Received: 23 September 2020 | Revised: 18 January 2021 | Accepted: 25 February 2021 | CAAI Transactions on Intelligence Technology

DOI: 10.1049/cit2.12043

ORIGINAL RESEARCH PAPER

Target-driven visual navigation in indoor scenes using reinforcement learning and imitation learning

Qiang Fang | Xin Xu | Xitong Wang | Yujun Zeng

Department of Intelligence Science and Technology, College of Intelligence Science and Technology, National University of Defense Technology, Changsha, China

Correspondence

Qiang Fang, Department of Intelligence Science and Technology, College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China.
Email: qiangfang@nudt.edu.cn

Funding information

National Natural Science Foundation of China, Grant/Award Numbers: 61703418, 61825305

Abstract

Here, the challenges of sample efficiency and navigation performance in deep reinforcement learning for visual navigation are focused and a deep imitation reinforcement learning approach is proposed. Our contributions are mainly three folds: first, a framework combining imitation learning with deep reinforcement learning is presented, which enables a robot to learn a stable navigation policy faster in the target-driven navigation task. Second, the surrounding images is taken as the observation instead of sequential images, which can improve the navigation performance for more information. Moreover, a simple yet efficient template matching method is adopted to determine the stop action, making the system more practical. Simulation experiments in the AI-THOR environment show that the proposed approach outperforms previous end-to-end deep reinforcement learning approaches, which demonstrate the effectiveness and efficiency of our approach.

1 | INTRODUCTION

Autonomous navigation plays an important role in the field of robotics. In recent decades, due to the benefits of low cost, lightweight and rich information, visual navigation has attracted considerable attention, and many studies have been presented. In general, most of the traditional methods for visual navigation are rule-based methods [1–5] that locate the robot's position accurately for the planning and control modules. However, these rule-based methods are relatively complex, involving considerable manual design work and computational resources.

Considering human navigation, there is no need to calculate the exact position; we only need to determine where to go according to the scenes we have seen. Inspired by such behaviour, there has been an increasing interest in end-to-end visual navigation approaches that directly navigate from images to actions. Deep learning (DL) can extract useful features directly from pixels, which has made great progress in many fields, such as image classification [6], object detection [7] and image segmentation [8]. Reinforcement learning (RL) is similar to the learning processes of humans, where policies are optimized through continuous trial-and-error and feedback with the environment to achieve a higher level of intelligence [9, 10]. Moreover, the deep reinforcement learning (DRL) method combining DL with RL has been proposed and achieved exciting performance in dealing with input problems for continuous high-dimensional states [11], so using the DRL method to solve end-to-end visual navigation is a good choice. Zhu [12] proposed a target-driven DRL (TD-DRL) architecture to solve end-to-end visual navigation problems and achieved good navigation performance; however, there are some limitations. Firstly, four sequential images are considered as the observation; in fact, the robot has access to the surrounding observations, which might provide more information. Secondly, the stop action is not considered, although it can be automatically determined in the simulation environment by judging whether the ID number of the current state is that of the target state, but such an approach is not suitable in reality because there is no ID number but only images. Last but most importantly, DRL requires a large number of samples to learn a good navigation policy during training, which leads to a long training time and low efficiency for training processes. To improve sample efficiency during training, the imitation learning (IL) method is an alternative that can imitate expert experience accurately, and it requires less training time to achieve a navigation model. However, using IL alone might result in the problem of overfitting compared to DRL.

Inspired by the limitations of TD-DRL and the advantages of IL, here, we focus on the challenges of sample efficiency and navigation performance in RL for the visual navigation task and propose a deep imitation reinforcement approach (DIRL), which combines IL with DRL. The main contributions are summarized as follows:

- A framework combining IL with RL is presented, which enables a robot to learn a stable navigation policy faster in the target-driven navigation task.
- For the observation, the surrounding images are considered as the observation instead of sequential images, which can improve navigation performance and accelerate the learning process for more information. Additionally, a simple template matching method is adopted to determine the action, which makes the system more practical.
- Simulation experiments are evaluated, which demonstrates the effectiveness of our approach compared with the state-of-the-art approaches.

The remainder of the paper is organized as follows: we introduces the related work in Section 2 and describes our proposed approach in Section 3. Experiments in AI2-THOR environment are evaluated and results are discussed in Section 4. Finally, a conclusion is drawn in Section 5.

2 | RELATED WORK

Robot visual navigation can be categorized into two methods: rule-based and learning-based. Here, we focus on the latter, so we will introduce the related work of learning-based navigation, especially the IL and DRL used in visual navigation.

IL is the process of learning a behaviour policy from a set of demonstrations, and a survey about IL can be found in [13]. Some IL methods treat the task as special supervised learning, that is, behaviour cloning (BC). NVIDIA [14] proposed a typical BC-based end-to-end visual navigation method used in unmanned driving areas, which collects a large number of expert samples from three cameras. Based on IL, Yang [15] proposed a multi-task architecture that can simultaneously predict the vehicle's speed and steering angles using image sequences. To drive a vehicle, Wang [16] proposed a novel angle-branch network, where the inputs include sequential images, the vehicle speed and the desired angle of the subgoal. Codevilla [17] proposed a condition-imitation-learning method based on high-level input, which might partly address the generalization ability of the models trained using IL. To address the limitation that most IL methods need a specifically calibrated actuation setup for the training dataset to be performed, Xu [18] proposed a novel framework (FCN-LSTM) to learn the policy of driving from uncalibrated large-scale videos. Although those imitation-learning-based methods are useful for visual navigation, the systems are easy to overfit using IL alone.

