **if** s_{t+1} is null **then**
21: continue_search = True
22: $s_{t+1}=s_t,\ r_t=-1$
23: **else**
24: **if** $s_{t+1}=s_\mathrm{e}$ **then**
25: done = True, $r_t=1$
26: **else**
27: continue_search = True, $r_t=f_\mathrm{r}(s_t,a_t)$
28: **end if**
29: **end if**
30: Store (s_t,a_t,r_t,s_{t+1}) in $\mathbb{S}$
31: $s_t=s_{t+1}$
32: **if** $N_\mathrm{s}>2000$ **then**
33: Sample batch size B_s from $\mathbb{S}$ randomly
34: Use B_s to train $Q_\mathrm{e}(s,a|\theta_\mathrm{e})$ and update θ_e by the RMSProp optimizer
35: **end if**
36: **if** continue_search **then**
37: Remove a_t in action_list
38: **end if**
39: **if** $N_\mathrm{s}\%2000==0$ **then**
40: $\theta_\mathrm{t}=\theta_\mathrm{e}$
41: **end if**
42: **end while**
43: **end for**
44: Return $Q_\mathrm{t}(s,a|\theta_\mathrm{t})$

4 Experiments

4.1 Dataset and setup

An active vision dataset benchmark (AVDB) (Ammirato et al., 2017, 2018) has been proposed to develop and compare robot AOD methods in a real-world environment, and here it is used to verify the performance of the proposed method. The AVDB included 14 family environments and one office environment to simulate a robot moving to find target objects in an indoor environment. Most homes were scanned once and some of them (e.g., Home_001 and

Algorithm 2 Generation method of the end state (GMES)

Require: initialized state s_1 and the target object o
Ensure: end state $s^o_{s_1}$
1: done=True
2: Current state $s_t=s_1$
3: Last action $a_\mathrm{l}=$ None
4: Last state $s_\mathrm{l}=$ None
5: **while** done **do**
6: Obtain the best action a_max in s_t by Eq. (9)
7: **if** a_max is the inverse action of a_l **then**
8: a_max is the action that owns the second largest value in Eq. (9)
9: **end if**
10: $a_\mathrm{l}=a_\mathrm{max}$
11: $s_\mathrm{l}=s_t$
12: Next state $s_{t+1}=T(s_t,a_\mathrm{max})$
13: **if** a_max == backward **then**
14: done = False
15: **end if**
16: **end while**
17: $s^o_{s_1}=s_\mathrm{l}$
18: Return $s^o_{s_1}$

Fig. 4 An example of the end state generation based on the GMES, where the bounding box represents the target object

Home_003) were scanned twice. Some images of various homes in the AVDB are shown in Fig. 5. The common manipulable object instances in a household environment appearing in the BigBIRD dataset (Singh et al., 2014) were placed in every home for object detection. There were 33 instance classes in the AVDB, and some examples are shown in Fig. 6.

As shown in Table 1, the data collected from 10 homes were used to train models and the trained models were tested in two homes. Two twice-scanned scenes (e.g., Home_001 and Home_005) appeared in both training and test data. In the second scan scenes (e.g., Home_001_2 and Home_005_2), some instances were moved around, some were removed completely, and some new instances were added.

The success rate (SR) and average step (AS, representing the average number of steps) were used

Fig. 5 Some scene images in the active vision dataset benchmark (AVDB)

Fig. 6 Some examples of the object instances in the active vision dataset benchmark (AVDB)

Table 1 Experimental environment setup

Training environment	Testing environment
Home_001_1, Home_002_1, Home_003_1, Home_004_1, Home_004_2, Home_005_1, Home_006_1, Home_008_1, Home_014_1, Home_014_2	Home_001_2, Home_005_2

to compare the performances of different AOD methods. SR is defined by

$$\mathrm{SR} = \frac{N_\mathrm{s}}{N}, \tag{11}$$

where N_s expresses the number of successful tasks and N is the number of all tasks. AS is computed by

$$\mathrm{AS} = \frac{\sum_{t=1}^{N_\mathrm{s}} L_t}{N_\mathrm{s}}, \tag{12}$$

where L_t is the step number of the success task t. The training parameters are given in Table 2.

