Mobile Robot Navigation Using Deep Reinforcement Learning in Unknown Environments

Roan Van Hoa[#1], L. K. Lai[*2], Le Thi Hoan[#3]

[*]*Thai Nguyen University of Technology, Viet Nam,* [#]*University of Economics – Technology Industry, Viet Nam*

Abstract

Mobile robots can cover a large range of real world missions such as environment surveillance, delivery, search and rescue missions. Such missions require different levels of self-navigation in order to react to the dynamic environment changes. However, most of the navigation methods rely on static obstacle map, and don't have the ability of autonomous learning. In this paper, we propose an end-to-end approach 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. Simulation results on the Gazebo framework show the feasibility of the proposed method. The robot can complete navigation tasks safely in an unpredicted dynamic environment and becomes a truly intelligent system with strong self-learning and adaptive abilities.

Keywords — *Autonomous navigation, Deep reinforcement learning, Artificial Intelligence, Mobile Robots.*

I. INTRODUCTION

Autonomous mobile robots are becoming increasingly prevalent in everyday life. Although robots have seen wide spread industrial application since the 1970's, robots are starting to see applications in other less controlled and complex domains such as delivery, the home, medical, military, entertainment, and dangerous environments that humans cannot enter. Extension into these new domains presents a wealth of new challenges.

In recent years, deep reinforcement learning (DRL) [1] has become one of the most concerned directions in the field of artificial intelligence. It combines the perception of deep learning (DL) with the decisionmaking ability of reinforcement learning (RL) and directly controls the behavior of agents through high-dimensional perceptual input learning. It provides a new idea for solving robot navigation problems. Among them, DL, as an important research hotspot in the field of machine learning, has achieved remarkable success in the fields of image analysis, speech recognition, natural language processing, and video classification. The basic idea of DL is to combine low-level features and form abstract, easily distinguishable high-level representations through multilayered network structures and nonlinear transformations to discover distributed feature representations of data [2]. Therefore, the DL method focuses on the perception and expression of things. RL, as another research hotspot in the field of machine learning, has been widely used in industrial manufacturing, simulation, robot control, optimization and scheduling, and game gaming. The basic idea of RL is to learn the optimal policy for accomplishing the goal by maximizing the cumulative reward value that the agent obtains from the environment [3]. Therefore, the RL approach is more focused on learning strategies to solve problems. With the rapid development of human society, in more and more complex real-world task tasks, it is necessary to use DL to automatically learn the abstract representation of large-scale input data and use this characterization as a self incentive RL to optimize problem-solving policy. As a result, Google's artificial intelligence research team DeepMind innovatively combines the sensible DL with the decision-making RL to form a new research hotspot in the field of artificial intelligence, namely deep reinforcement learning. Since then, in many challenging areas, the DeepMind team has constructed and implemented human expert-level agents. These agents build and learn their own knowledge directly from the original input signal, without any manual coding and domain knowledge. At present, DRL technology has been widely used in games [4], robot control [5], machine vision [6], [7], and other fields. In the field of navigation, some interesting and novel articles have appeared one after another. Gupta et al. proposed a neural architecture for navigation in a novel environment [8]. Zhu et al. proposed an actor-critic model and an AI2-THOR framework to improve generalization performance and an environment with high-quality 3-D scenes and physics engines [9]. In terms of an autonomous robot, some papers have been published in [10]-[12] shown that the Q learning network is trained by sampling minibatches of experiences from buffer uniformly at random. However, applying particularly reinforcement learning in robot setting suffers from

many challenges since the high dimensionality and continuous states and actions of the robot [13]. In a simulation, creating an accurate model robot and its environment are challenging and often require a lot of sufficient data samples. To address these dilemmas, the operation of learning of robots is simulated and designed in Gazebo since its compatibility with the complex structure of the robot. More than that, Gazebo enables to construct of a virtual environment [14], which is imperative in the process of scrutinizing reinforcement learning algorithms. Beside, one of major problems in mobile robot navigation is that how the robot can find a collision free path from its starting point. With reinforcement learning algorithms integrated in Gazebo, several methods have been proposed to deal with obstacle avoidances [15]-[17].

