Research on Target-Driven Navigation of Mobile Robot Based on Deep Reinforcement Learning and Preprocessing Layer

Gang Yu, Chang Zhang, Jiexiong Xu and Tongda Sun

Harbin Institute of Technology, Shenzhen, School of Mechanical Engineering and Automation, H 607, Shenzhen University Town HIT Campus, Shenzhen, P. R. China. Email: gangyu@hit.edu.cn

Abstract. Recently, with the rise of deep reinforcement learning model, robot navigation based on this method has a huge advantage compared with traditional slam method, which has attracted extensive attention. However, when the navigation algorithm trained in the virtual environment is transferred to the real environment, the navigation performance of the robot will decline sharply because of the great difference between the virtual environment and the real environment. In order to improve the navigation ability of mobile robot, this paper implements a mobile robot navigation system based on deep reinforcement learning without environment map and only visual input. At the same time, in order to solve the problem of poor generalization ability of deep reinforcement learning from virtual environment to real environment, this paper proposes a preprocessing layer with knowledge and combines it with deep reinforcement learning module. The combined algorithm model alleviates the performance fault problem caused by the migration algorithm and the performance difference between virtual sensor and real sensor. At the end of this paper, a navigation experiment based on the turtlebot is designed, which proves that the deep reinforcement learning algorithm with the preprocessing layer can alleviate the performance fault problem caused by the migration algorithm, and have a certain ability of obstacle avoidance and avoidance without the environment map.

1. Introduction
The traditional slam method has many problems which seriously affect its performance in robot[1]. Some papers show the feasibility of using deep reinforcement learning to train navigation ability in real robot [2], but the cost of training in real environment is very large, so scientists often use virtual environment training instead of real environment training, but this brings another problem: because the virtual environment is very different from the real environment, the trained navigation algorithm in the virtual environment is transferred to solve this problem, this paper proposes an improved navigation method based on deep reinforcement learning.

Most of the robot navigation work based on deep reinforcement learning is based on end-to-end training[3], that is, the navigation system based on deep reinforcement learning directly receives the environmental state information, and outputs actions directly after a feedforward operation. In order to solve the problem of environmental migration, this paper adds a preprocessing layer with knowledge before the deep reinforcement learning system. After receiving the environment input information, the system processes it and then outputs the information with knowledge from the preprocessing layer to the deep reinforcement learning system, which outputs the robot’s action. The main purpose of this preprocessing layer is to make the virtual environment and the real environment output the same state information after preprocessing layer for the use of the standard deep reinforcement learning algorithm, and then output the same action. In this way, only need to train the navigation strategy in the virtual
environment, the input of the navigation module will not be affected by the environment changes, once the environment changes, only need to retrain the preprocessing layer. In order to illustrate the idea of the preprocessing layer, this paper explains the preprocessing layer with an example of a digital wall environment as shown in Figure 1.

![Figure 1. Virtual training environment.](image)

2. Preprocessing Layer and Deep Reinforcement Learning Module

The preprocessing layer of this task is composed of image segmentation layer and image recognition layer. In the image segmentation layer, firstly, the image is binarized, and then the image is blurred by Gaussian filtering:

$$G(x,y) = \sum_{x-m}^{x+m} \sum_{y-m}^{y+m} \exp \left[ -\frac{x^2 + y^2}{2\sigma^2} \right], \quad m = \frac{n-1}{2}$$

(1)

Figure 2 shows the outputs of sementation layer

![Figure 2. Image segmentation diagram.](image)

The segmented image is input into the image recognition module. In the image recognition module, the numbers to be recognized is 1-9, 0 is the exception detection bit, so the output is 10 dimensions, the input image is reshaped to $32 \times 32$ size, so a CNN based image recognition network is designed. The first two layers of CNN are convolution layer and down-sampling layer. Convolution layer performs convolution operation on input data to create feature map. It uses local receptive domain (small size filter) and shared weight (filter with the same size) to achieve the invariance of the output when the input data is distorted. The output of convolution operation is input to sub sampling layer through nonlinear activation function. The sub sampling layer performs local averaging to reduce the dimension of the forward feature map, while maintaining the distortion invariance of the data. The integration of convolution layer and subsampling layer can be used in series in the architecture of convolution neural network. The output of the final subsampling layer is sent to the fully connected layer for classification or identification tasks.