Table 2 Training parameters

Parameter	Value
Image size	224×224
Mini batch size	64
Learning rate	0.001
Reward decay	0.9
Memory size	2000
Number of target iterations replaced	2000
Optimizer	RMSProp

4.2 Results and analysis

The proposed AOD method was compared with the following DQN-based models (DQN, DDQN, D3QN, and DQN_dueling).

1. DQN (Mnih et al., 2015): This is a deep Q-learning model that has the same architecture as in Fig. 2. The difference between the DQN and the proposed method is that the DQN is trained by raw training arithmetic.

2. DDQN: The double Q-learning training strategy (van Hasselt et al., 2016) is used to train the DQN.

3. D3QN: DDQN has a dueling architecture.

4. DQN_dueling: A dueling architecture is added in the DQN based on the idea of Han et al. (2019).

DQN, DDQN, D3QN, DQN_dueling, and our method were trained with the same parameters in Table 2 to ensure the fairness. After 4650 training steps, SR and AS test results of the above methods are provided in Table 3. Moreover, the well-trained performance of our method is compared with those of some state-of-the-art AOD methods in Table 4.

The following conclusions can be drawn from the results of the comparison experiments:

Table 3 Results of different DQN-based methods

Method	SR			AS		
	Home_001_2	Home_005_2	Average	Home_001_2	Home_005_2	Average
DQN	0.1984	0.7229	0.4607	26.1421	21.6625	23.9023
DDQN	0.3669	0.8705	0.6187	28.5379	18.2388	23.3884
D3QN	0.1294	0.8223	0.4759	28.1935	19.7875	23.9905
DQN_dueling	0.2559	0.9494	0.6027	23.0592	18.0036	20.5314
Our method	0.6020	0.9849	0.7935	24.1997	17.8532	21.0265

Table 4 Performance comparison with some state-of-the-art AOD methods

Method	SR
Ammirato et al. (2017)'s	0.51
Schmid et al. (2019)'s	0.61
Mousavian et al. (2019)'s	0.65
Han et al. (2019)'s	0.62
Xu et al. (2021)'s	0.77
Ours	0.84

1. In DQN-based methods, our method has the highest SR (0.7935) and the second-lowest AS (21.0265).

2. Because the environment Home_001_2 is more complicated than Home_005_2, SR in Home_001_2 are lower than that in Home_005_2. However, SR of our method is larger than 0.6 in Home_001_2, which is much higher than those of the other methods.

3. Compared with the DQN, the average AS of our method is 2.8758 smaller and the average SR of our method is 0.3328 higher, especially about 0.4 in Home_001_2. This proves that the model trained by the proposed TAM can improve the performance of the robot AOD with a higher SR and a lower AS.

4. Table 4 shows that our method is more accurate than state-of-the-art methods in the AOD tasks. Compared with the AOD model in Xu et al. (2021), the SR of our method is 0.07 higher.

In summary, our method outperforms the other methods on AOD tasks.

To analyze the performance difference between the proposed TAM and the raw training algorithm (RTA), we completed an experiment concerning data generation in the training process. When the DQN model is trained based on TAM and RTA, every generated transition T, $\{s_t, a_t, r_t, s_{t+1}\}$, is recorded. Different transitions are classified into two categories: positive transition (PT) and negative transition (NT). PT receives positive rewards while NT receives negative rewards. During training, the DQN model learns from various transitions and the number of transitions in one learning step is the mini batch size n_{bs}. To show the number of PTs in the current learning step t, the PT proportion in the current mini batch is calculated by

$$\mathrm{PTPM} = \frac{\mathrm{Numb}(\mathrm{PT}_t)}{n_{\mathrm{bs}}}. \tag{13}$$

The PT proportion from the beginning ($t = 1$) to the current learning step ($t = n$) is computed by

$$\mathrm{PTPB} = \frac{\sum_{t=1}^{n} \mathrm{Numb}(\mathrm{PT}_t)}{n \times n_{\mathrm{bs}}}. \tag{14}$$

The curves of T, PT, PTPB, and PTPM based on RTA and TAM are demonstrated in Fig. 7.