RL has recently been applied in robot navigation tasks due to its ability to interact with the environment, especially DRL. In general, one typical DRL method is the deep Q-learning algorithm (DQN) proposed by DeepMind, which successfully enabled robots to learn control policies at the human level in Atari games [19]. Subsequently, many algorithms for improving the DQN network model were successively presented and achieved impressive results in different fields [20–23]. Other methods based on the policy gradient include the deep deterministic policy gradient (DDPG) [24] and the A3C [25]. Based on the DRL methods, DeepMind proposed a dual-agent architecture [26] that trains different long short-term memories (LSTMs) through RL to achieve outdoor visual navigation and improves the mobility of the model. In [27], the authors aimed to learn a deep successor representation and proposed a feedforward architecture, which is flexible to the changing of rewards and is tested in the MazeBase gridworld such as 3D VizDoom. The AutoRL method [28] combining DRL with gradient-free hyperparametric optimization was also proposed to solve navigation tasks. However, in many cases, the reward function required by RL is difficult to design and cannot guarantee its optimality. Zhu [12] proposed a target-driven DRL (TD-DRL) architecture to solve end-to-end visual navigation problems and achieved good navigation performance. Although DRL can be used in visual navigation, the authors in [29] summarized nine challenges in RL; one of them is the low-sample efficiency, meaning it needs large samples to train, which is not suitable in real-world systems and limits its application in many tasks. The approaches combining DRL with IL have been proposed in many other tasks, such as the grasping task [30], the MuJoCo task [31, 32] and the traffic control task [33], however, such methods utilize some sequence data of trajectory for IL, while here, we propose an approach combining DRL with IL to address the sample efficiency for visual navigation tasks.

3 | PROPOSED APPROACH

3.1 | Markov decision process

The process of end-to-end vision navigation can be modelled as a Markov decision process (MDP) [9], where the agent considers the current observed images and the goal as input and obtains a new state and relevant reward from the environment by performing an action. Then, the MDP can be expressed as $< s_k, a_k, s_{k+1}, r_{k+1} >$, where s_k represents the states at step k, a_k is the corresponding action, and r_{k+1} is the reward function. The main goal is to find an optimal policy π^* to obtain the maximum return during the navigation process:

$$\pi^* = arg\max_{\pi} E(R|\pi) \tag{1}$$

The next state is dependent on the current state and on the decision/action by the involved entity/person; in other words, once the system knows its present state, the future is conditionally independent of the past.

3.2 | Actor-critic algorithm

To deal with MDPs with large (continuous) state or action spaces, here, we adopt the actor-critic algorithm [9] to deal with the end-to-end visual navigation task due to the ability to learn a policy and a state-value function simultaneously. The state-value function $V^{\pi}(s)$ is the expected return and can be formulated as:

$$V^{\pi}(s) = E(R \mid s, \pi) \tag{2}$$

Since the optimal state-value $V^{*}(s)$ is related to the optimal policy π^{*}, the function $V^{*}(s)$ can be further defined as:

$$V^{*}(s) = maxV^{\pi}(s) \tag{3}$$

Note that in the actor-critic learning algorithm shown in Figure 1, a critic network is designed to learn the value functions of the MDP, and an actor network is used to learn near-optimal policies using the estimated value functions in the critic to perform policy searching.

The current state value can be obtained by the value function network in experiments, and the values of actions can be used to evaluate the quality of actions as follows:

$$Q(s,a) = r + \gamma V(s') \tag{4}$$

Since the advantage function $A(s,a) = Q(s,a) - V(s)$ can eliminate the effect of the current state value and reflect values of actions better, it is assumed that some states are inherently better. If some states are always better, the value of each action will be higher only by using the current state value, resulting in exaggerated evaluations that are considered good actions. However, we can evaluate the qualities of actions more objectively after the state value is subtracted for the advantage function. The value of action is positive if it is higher than the evaluation value and negative if it is lower than the evaluation value. The n-step update method for the advantage function is defined as:

$$V(s_0) = r_0 + \gamma r_1 + \cdots + \gamma^n V(s_n) \tag{5}$$

The multi-steps update method can make changes in the advantage function spread faster. If the agent earns an occasional reward, the value function will only slowly move backwards by one step with each iteration. However, each iteration propagates back with an n-step iteration, so it is faster. The estimated $V(s)$ of the algorithm should be convergent. Then, errors can be calculated by the following equation:

$$e \leftarrow r_0 + \gamma r_1 + \cdots + \gamma^{n-1} r_{n-1} + \gamma^n V(s_n) - V(s_0) \tag{6}$$

According to the policy gradient method, the policy loss function is:

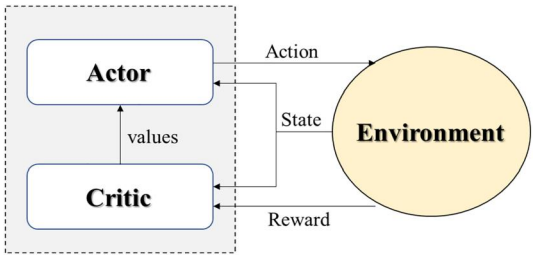

FIGURE 1 Actor-critic algorithm

$$L_{\pi} = \sum_{i=1}^{n} A(s_i, a_i) \log \pi(a_i \mid s_i; \theta') + \beta \sum_{i=1}^{n} H(\pi(s_i; \theta')), \tag{7}$$

where H represents the entropy, and the parameter β adjusts the step length of the entropy regularization term. The value loss function can be calculated as follows:

$$L_V = \frac{1}{2} \sum_{i=1}^{n} e_i^2 \tag{8}$$

The update formula of the policy network is defined as:

$$d\theta_{\pi} \leftarrow d\theta_{\pi} + \nabla_{\theta'} \log \pi(a_i \mid s_i; \theta) e + \beta \nabla_{\theta'} H(\pi(s_i; \theta')) \tag{9}$$

