Mobile Robot Navigation based on Deep Reinforcement Learning

Xiaogang Ruan^{1,2}, Dingqi Ren^{1,2}, Xiaoqing Zhu^{1,2*},Jing Huang^{1,2}

1. College of Automation, Beijing University of Technology, Beijing, 100124, China

2. Beijing Key Laboratory of Computational Intelligence and Intelligent System, Beijing, 10024, China
E-mail: adrxg@bjut.edu.cn,1363957429@qq.com,alex.zhuxq@bjut.edu.cn,huangjing@bjut.edu.cn

Abstract: Learning to navigate in an unknown environment is a crucial capability of mobile robot. Conventional method for robot navigation consists of three steps, involving localization, map building and path planning. However, most of the conventional navigation methods rely on obstacle map, and dont have the ability of autonomous learning. In contrast to the traditional approach, we propose an end-to-end approach in this paper using deep reinforcement learning for the navigation of mobile robots in an unknown environment. Based on dueling network architectures for deep reinforcement learning (Dueling DQN) and deep reinforcement learning with double q learning (Double DQN), a dueling architecture based double deep q network (D3QN) is adapted in this paper. Through D3QN algorithm, mobile robot can learn the environment knowledge gradually through its wonder and learn to navigate to the target destination autonomous with an RGB-D camera only. The experiment results show that mobile robot can reach to the desired targets without colliding with any obstacles.

Key Words: Robot Navigation, Deep Reinforcement Learning, Depth Image

1 INTRODUCTION

Robots are playing an increasingly important role in our daily life, like cleaning, rescuing, mining, unmanned driving and so on. Its a fundamental ability for robots to navigate autonomously in an unknown environment. Benefiting of the development of artificial intelligence and computer vision, great progress has been made on intelligent robot technologies. However, enabling robots to navigate in a real world autonomously is still a challenging task. Traditional navigation method consists of localization, map building, and path planning. Most of the methods up to now rely on metric or topological map designed by human built based radar or camera information. However, ranging sensor like radar usually capture limited information and they are expensive. On the other hand, RGB-D cameras provide more information about the environments.

In the recent years, deep learning has successfully used in computer vision, automatic speech recognition and natural language processing. Deep learning, developed from artificial neural network (ANN), was proposed by Hinton et al. [1] in 2006s. ANN was inspired by biological nervous system and proposed in 1940s by McCulloch and Pitts [2]. ANN aims to realize Artificial Intelligence by mimicking the neural network of human. Typically, deep neural network has more hidden layers than typical artificial neural network. So, it can solve abstract representation problems like human. Deep learning shows a great performance in feature representations and realize a human level intelligence. Deep learning extracts feature representations autonomously rather than designs by human in traditional methods like support vector machine. Based on the great performance of deep learning, several end-to-end learning methods were proposed for robot to avoid obstacles, environment exploration and navigation. However, robot makes decision by learning from a huge amount of labeled samples, other than by learn from the environment like human.

In 2013s, Google DeepMind proposed deep reinforcement learning (DRL) [3], and implement that agents can learn knowledge by interacting with the environment, without expert operation as supervision information. Through reward function, agent learns decision-making polices based on the raw image observations. Without any prior knowledge with Atari games, this method outperformed almost all of the state-of-the-art methods and human players.

In this paper, we propose an end-to-end method for mobile robot navigation in an unknown environment using depth image information only. The model is validated in Gazebo environment with ROS-based interfaces. And the result indicates the feasibility and efficiency of the method. The remainder of the paper is organized as follows: Section will review the recent related works in deep learning based decision and deep reinforcement learning based decision-making. Section introduces the proposed model, deep reinforcement learning based navigation. Experimental results are given in section . In addition, the conclusions drawn in the last Section.

2 RELATED WORK

2.1 Deep-Learning-based Navigation

Science the success of deep learning in ImageNet Classification with deep convolutional neural networks [4] in 2012, deep learning has been obtained tremendous progress in

This work is supported by Beijing Natural Science Foundation(4174083);National Natural Science Foundation of China(No. 61773027);Key Project(No. KZ201610005010)of S&T Plan of Beijing Municipal Commission of Education.

several domains. Deep learning has a huge potential in solving robot control problems end-to-end with raw observation information. Deep learning has been successfully applied in robot navigation using camera and LiDAR. Kim et al. [5] from Cornell University trained a controller strategy by Convolutional Neural Network that mimics an expert pilots choice of action using image information from a single camera to accomplish that a quadcopter navigates indoors and finds a specific target autonomously. Wang et al. [6] trained end-to-end controller for robot navigation using convolutional neural network, which directly outputs the velocity and angular rates using the current observation and the target with LiDAR sensor. Kim et al. [7] proposed a visuo-motor navigation system, which used RGB images as input to learn velocity controller, driving robot navigate in a real-world environment with low velocity error.