As shown in Fig. 7, the rate range of PTPM based on TAM is 0.10–0.33, much higher than that based on RTA (the rate range of PTPM based on RTA is from 0 to 0.15). The rate of PTPB based on TAM is >0.2 in every learning step, which is larger than that of PTPB based on RTA (about 0.05). The rates of PT based on RTA and TAM are almost straight lines. With increased learning steps, the number of PTs grows, but the growth rate of the PT based on TAM is faster. Therefore, the DQN model can learn more PTs based on the proposed TAM and avoids learning NTs repeatedly using RTA, reflecting that our method outperforms the other models with a higher SR and a lower AS. Meanwhile, more PTs can effectively improve the performance of the DQN model.

To study the relationship between the object detection accuracy (ODA) and the AOD performance, our model was tested with different ODAs. From the results in Table 5, it can be seen that ODA can affect the performance of the AOD model. A low ODA leads to a small SR; for example, the average SR is only 0.58 with ODA being 0.7. With an increase in ODA, the SR is improved. When ODA is 1.0, our model obtains the largest average SR of 0.84.

Therefore, it is important to ensure that the object detection model has high ODA when applying the AOD model to an actual scene.

4.3 Ablation study

In this subsection we discuss the ablation studies concerning the state space design and the training algorithm. In the proposed DQN model, the state space includes an RGB image observed by the robot (rgb) and the bounding box of the target object (box). Considering that the robot usually has an RGBD camera, the depth image (depth) is added to the state space to train and test the proposed DQN model to explain the rationality of our state space design. The DQN model testing results with various state spaces are given in Table 6. It can be seen from Table 6 that rgb+box has the best SR and AS. The RGB image can represent the environment better than the depth image, so rgb+box is more accurate and efficient than depth+box. When combining the RGB image and the depth image as the state information, the improved performance of the DQN model based on rgb+depth+box is not obvious because the feature extraction of the depth image can increase the complexity of the DQN model. Therefore, the designed state space rgb+box is reasonable for the proposed DQN model.

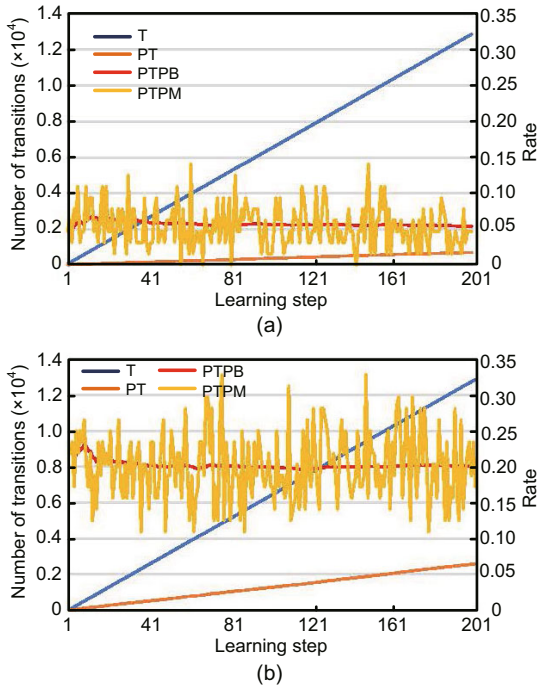

Fig. 7 Curves of T, PT, PTPB, and PTPM based on RTA (a) and TAM (b) during 201 learning steps

To assess the effectiveness of the TAM, training efficiency experiments were carried out. The proposed DQN model has been trained based on different training algorithms, RTA and TAM. During training, the model was saved every 465 learning steps and was tested in Home_001_2 and Home_005_2. The testing curves of RTA and TAM are shown in Fig. 8 and the best testing results of these two training algorithms are provided in Table 7. From the curves in Fig. 8, the RTA training speed is lower than that of TAM, which needs 27×465 steps to obtain the best SR of 0.79. However, the DQN model trained by TAM can achieve a 0.8 SR when the number of steps is only 5×465. From the results in Table 7, the average SR of the TAM is 0.84, much higher than that of the RTA (0.79). Meanwhile, the AS of the TAM is smaller than that of the RTA. In summary, the proposed

Table 5 Results of the DQN model based on different object detection accuracy (ODA) values

ODA	SR			AS		
	Home_001_2	Home_005_2	Average	Home_001_2	Home_005_2	Average
0.7	0.47	0.69	0.58	22.94	13.88	18.41
0.8	0.54	0.80	0.67	22.80	13.86	18.33
0.9	0.61	0.88	0.75	22.81	13.80	18.31
1.0	0.67	1.00	0.84	22.79	13.88	18.34