The parameters of the value function network θ_v can be updated by calculating the square of the TD error:

$$d\theta_v \leftarrow d\theta_v + \partial \frac{e^2}{\partial \theta_v'} \tag{10}$$

3.3 | Imitation learning

Here, we first collect expert data (s, a) to achieve experience with navigation [13]. The state s is the input for the expert network, and the action a is the expected output. Note that the output for the expert network is a four-dimensional vector, which is a vector obtained by the softmax regression function. Assuming that the original output is $y_1, y_2, \ldots, y_n$, the softmax regression is as follows:

$$soft\ \max(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_j}}, \tag{11}$$

The cross entropy is used as the loss function for the neural network. The cross entropy can reflect differences in the distribution of two samples. If the difference in the sample distribution is large, the cross entropy will also be large, and vice versa. The cross entropy $H(p, q)$ can be expressed as follows:

$$H(p,q) = -\sum p(x)\log q(x) \tag{12}$$

$q(x)$ are the actual output and $p(x)$ are the desired output, between which the cross entropy can be used to obtain the deviation. After obtaining the loss function, we use the Adam optimizer to optimize and train the network, which is a specialized gradient descent algorithm that uses the computed gradient, its statistics, and its historical values to take small steps in its opposite direction inside the input parameter space as a means of minimizing a function.

3.4 | Template matching

Template matching is a general approach for determining the similarity of two feature vectors (or matrices), especially image features. Since the features of the observation and target images are both extracted from the same networks, here, we use the normalized cross correlation (NNC) [34] method to match the relation of the two inputs to sequentially determine whether the robot has arrived at the goal location. The method is formulated as follows:

$$V_{NNC} = \sum_{i=1}^{N} \frac{(s_i - s_i)(g_i - g_i)}{\left(\sqrt{\sum_{i=1}^{N}(s_i - s_i)^2}\sqrt{\sum_{i=1}^{N}(g_i - g_i)^2}\right.} \tag{13}$$

where N is the dimension of the feature vector, s_i and g_i represent the *ith* feature values of observation and target, respectively, V_{NNC} is the value of NNC, which is limited in [−1,1], and the higher the value is, the better the method, and vice versa.

3.5 | Deep imitation reinforcement learning

Our purpose is to find the best actions to move a robot from its initial location to a goal using only an RGB camera. We focus on learning a good visual navigation policy function π via DRL, and the action a_t at time t can be formulated as [9]:

$$a_t \sim \pi(s_t, g|\theta), \tag{14}$$

where θ is the system parameter, s_t is the current observation, and g is the observation of the target.

Our network, shown in Figure 2, is inspired by the TD-DRL method proposed in [12], which consists of two parts. The first part is the generic Siamese layers, which are the pretrained ResNet-50, and the outputs are embedded into a 512-day space. The second part is the scene-specific layers that capture the common representations of the same scene. The output of the network contains four policy outputs and a single value output. However, there are some improvements that are marked in red. First, the initial parameters of the network are copied from the training results of the IL. Since the robot can capture the surrounding four views in AI-THOR, which is also possible in the real world for the low-speed robot or with a panoramic camera, the second modification adopts the surrounding images to represent the current observation of a robot instead of the sequential four images. Since both use four images, the computational complexities are the same. Moreover, we add a template matching module to determine whether to stop just after the feature extraction. We attempted to add the done action directly into the policy output, but during the training process, we found it is not a good choice. The reason might be that the value of the done action is too sparse, which only occurs at the end of each episode.

The explicit deep imitation reinforcement learning (DIRL) algorithm is summarized in Algorithm 1. The initial weights are from IL. Here, we use the simple BC method, where most of the network is the same as the DRL shown in Figure 2. The difference is that the policy outputs are known and labelled by experts. Moreover, the parameters related to the value output are not trained and frozen during imitation training. Note that the performed action is determined by an additional template matching process because it can be calculated online using Equation (13), so there is no need to train.

Algorithm 1: Deep imitation reinforcement learning (DIRL) algorithm framework

Collect expert data to obtain sample set D_e.
Imitation learning: train the expert network.
Set the weights obtained from imitation learning as the initial weights of the networks.
Set the target T, the matching coefficient r_0, the number of episodes M and the max steps N in each episode.
for *episode* = 1 to M **do**
  Initialize observation S_0.
  for t = 1 to N **do**
    Obtain action a_t according to the DRL algorithm;
    Obtain reward R_{t+1} and state S_{t+1} from the environment;
    Train the model by data $(S_t, a_t, S_{t+1}, R_{t+1})$;
    Update the network parameters based on gradients for DRL.
    Calculate the NNC using Equation (13), if V_{NNC} is larger than a threshold, go to the new episode.
  end for
end for