The rest of the paper is organized as follows. The second section introduces the proposed model, deep reinforcement learning based navigation. The experimental results will be given in the third section. Finally, the fourth section is the conclusions of this paper.

II. IMPLEMENTION OF DRL NAVIGATION

A. Deep Q-Learning

We adopt the deep Q-learning to train the planner. We formalize this task as a Markov Decision Process (MDP), where the robot interacts with the environment through a sequence of observations, actions and reward signals. For each time step t, the robot perceives a state s_t and needs to select a possible action a_t according to a policy π, where the π is the probability of selecting an action a to be performed for a given state s. Once the action has been executed, a positive or a negative value, which may not be delivered immediately, will be provided as a reward r_t for the robot by the environment together with the next state s_{t+1}. During learning, the robot's aim is to find a policy that collects the highest reward possible over the long run. Given a policy π, the action-value (Q-value) of a state-action pair (s, a), which indicates the expected total discounted reward when executing actions following policy π from state s, is defined as follows:

$$Q^{\pi}(s,a) = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^{t} r_t \mid s_0 = s, a_0 = a, \pi\right] \quad (1)$$

Where the expectation is with respect to the transition distribution under policy π and r_t is the reward for action $a = a_t$ under the policy π in the state $s = s_t$. γ is the discount rate determining future action's influence ($0 < \gamma < 1$). The Q-value function can be computed using the Bellman equation

$$Q^{\pi}(s_t, a_t) = r_t + \gamma \mathrm{E}\left[Q^{\pi}(s_{t+1}, a_{t+1}, \pi)\right] \quad (2)$$

The optimal $\pi*$ corresponds to taking the best action in any state s and the optimal Q-value function Q^* can be obtained as follows:

$$Q^{*}(s,a) = r_t + \gamma \max_{a'} \mathrm{E}\left[Q^{*}(s_{t+1}, a')\right] \quad (3)$$

Where a' represents the possible actions in the future state s_{t+1}.

The basic idea behind many RL algorithms is to estimate the Q-value by iteratively updating based on the Bellman equation. Traditional methods usually calculate the Q-value function directly over a large state, which has low efficiency and lacks generality. Recent successes of RL in many applications rely on the technique of combing deep neural network and RL, where neural networks are used to estimate the Q-value function. This is the main idea behind DQN [18]. For a neural network that works as a function approximator for the Q-function, its parameters are updated as follows:

$$\theta_{i+1} = \theta_i + \alpha\left(r + \gamma \max_{a'} Q(s', a', \theta_i) - Q(s, a, \theta_i)\right)\nabla_{\theta_i} Q(s, a, \theta_i) \quad (4)$$

Where θ_i are parameters of the network at iteration i and α is the learning rate.

B. Double Q-Learning

As shown in Equation (3), the objective of the Q-learning is to bring the current value of $Q(s, a)$ to the target value of $\Upsilon_t^{Q} = r + \gamma \max_{a'} Q(s', a')$. During the learning process, the Q-function $Q(s, a; \theta)$ that evaluates the future approximated action values is also used to select the action. This can sometimes overestimate the action values, resulting in overoptimistic value estimations and slow learning speed. To solve this problem, Van Hasselt et al, [19] proposed the Double DQN (D-DQN) that uses two sets of weights θ and θ^{-}, where the online network ($Q(s, a; \theta)$) is used to select the action and the target network ($Q(s, a; \theta^{-})$) is used to evaluate the action values. The implementation only requires a minor change to the DQN algorithm. Recall that the target in the DQN is calculated as:

$$\Upsilon_t^{DQN} = r + \gamma \max_{a'} Q\left(s', \arg\max_{a'} Q(s', a', \theta), \theta\right) \quad (5)$$

The target in D-DQN can be written as follows:

$$\Upsilon_t^{DDQN} = r + \gamma \max_{a'} Q\left(s', \arg\max_{a'} Q(s', a', \theta), \theta^{-}\right) \quad (6)$$

Where θ is a set of parameters for the online network and θ^{-} is another set of parameters for the target network. During learning, θ are updated at every training step while θ^{-} are fixed over a short period and then copied from the weights θ. D-DQN has been found to learn better policies than DQN on Atari games [19].