Table 6 Results of the DQN model with different state spaces

State space	SR			AS		
	Home_001_2	Home_005_2	Average	Home_001_2	Home_005_2	Average
rgb+box	0.6747	0.9970	0.8359	22.7858	13.8761	18.3310
depth+box	0.4730	0.9398	0.7064	26.4076	19.5962	23.0019
rgb+depth+box	0.5815	0.8102	0.6959	26.5360	19.4572	22.9966

Table 7 Results of the DQN model based on different training algorithms (TAs)

TA	SR			AS		
	Home_001_2	Home_005_2	Average	Home_001_2	Home_005_2	Average
RTA	0.58	0.99	0.79	22.77	13.98	18.38
TAM	0.67	1.00	0.84	22.79	13.88	18.34

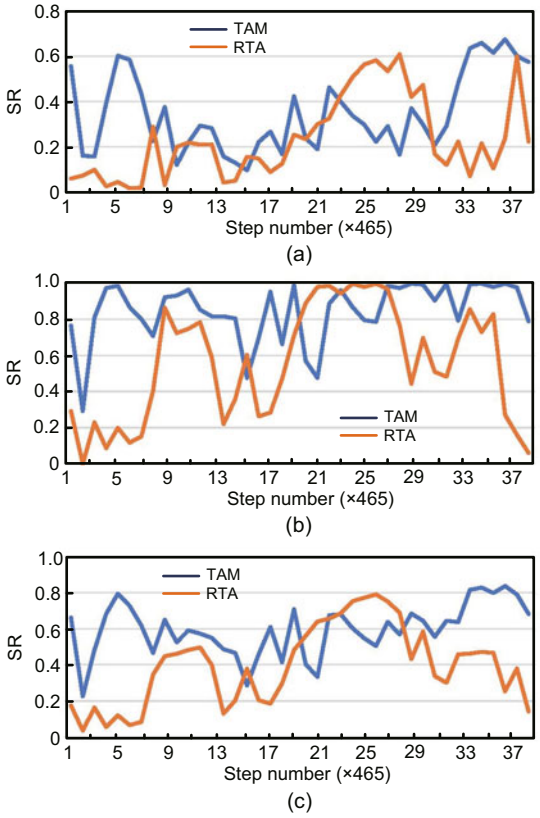

Fig. 8 Test curves of SR based on RTA and TAM in Home_001_2 (a), Home_005_2 (b), and the average SR (c)

TAM can improve the training performance and test accuracy of the DQN model, and guarantee high detection efficiency.

5 Conclusions

In this paper, a DQN-based AOD model with a novel training algorithm is proposed to increase the training efficiency and improve the performance of the DQN model. In particular, a training algorithm based on memory is designed to avoid repetitive learning of the data with negative rewards and increase the amount of data with positive rewards in the training process. Our method is evaluated on an AOD dataset, and the experimental results demonstrate that our method can improve the performance of the robot AOD and speed up the DQN training process. In addition, the ablation studies explain the rationality of the proposed method. In the future, we will study how a robot makes an appropriate action prediction if the target object is not detected at the beginning of the AOD task.

Contributors

Shaopeng LIU and Guohui TIAN designed the research. Shaopeng LIU addressed the problems, processed the data, and drafted the paper. Guohui TIAN, Yongcheng CUI, and Xuyang SHAO helped organize the paper. Shaopeng LIU and Guohui TIAN revised and finalized the paper.

Compliance with ethics guidelines

Shaopeng LIU, Guohui TIAN, Yongcheng CUI, and Xuyang SHAO declare that they have no conflict of interest.

References

Ammirato P, Poirson P, Park E, et al., 2017. A dataset for developing and benchmarking active vision. Proc IEEE Int Conf on Robotics and Automation, p.1378-1385. https://doi.org/10.1109/ICRA.2017.7989164

Ammirato P, Berg AC, Košecká J, 2018. Active vision dataset benchmark. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition Workshops, p.2046-2049. https://doi.org/10.1109/CVPRW.2018.00277

Dos Reis DH, Welfer D, De Souza Leite Cuadros MA, et al., 2019. Mobile robot navigation using an object recognition software with RGBD images and the YOLO algorithm. *Appl Artif Intell*, 33(14):1290-1305. https://doi.org/10.1080/08839514.2019.1684778