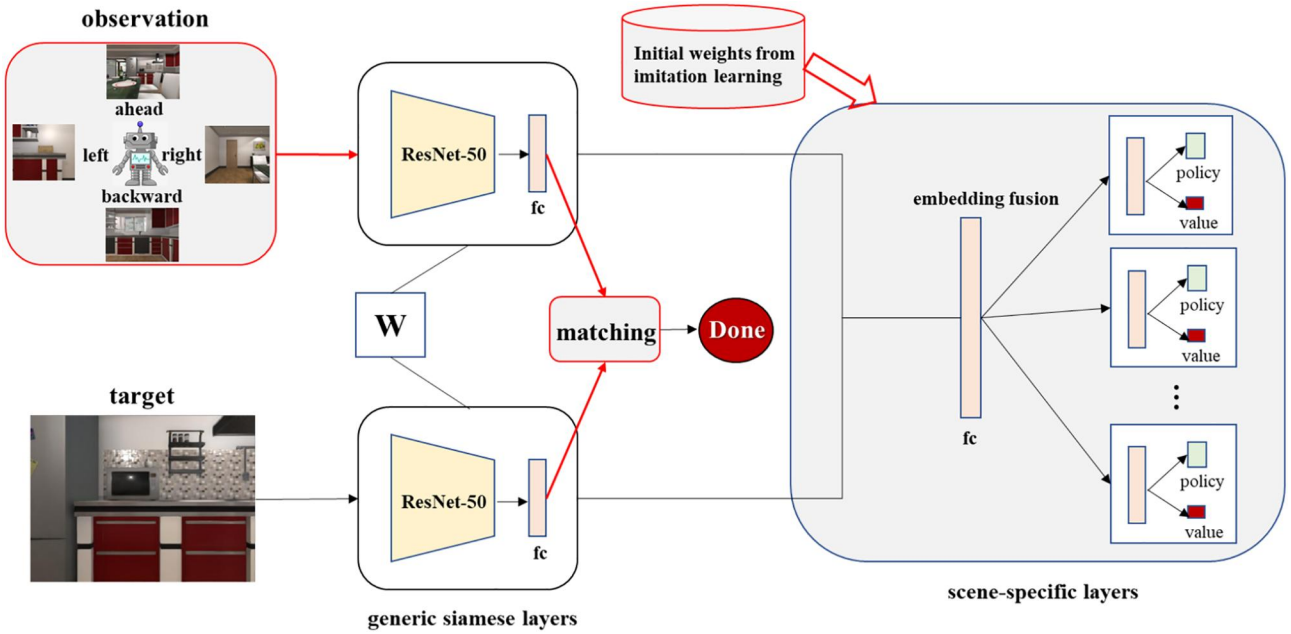

FIGURE 2 Network architecture of our DIRL model. The generic Siamese layers in black squares are shared by all the targets, and the feature extraction layers (ResNet-50) are pre-trained on ImageNet and frozen during training. Four surrounding images are adopted as inputs instead of four sequential images, and the initial parameters of the scene-specific layers are from imitation learning. Moreover, a simple yet efficient method of template matching is added to determine whether the task is finished or not. DIRL, deep imitation reinforcement learning

4 | EXPERIMENTS AND RESULTS

We evaluate our approach on visual navigation tasks based on the AI2-THOR [12] environment. The main purpose of visual navigation here is to find the minimum steps from a random initial location to the goal location. We first train and evaluate the navigation results using our model. Then, an additional experiment is performed to investigate the ability of our approach to transfer knowledge across new targets. All of the models are implemented in TensorFlow and trained on an Nvidia GeForce RTX 2080Ti GPU.

4.1 | The AI2-THOR framework

To efficiently train and evaluate our model, we use The House Of InteRactions (AI2THOR) framework proposed in [19], which is a good representation of the real world in terms of the physics of the environment. We use four scenes of the simulated environment [1]: bathroom 02, bedroom 04, kitchen 02 and living room 08. Note that the number of indoor areas is scalable using our model. Figure 3 illustrates some sample images from different scenes, and each scene space is represented as a grid world with a constant step size of 0.5 m. To be more specific, the following information is listed:

State space. In our experiments, the input includes two parts, $S = < F, T >$ and F, which represent first-person views of the agent, where the resolution of one image is $224 \times 224 \times 3$. In addition, T represents the goal. We use four surrounding images to represent the observation. Each image is embedded as a 2048D feature vector, which is pretrained on ImageNet with Resnet-50.

FIGURE 3 AI2-THOR environment. There are four common scene types: kitchen, living room, bathroom and bedroom, respectively

Action space. We consider five discrete actions: ahead, backwards, left, right and done. The robot takes 0.5 m as a fixed step length, and its turning angle is 90° in the environment. Note that the former four actions are trained by the proposed network, while the done action, which is the termination action, is obtained and determined by the template matching methods.

4.2 | Implementation details

For training, the navigation performance is performed on 20 goals randomly chosen from four indoor navigation environments in our data set. All of the learning models are trained within 10 million frames, and it takes about 1.5 h to pass through one million training frames across all threads. An episode ends when either the robot arrives at the goal location, or after it takes more than 5000 steps and for evaluation, we perform 4000 different episodes (1000 for each scene). In the matching process, the matching coefficient is set to 0.9. We select 5% of the total states as the expert data for IL training. The expert data is collected and stored automatically based on the shortest length to the target point and is selected in a fixed interval across the ID in each scene. The reward is set as 10.0 once the robot reaches the goal, and to encourage shorter trajectories and collision avoidance, the reward is set as −0.01 and −0.1 as an immediate reward.

4.3 | Evaluation metrics

The performance of our approach is evaluated using three metrics: the average trajectory length (ATL), success rate (SR) and success weighted by path length (SPL). All of the metrics are used to reflect information about navigation efficiency. ATL represents the average steps per episode it takes for a robot to reach a goal. SR is defined as $\frac{1}{N}\sum_{i=1}^{N} S_i$ and SPL is defined as $\frac{1}{N}\sum_{i=1}^{N} S_i \frac{L_i}{max(P_i,L_i)}$, where N is the number of episodes, S_i is a binary indicator of success in episode i, P_i denotes the evaluated step length and L_i is the shortest length of the trajectory.