Deep learning has shown great probability in robotics, and good navigation performance. However, all the methods mentioned above were learning actions with labeled observations, but not learn by itself from the environment. It costs many efforts on the labeling work and has not a good generalization performance. Therefore, it is significant for us to study the method that do not need labeled data.

2.2 Deep-Reinforcement-Learning-based Navigation

Deep Reinforcement Learning was proposed by Minh et al. [3] in 2013 playing Atari games. The agent successfully learned control policies directly from images and surpassed a human expert on some games. Tai et al. [8] utilized Deep Q-Networks (DQN) with depth image from RGB-D sensor by estimating q value corresponding to all moving commands realizing a robot exploration without any collision. Tai et al. [9] proposed a DQN approach using features trained by Convolutional Neural Networks from depth image information as input. After training a certain number of times, the robot can travel in new environment autonomously. To solve the disadvantage of over estimation of DQN, Deep Reinforcement Learning with Double Q-learning (Double DQN) was proposed in 2015 by Hasselt et al. [10], which reduces the observed overestimations by decouple the selection from the evaluation. Schaul et al. [11] developed a framework for prioritizing experience, which use prioritized experience replay in DQN. This method learns more efficiently by considering the significance of the experiences. Xie et al. [12] proposed an approach called a dueling architecture based deep double Q network (D3QN) which combined Double DQN and Dueling Network Architectures for Deep Reinforcement Learning (Dueling DQN) [13]. However, the DQN algorithm mentioned above can be used only in tasks with a discrete space. In some conditions, the control problem is continuous. Tai et al. [14] adapted Asynchronous Deterministic Policy Gradients (ADDPG) [15] based Deterministic Policy Gradients (DDPG) [16] using LiDAR sensor only navigating the robot to the desired targets without map.

3 IMPLEMENTION OF DRL NAVIGATION

In this section, we concentrate on the target of navigating in an unknown indoor environment. Meanwhile, the robot should avoid any collision with the obstacles. The input information is the raw depth image with size scale only, and output the linear and angular velocities.

Table 1: Algorithm 1

```
Dueling Double DQN Algorithm
Initialize the weights of evaluation networks as $\theta$,$\theta^{'}$
Set the memory D to store experience replay
Set replay buffer maximum size $N_r$, training batch size $N_{b'}$
and target network replacement frequency $N_{rf}$
Set the collision distance threshold $d_{arrive}$,$d_{collision}$
for episode = 1,M do
    Set the Turtlebot to the start position
    Get the minimum intensity of depth image as $d_t$
    while $d_t > d_{arrive}$ or $d_t > d_{collision}$ do
        Capture the depth image $s_t$
        With probability $\varepsilon$ select a random action $a_t$
        Otherwise select $a_t = argmax_a Q(s_t, a, \theta)$
        Move with the selected moving command $a_t$
        Update $d_t$ with new depth information
        if $d_t < d_{arrive}$ then
            $r(s_t, a_t) = r_{arrive}$
        else if $d_t > d_{arrive}$ or $d_t < d_{collision}$ then
            $r(s_t, a_t) = r_{collision}$
        else
            $r(s_t, a_t) = c(d_{t-1} - d_t)$
            Capture the new depth image $s_{t+1}$
        end if
        Store the transition $(s_t, a_t, r_t, s_{t+1})$ in D, Replacing
        the oldest tuple if $|D| \geq N_r$
        Select a batch of $N_b$ transitions $(s_k, a_k, r_k, s_{k+1})$
        randomly from D
        for each transition:
            $a^{\max}(s_{k+1};\theta) = argmax_{a'} Q(s_{k+1}, a^{'}; \theta)$
            if $r_k = r_{arrive}$ or $r_k = r_{collision}$ or $r_k = r_{timeout}$ then
                $y_k = r_k$
            else
                $y_k = r_k + \gamma Q(s_{k+1}, a^{\max}(s_{k+1};\theta); \theta^{'})$
            end if
            Update  through a gradient descent with loss
            $(y_k - Q(s, a; \theta))^2$
            Replace target parameters $\theta^{'}$ with $\theta$ every $N_{rf}$ steps
    end while
end for
```