After the image is processed by the preprocessing layer, the outputs is a six-dimension vector. The first dimension is the block diagram size of the obstacle in the image, the second dimension is the position of the obstacle in the transverse decile of the image, and the third dimension is used to record the left or right vanishing positions if the obstacle disappears. The fourth dimension is the block
Diagram size of the target, the fifth dimension is the location of the target in the image transverse decile, and the sixth dimension is used to record the left or right vanishing positions if the target disappears.

The information processed by the preprocessing layer will be input into the deep reinforcement learning module. The input is a six-dimensional feature, followed by two fully connected layers, each with 50 and 20 neurons, and the output is a three-dimensional vector, corresponding to a predefined three-dimensional vector space. The Adam optimizer is used to train the above network, and the learning rate is $(1 \times 10)^{-4}$.

In this module, the robot is trained by interacting with the environment, that is, by executing actions in the environment and receiving rewards at the same time, and constantly repeating the process. The training goal is to select the actions that can get the maximum reward in the current environment. At each time point of a group of discrete time (the interval is time step T), the robot inputs the state $s_t \in S$ (S is a collection of possible states) and outputs the action $a$ from a group of action sets $A = 1, 2 \ldots k$. Among them, the state machine that obtains the output according to the input is called the strategy. As a result of each action, the robot receives a scalar reward $r_t \in R$, and observes the next state after one step $s_{t+1} \in S$. The probability of every possible next state $s_{t+1}$ comes from the probability distribution of state transition, which is characterized as $P(s_{t+1}|s_t,a_t); s_{t+1} \in S, a_t \in A(s_t)$. Similarly, the probability of each possible reward comes from the probability distribution of the reward $P(r_t|s_t,a_t); s_t \in S, a_t \in A(s_t)$. Therefore, through the probability distribution expectation of reward, the expected reward $E[r_t|s_t,s_{t+1} = s, a_t = a]$ received for performing action $a$ in the current state $s$ can be calculated. The purpose of the robot is to learn the best strategy to maximize the reward $R$:

$$R_t = E \left[ r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \right] = E \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \right]$$

(2)

In the task of robot visual navigation, as shown in Figure 3, the area size of the bounding box containing the target object is used as the reward. For the environment state obtained by the robot at each training time point, if the robot detects the target, the reward is defined as the size of the detected bounding box containing the target object: $r = S_{\text{target}}$. For the environment that the robot unable to detect the object, the robot receives a negative reward $r = -0.01$ as a penalty, that is, the longer the robot travels, the more negative rewards accumulate. When the size of the bounding box of the task target object is greater than 100, the task is completed, and the bounding box area is returned as reward, the completion signal is returned and the task is resumed.

**Figure 3.** Target number border size as reward.

Action value function $Q^\pi(s, a)$ is defined as follows:

$$Q^\pi(s, a) = E_{\pi} \left( R_t | s_t = s, a_t = a \right) = E_{\pi} \left( \sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a \right)$$

(3)

Use the following iterative update to estimate the action value function:
\[ R_t = E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots] = E \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \right] \] (4)

Then the gradient descent method is used to get the best action function \( Q^*(s, a) \).

In addition to the above-mentioned basic technologies, this paper also uses a technology called experience pool[4], which stores the feedback from the current environment state and the state from the next time point, in dataset D, and places the corresponding dataset of each round in the memory. During the implementation of the algorithm, samples are randomly selected from the sample pool to update the network parameters. After updating the network parameters, the robot chooses actions and executes them according to the greedy strategy.

The learning goal of deep reinforcement learning is to learn the random strategy function \( \pi \). The mobile robot continuously repeats the actions output from the strategy according to the environment until it reaches the destination. In this way, the generation of action sequence is based on the environment state and task goal. No matter how the goal changes, the input of deep reinforcement learning module is always the relative position information of the goal in the field of vision, and no need to retrain the deep reinforcement learning module if we have a new navigation goal. The ability of a robot to output its own sequence of actions can be used as a measure of the intrinsic curiosity of the robot itself[5]. The bottom mechanism must be considered when the real world mobile robot executes the action instructions. However, a variety of mechanical mechanisms and their physical effects will make training more complex[6]. A common way to solve this problem is to train the robot model's motion module at a certain abstract level and the underlying mechanism is handled by other lower level controllers. Using simple operations at a high level of abstraction to train a model. For the above training tasks, the robot actions are discretized. There are three different actions in the robot's action space: 1) forward at a fixed distance of 0.5m; 2) counter clockwise at a fixed angle of 45 degrees; 3) clockwise at a fixed angle of 45 degrees. In order to simulate the uncertainties in real dynamic system, Gauss noise \( \mathcal{N}(0,0.1) \) is added to the forward motion and the Gauss noise \( \mathcal{N}(0,1.0) \) is added to the left turn and right turn actions. This algorithm takes about 0.5 seconds from the input environment information to the output action of the model.