4.4 | Baselines

Here, five baselines are chosen for comparison, which are shown as follows:

1. **Shortest path.** The shortest path is from the initial location to the goal location. Here, it is considered the ideal result.
2. **Random action.** The action is randomly sampled using a uniform distribution at each step. Here, the robot randomly selects the action from four actions during navigation.
3. **IL.** The output of the policy is directly determined using the networks trained by expert data.
4. **TD-DRL.** The original target-driven method using DRL was proposed in [12]. Targets use and update the same Siamese parameters but different scene-specific parameters.
5. **A3C.** It is an asynchronous advantage actor-critic approach. It has been demonstrated that the more threads the system uses, the higher the data efficiency during training. Here, we use four threads to train, and the observation is the same as TD-DRL.

4.5 | Results

In this subsection, we focus on three results: the sample efficiency during training, the navigation performance of our approach to the trained targets and the generalization to the new targets. Additionally, the baselines are evaluated for comparison.

4.5.1 | Sample efficiency

One of the main purposes of our proposed approach is to improve the sample efficiency during training. To analyse the sample efficiency in the training process, for all learning models, the training performances are measured by the ATL over all targets, and the performances are reported after being trained within 10 million frames (across all threads).

Figure 4 shows the performance of sample efficiency during training among different approaches: DIRL, TD-DRL and A3C. Note that the shortest path, random action and IL are not considered here because these approaches can be directly evaluated in the testing process, so they are used in the following navigation comparison. All of the approaches are trained in 10 million frames, and the curves in the figure represent the ATL of all training targets versus the training frames. It can be seen that it only takes approximately 1.5 million frames (2.25 h) to achieve a stable navigation policy in our approach. Although the ATL also decreases after some training frames in the other two approaches, it costs nearly 9.0 million frames and 6.1 million frames to train in A3C and TD-DRL, respectively. The reason that TD-DRL performs better than A3C is the specific layers used in TD-DRL, which was demonstrated in [12]. Therefore, the results indicate that the performance of our approach is the best and demonstrate the efficient learning performance of our approach.

4.5.2 | Navigation performance on trained targets

To analyse the navigation performance of our approach, we first analyse the performance on the trained targets. We compare our method with five baselines: shortest path, random action, IL, A3C and TD-DRL. Three evaluation metrics (ATL, SR and SPL) are used to reflect the navigation performance. For each target, we randomly select the initial location of the robot and evaluate 500 episodes. An episode ends when either the robot arrives at the target or after it takes more than 200 steps. We focus on the influence of the training frames, so two models are used where the parameters are trained in 5 and 10 million frames. The navigation results are given in Tables 1 and 2.

Table 1 shows the results on trained targets after 5 million training frames. Our approach performs best in terms of ATL, SR and SPL metrics. For ATL, the fewer steps taken, the better the performance is. The shortest path is approximately 12.66 steps, and our method is approximately 13.62 steps, which is the closest to the ideal result; however, it takes more navigation

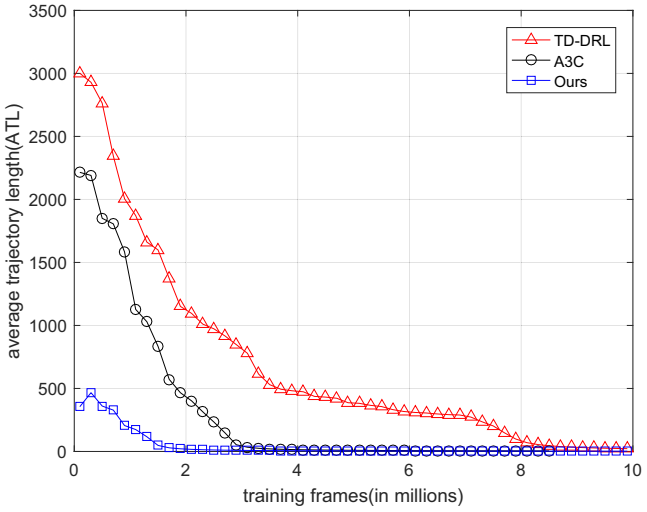

FIGURE 4 Comparison results of sample efficiency with the baselines: DIRL (blue), TD-DRL (red) and A3C (black). All of the approaches are trained in 10 million frames and the ATL of all targets is adopted to reflect the performance. In DIRL, it needs about 1.5 million frames to learn a stable navigation policy, however, it costs nearly 6.1 million frames and 9.0 million frames to train in TD-DRL and A3C, respectively, which indicates that our approach learns fastest to obtain a good navigation policy. ATL, average trajectory length; DIRL, deep imitation reinforcement learning approach; TD-DRL, target-driven deep reinforcement learning

TABLE 1 Quantitative results on trained targets after 5 million training frames

Algorithm metrics	ATL	SR	SPL
Shortest path	12.66	1.00	1.00
Random action	200	0.00	0.00
IL	146.19	0.19	0.01
A3C	179.55	0.32	0.10
TD-DRL	50.84	0.71	0.33
DIRL (ours)	**13.62**	**1.00**	**0.62**

Abbreviations: ATL, average trajectory length; DIRL, deep imitation reinforcement learning approach; IL, imitation learning; SPL, success weighted by path length; SR, success rate; TD-DRL, target-driven deep reinforcement learning.

steps in TD-DRL, A3C and IL, which are approximately 3.7 times, 13 times and 10.7 times those of ours, respectively. The SR is 1 using our approach, meaning the system can arrive at the goal in 200 steps because after 5 million training frames, our model has learned a good navigation policy, which can also be seen in Figure 4, while for the other methods, the SR is very low for imperfect learned policy, so does the performance of SPL. These results indicate that our approach can learn good performance faster.