3.1 Dueling Double DQN

Dueling Double DQN is a combination of Double DQN and Dueling DQN. As a result, it overcomes the estimation problem and improves the performance. As the standard reinforcement learning method, the learning sequences s is regarded as a Markov Decision Process (MDP). The robot makes decisions by interacting with the environment.At each time step, the robot chooses an action a_t according to the current observation s_t at time t firstly, where the observation is a depth image stack consists of four image frames. And then observes an reward signal $\mathrm{r}(\mathrm{s}_t, a_t)$ produced by reward function. In this paper, the actions contain moving forward, turning half left, turning left,turning left right, turning right. Lastly, the robot transits to the next observation s_{t+1}. The accumulative future reward is

$$\mathrm{R}_t = \sum\nolimits_{\tau}^{T} \gamma r_\tau \tag{1}$$

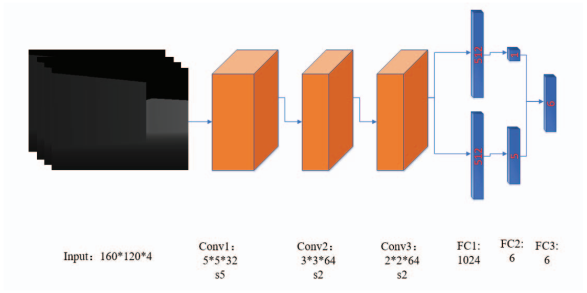

Figure 1: Network architecture of Dueling DDQN. It contains a 3-layer CNN and the dueling network.

, and the robot is aimed to maximum the discounted reward.In this formulation, γ is a discount factor between 0 and 1 that trades-off the importance of immediate and future rewards. The smaller γ is, the more important immediate will be, and vice visa. T means the termination timestep. The target of the algorithm is to maximize the action value function Q. Compared with DQN, the Q function of Dueling DQN is

$$Q\left(s, a; \theta, \alpha, \beta\right) = \mathrm{V}(s, \theta, \beta) + \\ \left(A(s, a; \theta, \alpha) - \max(s, a; \theta, \alpha)\right) \tag{2}$$

Here, denotes the parameters of the convolutional layers, while and are the parameters of the two streams of fully connected layers. The loss function to train the Q-network is

$$L(\theta) = \frac{1}{n}\sum\nolimits_{k}^{n}\left(y_k - Q(s, a; \theta)\right)^2 \tag{3}$$

3.2 Network Structure

The network structure is shown in Fig. 1. It is composed of the perception network and the control network. The perception network layer is a 3-layer convolutional neural network (CNN), including convolution and activation in every layer. The first CNN layer uses 32 convolution kernels of size 5 x 5, with stride 2. The second layer uses 64 convolution kernels of size 3 x 3 with stride 2. The third layer uses 64 convolution kernels of size 2 x 2 with stride 2.The control network is the dueling network. The dueling network has two sequences of fully connected (FC) layers to separately estimate state-value and the advantages for each action. In FC1 layer, there are two FC layers consists of 512 nodes respectively. In FC2 layer, there are two FC layers consist of one node and six nodes respectively. In FC3 layer, there is a FC layer consists of six nodes.activation function all of the layers use ReLu function. The parameters of Dueling Double DQN model are given in Table 1.

3.3 Model Settings

To train the network to the target position and avoid any collision, reward function is necessary. The instantaneous reward function is defined as

$$r(s_t, a_t) = c(d_{t-1} - d_t) \tag{4}$$

Table 2: Parameters of Dueling Double DQN

Layer	Size	Stride
Conv1	5x5x32	5
Conv2	3x3x64	2
Conv3	2x2x64	2
FC1(action-value)	512	-
FC1(advantage)	512	-
FC1	5	-
FC2	1	-
FC3	6	-

, and the termination reward function is defined as

$$\begin{aligned} r(s_t, a_t) &= r_{arrive} \quad if \quad d_t < d_{arrive} \\ r(s_t, a_t) &= r_{collision} \quad if \quad d_t < d_{collision} \\ r(s_t, a_t) &= r_{timeout} \quad if \quad t > t_{max} \end{aligned} \tag{5}$$

, where s_t, a_t and d_t are the state of robot, the action chosen and the distance from the target point respectively and c is a hyper-parameter. The instantaneous reward function is designed to motivate the robot to get closer to the target position and the total episode reward is the accumulation of instantaneous of all steps within an episode.A r_{arrive} is arranged if the robot arrives at the target position. But if a collision is detected, the episode terminates immediately and a punishment of $r_{collision}$ is arranged.Otherwise, the episode lasts until it reaches the maximum number of steps and terminates with punishment of $r_{timeout}$.