3. Simulation Experience
In order to verify the navigation ability of the system, the following tasks are designed: Find the given digital object in the simulation scene of Figure 4, and set obstacles between the robot and the target.

![Figure 4. Scenes with obstacles.](image)

For each experimental setup, due to the limited conditions, this paper compares the performance of the proposed navigation algorithm with the random walk model. However, the proposed algorithm will not be compared with the classical search algorithm, such as A-star algorithm, depth first or breadth search algorithm, because it is assumed that the global map of the environment is unknown, and the actions taken by the robot may lead to collision, and this uncertainty will make these deterministic algorithms unusable.

In order to reduce the randomness of the environment and make a fair comparison, for each target object, the initial position of the robot is initialized randomly. When the robot finds the target object, that is, the task is executed successfully, or it has executed 5000 steps, that is, the task fails to execute, and one task is finished.
Figure 5. A comparison of the number of steps of tasks performed by random walk model and deep reinforcement learning model based on preprocessing layer.

Figure 5 shows that the random walk model does not have the ability to complete the specified navigation target, but the combination algorithm of deep reinforcement learning with preprocessing layer has a certain navigation ability for the specified target in the indoor environment with obstacles, and has a certain ability to avoid obstacles.

Figure 6 shows the virtual environment and the preprocessing results in this environment.

Experiments show that deep reinforcement learning with preprocessing layer have navigation ability in virtual environment and have a generalization ability to recognize the goal. The trained mobile robot can only use the information of vision input and target object, and spend relatively less time and less collision times in the environment to navigate from its random starting position to the vicinity of the target object.

After training the navigation strategy in the virtual environment, there is no need to rebuild the virtual scene for different scenes, only to retrain the preprocessing layer. In order to verify whether the navigation system with environment knowledge has navigation ability in the real environment after training in the virtual environment, experiments will be carried out in the real environment on the turtlebot.

The real world scene is set up in the laboratory. Turtlebot is equipped with RBG camera and navigation system trained in the virtual environment. The task is set as turtlebot to avoid fire in real environment and search for the target to be rescued, so as to verify the navigation effect of environmental knowledge preprocessing layer and deep reinforcement learning in real environment.

As shown in Figure 7, in order to train the preprocessing layer, the labeled object bounding box is used as the dataset. In the robot task, the location and angle of the image are randomly collected, and the bounding box of each object in each image is manually marked. The coordinate value and area size of the marked bounding box are extracted as the output mark to train the neural network.
Figure 7. Image segmentation and recognition in real environment. After the preprocessing layer is trained, the robot has navigation ability. Figure 8 is the decomposition diagram of the robot performing a task.

Figure 8. Execution task breakdown.

4. Conclusion
It can be concluded that the robot trained in the virtual environment can recognize the flame as the obstacle, bypass the obstacle, recognize the human as the target, and finally navigate to the vicinity of the human target. It proves that the deep reinforcement learning algorithm with the preprocessing layer have a certain ability of obstacle avoidance and avoidance without the environment map.

5. References
[1] L Moreno, S Garrido, D Blanco, and M L Muñoz, Differential evolution solution to the SLAM problem[C] Robotics and Autonomous Systems vol. 57 pp. 441-450 2009.
[2] Leonard, H F and Durrant-Whyte Mobile robot localization by tracking geometric beacons[C] Robotics and Automation IEEE Transactions on vol. 7 pp. 376-382 1991.
[3] S Levine, C Finn, T Darrell and P Abbeel, et al. End-to-end training of deep visuomotor policies[C] JMLR 2016.
[4] V Mnih, K Kavukcuoglu, D Silver, A A. Rusu, J Veness, M G. Bellemare, A Graves, M Riedmiller, A K Fidjeland and G Ostrovski et al. Human-level control through deep reinforcement learning[C] Nature 2015.
[5] G Dissanayake, S Huang, Z Wang, and R Ranasinghe A review of recent developments in Simultaneous Localization and Mapping[C] in 6th International Conference on Industrial and Information Systems (ICIIS) Sri Lanka 2011. 477-482.
[6] Krishan Rana, Vibhavari Dasagi, Ben Talbot, Michael Milford and Niko Sünderhauf Multiplicative Controller Fusion: A Hybrid Navigation Strategy For Deployment in Unknown Environments arXiv:2003.05117 [cs.RO]