We further analyse the navigation performance on trained targets after 10 million training frames, where both TD-DRL and A3C have learned good navigation policies. Table 2 shows the comparison results. We find that although the performances of SR are 1 in T-DRL, A3C and our approach, our approach DIRL performs the best in SPL, which is an approximately 6% improvement compared to TD-DRL and two times compared to that of A3C; the reason might be the different observations. The results further indicate the efficiency of our approach.

TABLE 2 Quantitative results on trained targets after 10 million training frames

Algorithm metrics	ATL	SR	SPL
Shortest path	12.66	1.00	1.00
Random action	200	0.00	0.00
IL	146.19	0.19	0.01
A3C	15.44	1.00	0.57
TD-DRL	14.56	1.00	0.70
DIRL (ours)	**13.51**	**1.00**	**0.76**

Abbreviations: ATL, average trajectory length; DIRL, deep imitation reinforcement learning approach; IL, imitation learning; SPL, success weighted by path length; SR, success rate; TD-DRL, target-driven deep reinforcement learning.

4.5.3 | Generalization across new targets

To analyse the ability of generalization for navigation, we evaluate our model to new targets for the visual navigation task. Note that these new targets are not trained in advance, but they might share the same routes as the trained targets. During testing, we randomly select eight new targets from a series of locations that have a fixed distance (one, two, four or eight steps) from the nearest trained targets. For each new target, we randomly select the initial location of the robot and evaluate 500 episodes. An episode ends when either the robot arrives at the target or after it takes more than 500 steps. We compare our method with A3C and TD-DRL and use the SR metric to analyse the navigation performance, as shown in Figure 5. We find that all the approaches have the ability to generalize new targets, but our approach performs the best under different conditions. This indicates that our learned model has a stronger ability to understand the surrounding regions of the trained targets than other approaches.

We can also intuitively analyse what systems have learned by the navigation policies using different approaches from Figure 6, which shows the explicit control state at three different steps. Note that the initial location and the goal are set the same among all of the approaches during testing. We take visual navigation in the bathroom as an example. It takes just 55 steps to arrive at the goal location using our approach; however, it is 151 steps and 432 steps in TD-DRL and A3C, respectively. Moreover, we can further investigate the policy of the system among different approaches. In TD-DRL, the robot rotates frequently, shown in three consecutive steps (t = 18, 19, 20). For A3C, the robot moves in the wrong direction frequently, which is also shown in three consecutive steps (t = 46, 47, 48). It can be seen that the control policy is good using our approach, which further indicates the generalizability of the system learned by our proposed approach.

4.5.4 | Ablation study

In this subsection, we perform ablations from different views on our approach to obtain further insight into the results.

To perform the action, we introduce an additional module using the template matching method to determine whether it is finished; however, in TD-DRL and A3C, the done action is automatically determined by the environment, there is no problem in the simulation environment, and we find that during the simulation environment, the matching coefficient is always equal to 1 once the robot arrives at the goal location because the actions are discrete and the scenes are stations in AI-THOR. However, in real-world navigation tasks, the matching coefficient is below 1 due to image noise or the dynamic environment; moreover, the navigation system should determine the done action. Therefore, our approach is more practical.

For the observation, Figure 7 shows the ablation results using two observations: sequential (red) and surrounding (black). Note that the former is the method proposed in TD-DRL, where four sequential images are considered as the input, and the latter is our proposed approach using four surrounding images. We still compare the sample efficiency, and both approaches are trained in 10 million frames, and the ATL of all targets is adopted to reflect the performance. The former needs nearly 6.1 million frames to learn a stable navigation policy, but it costs only approximately 2.3 million frames for our approach, which indicates that our proposed approach learns faster.

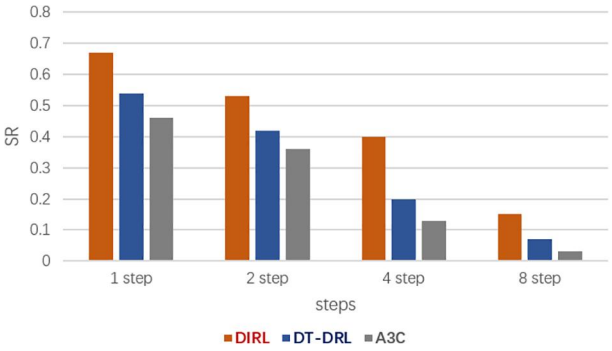

FIGURE 5 Generalization across new targets. Each histogram group represents the SR of navigation to new targets with fixed steps from the trained targets. It indicates that our learned model has a stronger ability to understand the surrounding regions of the trained targets than other approaches. SR, success rate

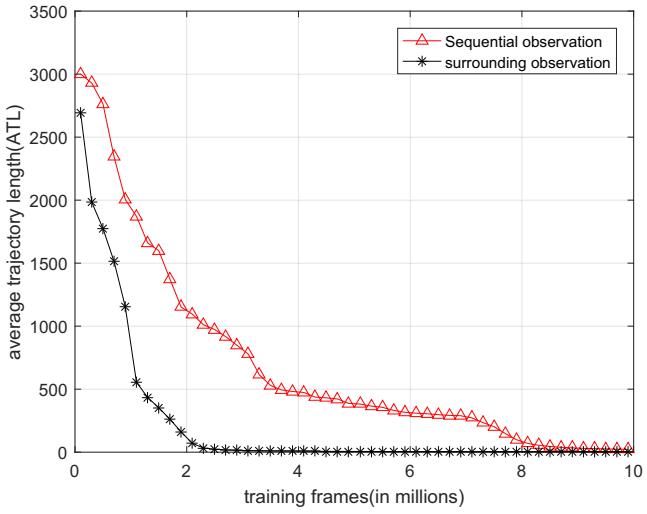

FIGURE 7 Observation ablation. Sample efficiency using different observation: surrounding observation(black) and sequential observation (red). We compare the sample efficiency and both approaches are trained in 10 million frames and the ATL of all targets is adopted to reflect the performance. To the former, it needs about 2.3 million frames to learn a stable navigation policy, however, it costs nearly 6.1 million frames for the latter, which indicates that the observation affect the navigation performance and our proposed approach learns faster. ATL, average trajectory length