4 EXPERIMENT AND RESULTS

In this section, we evaluated the Dueling DDQN model in two simulated environments, one simple and another complex built in Gazebo simulator.

4.1 Simulated Environments

To validate the ability of the autonomous navigation of our system, we established two indoor environments with obstacles in Gazebo simulator as shown in Fig. 2 and Fig. 3.One is simple (Env1) with few obstacles and another is complex (Env2) with much more obstacles. In both environments, Turtlebot is placed at the same place and the cube represents the navigation target. Besides, the environments are of size 8 x 8 m^2. The robot platform we use is Turtlebot. Turtlebot is a wheeled mobile robot with two wheels that driven by Kobuki and a Kinect sensor which can collect RGB images and depth images simultaneously.

4.2 Simulated Environments

The depth image collected from is 640x480 pixels as shown in Fig. 4. Here, we preprocess it in size of 160x120 pixels so that the computational cost will reduce largely. The maximum learning iterations M is set as 3000 and 6000 respectively. And is set as 0.99, which means that emphasizes the importance of current reward. And is decay from 0.1 (Initial_Epsilon) to 0.0001 (Final_Epsilon) according to equation (6).

$$\begin{aligned} Epsilon &= Epsilon \\ &\frac{Initial\_Epsilon - Final\_Epsilon}{M} \end{aligned} \tag{6}$$

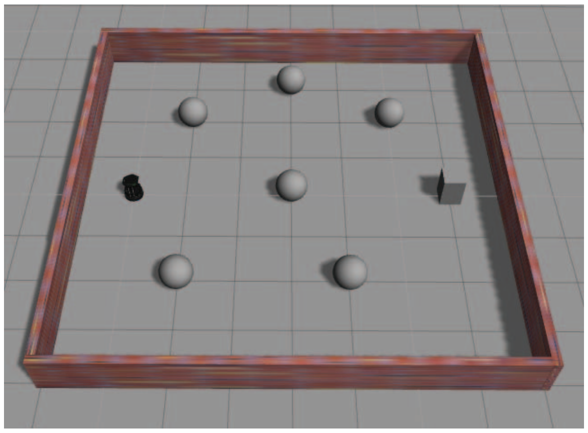
Figure 2: The simple environment (Env1) simulated in Gazebo. The size of the environment is 8m both in length and width. It consists of an experimental agent called Turtlebot, six ball obstacles and a target cube.

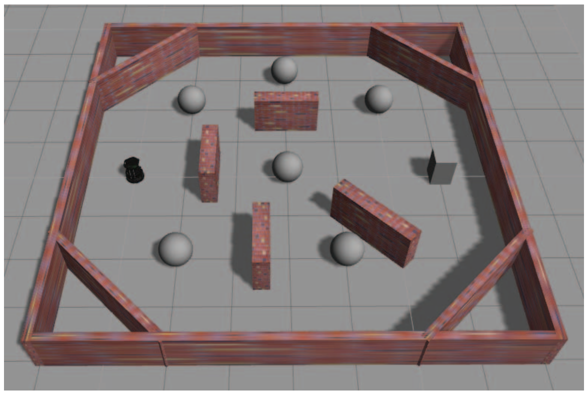
Figure 3: The complex environment (Env2) simulated in Gazebo. The size the same as Fig. 2. Besides the robot, obstacles and target in Fig. 1, it consists more cuboid obstacles.

Figure 4: Depth image collected from Kinect camera.

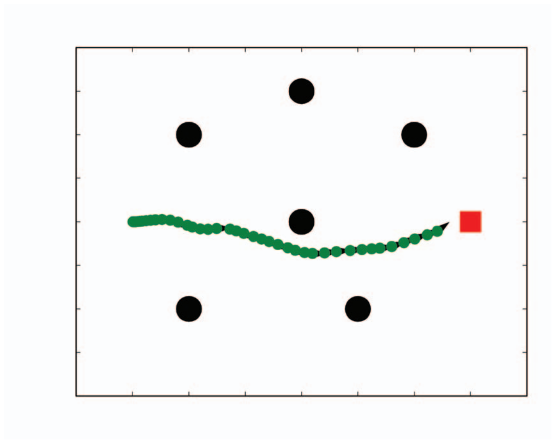
Figure 5: Navigation in simple environment. The green dot, arrows and red dot represent the position of the robot, the orientation and the destination.