C. Dueling Q-Learning

The Q-value $Q(s, a)$ corresponds to how good it is to take a certain action given a certain state, which implicitly contains two elements: the value of

being at the state and the advantage of taking the action at that state. For a state with multiple action choices, DQN usually aims to estimate the Q-value of each state-action pair. However, sometimes it is unnecessary to calculate the value of each action, considering that for states where their actions do not affect the environment in any relevant way. With the aim to explicitly separate the state value and the action advantage, Wang et al, [20] proposed the dueling network architecture. In this architecture, two streams of networks are used to separately estimate the state value function *V(s)* and the associated advantage function *A(s, a)*, which are then combined together to estimate the action-value function *Q(s; a)*. The Q-value can be constructed as the sum of *V*(*s*) and *A*(*s*, *a*)

$$Q(s, a) = A(s, a) + V(s) \quad (7)$$

It has been demonstrated that, compared with DQN and D-DQN, the dueling DQN can lead to faster learning speed and better performance in a number of tasks [20].

D. Implementation of D3QN

RL requires huge amounts of data and time for obtaining appropriate behaviors. For the avoidance behavior learning, actions that collide with obstacles need to be iterated, which is not possible for a robot in the real world. A feasible solution is to implement the training in a simulator and then transfer the learning results to the real robot. However, this is a challenging task considering the huge difference between the structural simulation environment and the highly complicated real-world environment, especially for vision-based learning. In this work, the planner is trained based on the laser scan data. Compared with visual images, laser scans are relatively low-dimensional and the difference between the simulation and the real world is smaller. It is possible to enable an easier transfer from simulation to reality.

In this paper, we adopt the D3QN model [20] that combines the double and dueling techniques to train the planner to perform obstacle avoidance. The architecture is shown in Figure 1. and corresponding implementation details of each layer are summarized in Table 1. The input is a 36-dimensional vector consisting of 36 laser range findings which are sampled from the raw laser range findings between -180^0 and 180^0 in a fixed angle distribution of 5 degrees. After the input layer, a fully connected layer of 100 nodes is shared by the value network and the advantage network which both consist of 2 fully connected layers and calculate the value and advantage, respectively. The value network has 1 output and the advantage network has 5 outputs referring to the number of valid actions. The outputs of these two networks are finally combined to compute the Q-values of the state-action pairs.

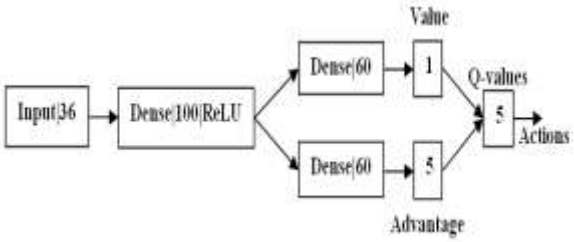

Figure 1: Deep double Q network (D3QN) network structure

TABLE 1: Implementation details of the deep double Q network (D3QN) for obstacle avoidance

Layer Name	Number of Neurons	Activation Type
Input	36	-
Shared FC	100	ReLU
FC1 for value	60	ReLU
FC1 for advantage	60	ReLU
FC2 for value	1	Linear
FC2 for advantage	5	Linear
Output	5	-

III. EXPERIMENTAL RESULTS

A. Training in Simulation

The training procedure of the planner was implemented in a virtual environment simulated by Gazebo [14]. Figure 2 and 3 shows an overview of the simulation environment which contains a number of obstacles of different shapes and sizes; White is a moving obstacle in each group, black is a TurtleBot simulation mobile robot, blue is the laser range of the mobile robot, and red square is the target point of a training.