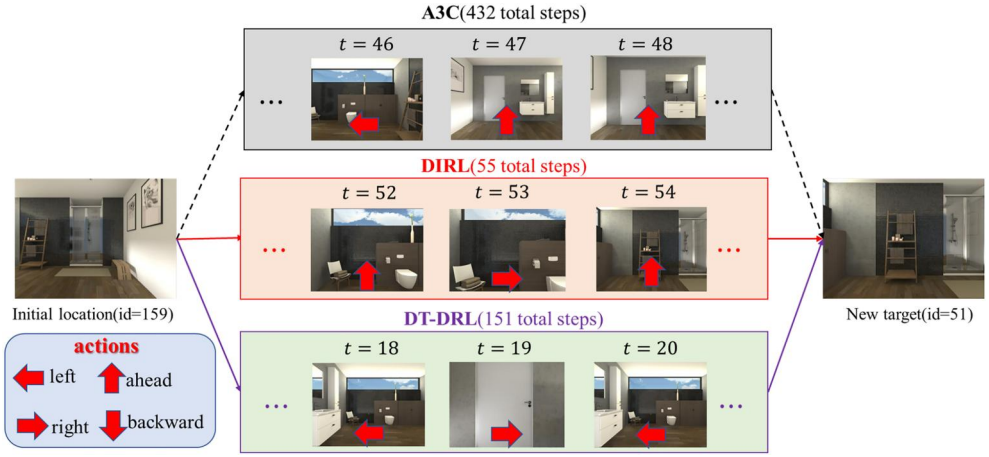

FIGURE 6 Visualization of the control processes. It takes just 55 steps to arrive at the goal location using our approach, however, it is 151 steps and 432 steps in TD-DRL and A3C, respectively. Moreover, the robot frequently rotates around in TD-DRL and moves to wrong direction in A3C. It can be seen that the control policy is almost good using our approach, which further indicates the ability of generalization of the system learned by our proposed approach TD-DRL, target-driven deep reinforcement learning

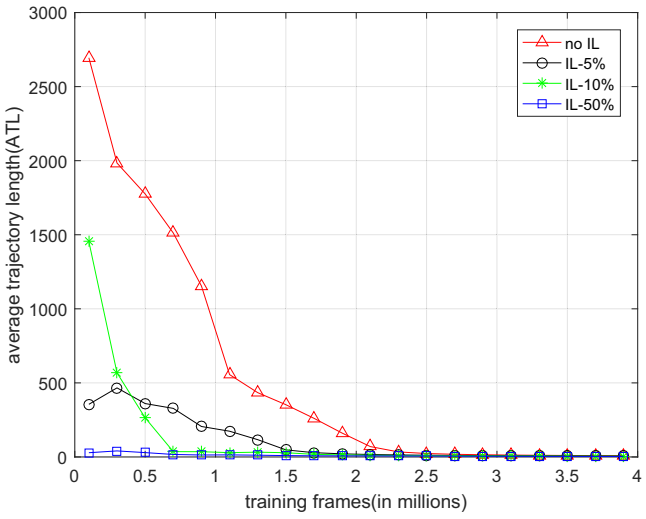

FIGURE 8 Number of expert samples ablation. We report the performances among different number of expert data: none, 5%, 10% and 50% of the total states. It can be seen that the sample efficiency improves according to the increasing number of expert data for IL, which needs about 2.3 million frames, 1.5 million frames, 0.7 million frames and 0.2 million frames to learn a stable navigation policy, respectively. It further indicates that it is useful to combine DRL with IL to deal with visual navigation tasks. DRL, deep reinforcement learning; IL, imitation learning

According to our intuition, the more expert samples we select from the state distribution, the better performance we might achieve during training. We report the performances among different numbers of expert data: none, 5%, 10% and 50% of the total states. Figure 8 shows the ablation results. It can be seen that the sample efficiency improves according to the increasing number of expert data for IL, which needs approximately 2.3 million frames, 1.5 million frames, 0.7 million frames and 0.2 million frames to learn a stable navigation policy. The different ATLs at the beginning are caused by the random initial locations and different navigation policies, but in the end, all approaches achieve nearly the same ATL. The results further indicate that it is useful to combine DRL with IL to deal with visual navigation tasks.

5 | CONCLUSIONS AND FUTURE WORK

Here, we proposed an approach for indoor visual navigation combining IL and DRL. The proposed navigation system improved not only the data efficiency during the training process but also the navigation performance. Unlike traditional visual navigation methods, which require calculating the robot's position accurately, our approach is an end-to-end method, so it might provide an alternative method for visual navigation. Moreover, simulation results demonstrate that the proposed DIRL approach is general and efficient compared with previous DRL methods, such as TD-DRL and A3C. Generally, the proposed approach can adapt to other DRL networks, such as DDPG and PPO. Our future work includes using the meta-learning method in our framework to improve the generalization to new scenes and evaluating our model in real-world navigation environments.

ACKNOWLEDGEMENT

The authors would like to thank Yichuan Zhang and Pinghai Gao for their helpful discussion and feedback. This study was supported by the National Natural Science Foundation of China (Grant nos. 61703418 and 61825305).