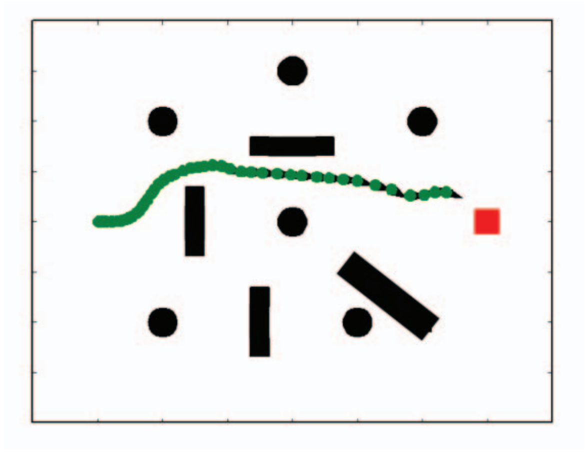
Figure 6: Navigation in simple environment. The green dot, arrows and red dot represent the position of the robot, the orientation and the destination.

4.3 Results

The results are shown in Fig. 5 and Fig. 6. Fig 5 is the navigation result in Env1 and Fig. 6 in Env2. In Fig 5 and Fig 6, the black objects represent obstacles and the red object represents the target. The robot crashed the obstacles frequently at the beginning and couldnt reach to the destination. However, the robot learned environments knowledge through interacting with the environment. Eventually, the robot could navigate to the destination quickly and autonomously in both simple and complex environment without any collision with obstacles. The experiments validate the effectiveness of our model.

5 CONCLUSION

In this paper, a deep reinforcement learning based algorithm is put forward for navigation in unknown environment by using depth images as input only. The experiments in the virtual world demonstrate the effectiveness of autonomous learning from the environment without any supervision signal. The robot can navigate in the environment autonomously, and avoid collisions with obstacles meanwhile. In the future, we will extend the experiment with dynamic environments and transfer to real world.

REFERENCES

[1] Hinton G E , Osindero S , Teh Y W . A Fast Learning Algorithm for Deep Belief Nets[J]. Neural Computation, 2006,18(7):1527-1554.

[2] Fitch F B . McCulloch, Warren S. and, Pitts, Walter. A logical calculus of the ideas immanent in nervous activity.Bulletin of mathematical biophysics, vol. 5 (1943), pp. 115C133.[J]. Journal of Symbolic Logic, 1944, 9(2):36.

[3] Mnih, V., et al., Playing Atari with Deep Reinforcement Learning. 2013.

[4] Krizhevsky A , Sutskever I , Hinton G E . ImageNet Classification with Deep Convolutional Neural Networks[C]//International Conference on Neural Information Processing Systems. Curran Associates Inc. 2012.

[5] Kim, D.K. and T. Chen, Deep Neural Network for Real-Time Autonomous Indoor Navigation. 2015.

[6] Wang J K , Ding X Q , Xia H , et al. A LiDAR based end to end controller for robot navigation using deep neural network[C]// IEEE International Conference on Unmanned Systems. IEEE, 2017.

[7] Kim Y H , Jang J I , Yun S . End-to-end deep learning for autonomous navigation of mobile robot[C]// IEEE International Conference on Consumer Electronics. IEEE, 2018.

[8] Tai, L. and M. Liu, Towards Cognitive Exploration through Deep Reinforcement Learning for Mobile Robots. 2016.

[9] Tai L , Liu M . Mobile robots exploration through cnn-based reinforcement learning[J]. Robotics and Biomimetics, 2016,3(1):24.

[10] van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double Q-learning. arXiv preprint arXiv:1509.06461, 2015.

[11] Schaul T , Quan J , Antonoglou I , et al. Prioritized Experience Replay[J]. Computer Science, 2015.

[12] Xie, L., et al., Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning.

[13] Wang, Z., et al., Dueling Network Architectures for Deep Reinforcement Learning. 2015.Prioritized Experience Replay

[14] Tai, L., G. Paolo and M. Liu, Virtual-to-real Deep Reinforcement Learning: Continuous Control of Mobile Robots for Mapless Navigation. 2017.

[15] S. Gu, E. Holly, T. Lillicrap, and S. Levine, Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates,arXiv preprint arXiv:1610.00633, 2016.

[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa,D. Silver, and D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971, 2015.