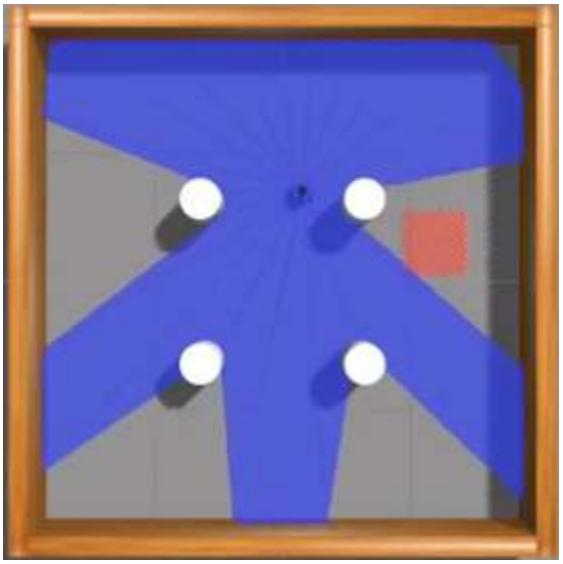

Figure 2: Navigation in simple environment

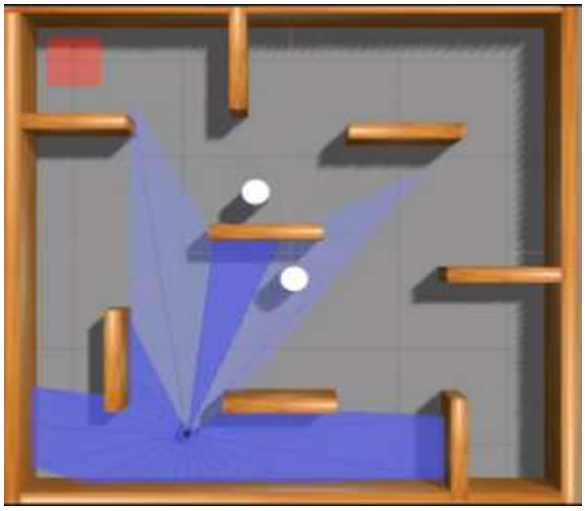

Figure 3: Navigation in complex environment

We adopted the D3QN model to train the planner and related training settings of this learning task are shown as follows:

a) Action space

In this navigation task, the robot is instructed to move forward with a constant step length (0.3 m) and the actions are defined to control the robot's angular velocity in a discretized format. It includes 5 actions: Turning left by 60^0, turning left by 30^0, moving forward, turning right by 30^0, turning right by 60^0.

b) Observations and Target

The state of the robot is represented by 36 sampled laser range findings from the raw laser scan. During navigation, the robot's objective is to learn the action policy that enables the robot to bypass the objects placed in the environment. Since the robot moves with a constant linear velocity, the learning task basically requires the robot to change its angular velocity based on the relative spatial positions between itself and the objects.

c) Reward Function

The objective of the agent is reflected in the design of the reward function. For the problem of autonomous mobile robot navigation, the reward function should reward the agent for moving the robot toward and reaching the target location, and penalize it for moving away from the target and colliding with obstacles. To achieve this, the following reward function was designed:

$$r = \begin{cases} 1 & d_{r-t} < 0.5 \\ -1 & collision \\ v\cos(\theta) & otherwise \end{cases} \quad (8)$$

If the distance d_{r-t} between the robot and the target is less than 0.5 meters, the agent receives a reward of 1. A collision is considered if robot detects an object closer than 5 cm, and the agent receives a reward of -1. At all other time steps, the agent receives a reward of $v\cos(\theta)$, where v is the linear velocity of the robot, θ is the heading to the target. The reward function was modified to not only encourage the robot to not crash, but also move in the direction θ toward the target location.