Duan KW, Bai S, Xie LX, et al., 2019. CenterNet: keypoint triplets for object detection. Proc IEEE/CVF Int Conf on Computer Vision, p.6568-6577. https://doi.org/10.1109/ICCV.2019.00667

Han XN, Liu HP, Sun FC, et al., 2019. Active object detection with multistep action prediction using deep Q-network. *IEEE Trans Ind Inform*, 15(6):3723-3731. https://doi.org/10.1109/TII.2019.2890849

He KM, Zhang XY, Ren SQ, et al., 2016. Deep residual learning for image recognition. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.770-778. https://doi.org/10.1109/CVPR.2016.90

Liu SP, Tian GH, Zhang Y, et al., 2022a. Active object detection based on a novel deep Q-learning network and long-term learning strategy for the service robot.

IEEE Trans Ind Electron, 69(6):5984-5993. https://doi.org/10.1109/TIE.2021.3090707

Liu SP, Tian GH, Zhang Y, et al., 2022b. Service planning oriented efficient object search: a knowledge-based framework for home service robot. *Exp Syst Appl*, 187:115853. https://doi.org/10.1016/j.eswa.2021.115853

Mnih V, Kavukcuoglu K, Silver D, et al., 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529-533. https://doi.org/10.1038/nature14236

Mousavian A, Toshev A, Fišer M, et al., 2019. Visual representations for semantic target driven navigation. Proc IEEE Int Conf on Robotics and Automation, p.8846-8852. https://doi.org/10.1109/ICRA.2019.8793493

Paletta L, Pinz A, 2000. Active object recognition by view integration and reinforcement learning. *Robot Autom Syst*, 31(1-2):71-86. https://doi.org/10.1016/S0921-8890(99)00079-2

Pu SL, Zhao W, Chen WJ, et al., 2021. Unsupervised object detection with scene-adaptive concept learning. *Front Inform Technol Electron Eng*, 22(5):638-651. https://doi.org/10.1631/FITEE.2000567

Ren SQ, He KM, Girshick R, et al., 2017. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans Patt Anal Mach Intell*, 39(6):1137-1149. https://doi.org/10.1109/TPAMI.2016.2577031

Schmid JF, Lauri M, Frintrop S, 2019. Explore, approach, and terminate: evaluating subtasks in active visual object search based on deep reinforcement learning. Proc IEEE/RSJ Int Conf on Intelligent Robots and Systems, p.5008-5013. https://doi.org/10.1109/IROS40897.2019.8967805

Shuai W, Chen XP, 2019. KeJia: towards an autonomous service robot with tolerance of unexpected environmental changes. *Front Inform Technol Electron Eng*, 20(3):307-317. https://doi.org/10.1631/FITEE.1900096

Singh A, Sha J, Narayan KS, et al., 2014. BigBIRD: a large-scale 3D database of object instances. Proc IEEE Int Conf on Robotics and Automation, p.509-516. https://doi.org/10.1109/ICRA.2014.6906903

van Hasselt H, Guez A, Silver D, 2016. Deep reinforcement learning with double Q-learning. Proc AAAI Conf on Artificial Intelligence, p.2094-2100. https://doi.org/10.1609/aaai.v30i1.10295

Wan SH, Goudos S, 2020. Faster R-CNN for multi-class fruit detection using a robotic vision system. *Comput Netw*, 168:107036. https://doi.org/10.1016/j.comnet.2019.107036

Wang Q, Fan Z, Sheng WH, et al., 2019. Finding misplaced items using a mobile robot in a smart home environment. *Front Inform Technol Electron Eng*, 20(8):1036-1048. https://doi.org/10.1631/FITEE.1800275

Xu QL, Fang F, Gauthier N, et al., 2021. Towards efficient multiview object detection with adaptive action prediction. Proc IEEE Int Conf on Robotics and Automation, p.13423-13429. https://doi.org/10.1109/ICRA48506.2021.9561388

Zhang H, Liu HP, Guo D, et al., 2017. From foot to head: active face finding using deep Q-learning. Proc IEEE Int Conf on Image Processing, p.1862-1866. https://doi.org/10.1109/ICIP.2017.8296604

Zhou XY, Zhuo JC, Krähenbühl P, 2019. Bottom-up object detection by grouping extreme and center points. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.850-859. https://doi.org/10.1109/CVPR.2019.00094