ORCID

Qiang Fang https://orcid.org/0000-0002-5063-6889

REFERENCES

1. Bonin-Font, F., et al.: Visual navigation for mobile robots: a survey, J. Intell. Robot. Syst. 53(3), 263–296 (2008)
2. Vakhitov, A., Lempitsky, V.: Learnable line segment descriptor for visual slam, IEEE Access, 7, 39923–39 934 (2019)
3. Davison, A.J., et al.: MonoSLAM: real-time single camera slam, IEEE Trans. Pattern Anal. Mach. Intell. 29(3), 1052–1067 (2007)
4. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces, In: Proceedings of the IEEE and ACM international Symposium on Mixed and Augmented Reality (ISMAR), pp. 225–234, Nara, 13–16 November 2007
5. Lupton, T., Sukkarieh S.: Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions, IEEE Trans. Robot. 28(1), 61–76 (2012)
6. Tang, P., et al.: Deep fishernet for image classification, IEEE Trans. Neural Netw. Learn. Syst. 30(7), 2244–2250 (2019)
7. Liu, L., et al.: Deep learning for generic object detection: a survey, Int. J. Comput. Vis. 128, 261–318 (2020)
8. Liu, Z., et al.: Deep learning Markov random field for semantic segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 40(8), 1814–1828 (2017)
9. Sutton, R.S., Barto, A.G.: Introduction to reinforcement learning. MIT Press, Cambridge, (2018)
10. Xu, X., et al.: Manifold-based reinforcement learning via locally linear reconstruction, IEEE Trans. Neural Netw. Learn. Syst. 28(4), 934–947 (2017)
11. Mnih, V., et al.: Playing Atari with deep reinforcement learning, arXiv preprint arXiv:1312.5602 (2013)
12. Zhu, Y., et al.: Target-driven visual navigation in indoor scenes using deep reinforcement learning, In: Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 3357–3364 (2017)
13. Hussein, E.E.A., et al.: Imitation learning: a survey of learning methods, ACM Comput. Surv. 50(2), 934–947 (2017)
14. Bojarski, M., et al.: End to end learning for self-driving cars, arXiv preprint arXiv:1604.07316 (2016)
15. Yang, Z., et al.: End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perception, In: Proceedings of the 24th International Conference on Pattern Recognition (ICPR), pp. 2289–2294 (2018). https://doi.org/10.1109/ICPR.2018.8546189
16. Wang Q., et al.: End-to-end autonomous driving: an angle branched network approach, IEEE Trans. Veh. Technol. 68(12), 11599–11610 (2019)
17. Codevilla, F., Mãijller, M.: End-to-end driving via conditional imitation learning, In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 4693–4700, Brisbane, 21–25 May 2018
18. Xu, H., et al.: End-to-end learning of driving models from large-scale video datasets, In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3530–3538, Honolulu, 21–26 July 2017

19. Mnih, V., et al.: Human-level control through deep reinforcement learning, Nature. 518(7540), 529 (2015)
20. Van Hasselt, H., et al.: Deep reinforcement learning with double q-learning, In: Proceedings of the 30th AAAI conference on artificial intelligence, pp. 2094–2100 (2016)
21. Schaul, T., et al.: Prioritized experience replay, arXiv preprint arXiv: 1511.05952 In: *International Conference on Learning Representations*, (2016)
22. Wang, Z., et al.: Dueling network architectures for deep reinforcement learning. In: Proceedings of The 33rd International Conference on Machine Learning, arXiv preprint arXiv:1511.06581 pp. 1995–2003 (2016)
23. Gu, S., et al.: Continuous deep q-learning with model-based acceleration, In: Proceedings of the International Conference on Machine learning, pp. 2829–2838 (2016)
24. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning, Computer Science, (2015)
25. Mnih, V., et al.: Asynchronous methods for deep reinforcement learning, In: Proceedings of the International Conference on Machine Learning, pp. 1928–1937 (2016)
26. Mirowski, P., et al.: Learning to navigate in cities without a map, In: Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS), pp. 2419–2430 (2018)
27. Zhang, J., et al.: Deep reinforcement learning with successor features for navigation across similar environments, In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2371–2378 (2017)
28. Chiang, H.-T.L., et al.: Learning navigation behaviors end-to-end with AutoRL, IEEE Robot. Autom. Lett. 4(2), 2007–2014 (2019)
29. Dulac-Arnold, G., Mankowitz, D., Hester, T.: Challenges of real-world reinforcement learning, arXiv preprint arXiv:1904.12901 (2019)
30. Zhu, Y., Wang, Z.: Reinforcement and imitation learning for diverse visuomotor skills, In: Proceedings of the Robotics: Science and Systems, Pittsburgh (2018)
31. Chen, X., et al.: Bail: best-action imitation learning for batch deep reinforcement learning, In: Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), vol. 33, Vancouver, (2019)
32. Reddy, S., Dragan, A.D., Levine, S.: SQIL: imitation learning via reinforcement learning with sparse rewards, In: Proceedings of the 2020 International Conference on Learning Representations(ICLR), (2020)
33. Huo, Y., et al.: Cooperative control for multi-intersection traffic signal based on deep reinforcement learning and imitation learning, IEEE Access. 28(1), 199 573–199 585 (2020)
34. Tsai, D.-M., Lin, C.-T.: Fast normalized cross correlation for defect detection, Pattern Recogn. Lett. 24(15), 2625–2631 (2003)

How to cite this article: Fang, Q., et al.: Target-driven visual navigation in indoor scenes using reinforcement learning and imitation learning. CAAI Trans. Intell. Technol. 7(2), 167–176 (2022). https://doi.org/10.1049/cit2.12043