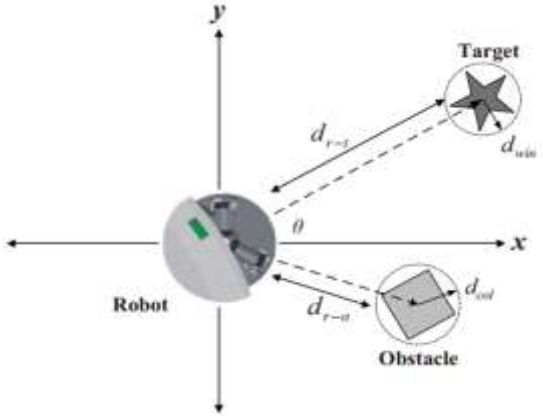

Figure 4: The mobile robot environment with a target and an obstacle

During training, a series of processes from action selection to learning were iterated. In every episode, the robot was initialized at a random position with a random orientation in the simulator, which was guaranteed to be collision-free with objects. The robot navigated through the simulator episodes by episodes, during which the parameters of the neural network were updated based on the interactions with the simulator. To train the network, we used the Adam optimizer [21] with a learning rate of 0.0001. The action selection policy was based on ε-greedy with ε annealed linearly from 1 to 0.1 over the duration of the training. An experience memory of size 50000 was built to store experiences and mini-batches of 8 were used to randomly retrieve experiences from the experience memory for learning and updating the neural network parameters. During experiments, we found adding noise to the training data could make the trained models transferable better from simulation to reality. For this aim, the laser scans for training in the simulator were corrupted with noise randomly sampled from a normal distribution with parameters mean = 0, std = 0.1.

B. Simulation results

Figure 5 presents the learning result in the Gazebo simulator. As we can see, the average reward of the robot in each episode keeps increasing as the training continues. 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.

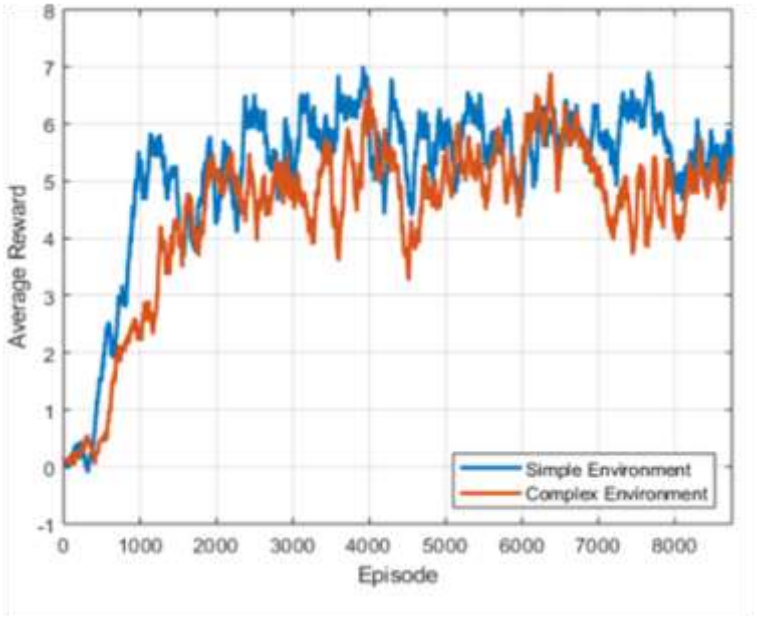

Figure 5: Training average reward results

IV. CONCLUSIONS

In this paper, we presents a reinforcement-learning based methodology for the autonomous navigation in dynamic and unknown environments by using RPLidar as the input data. The simulation results in the Gazebo environment demonstrate the ability of mobile robots to navigate to desired target locations in simple and complex environments. The practical model for the mobile robot will be constructed and the reinforcement learning network will be implemented in the robot's navigation task in the real environments for the future work.

ACKNOWLEDGMENT

This study was supported by Thai Nguyen University of Technology; http://www.tnut.edu.vn/.

REFERENCES

[1] Li Y, "*Deep reinforcement learning*", In: ICASSP 2018—2018 IEEE international conference on acoustics, speech and signal processing (ICASSP), Calgary, AB, Canada, 15–20, April 2018.

[2] Sun ZJ, Xue L, Xu YM, et al, "*Overview of deep learning*", Appl Res Comput 2012, 12, pp. 2806–2810.

[3] Sutton RS and Barto AG, "*Reinforcement learning: an introduction*", IEEE Transactions on Neural Networks, 2005.

[4] Hosu I-A and Rebedea T, "*Playing Atari games with deep reinforcement learning and human checkpoint replay*", 2016. ArXiv, abs/1607.05077.

[5] Lillicrap TP, Hunt JJ, Pritzel A, et al, "*Continuous control with deep reinforcement learning*", Comput Sci 2015, 8(6): A187.

[6] Caicedo JC and Lazebnik S, "*Active object localization with deep reinforcement learning*", In: Proceedings of the IEEE international conference on computer vision, Santiago, Chile, 2015, pp. 2488–2496.

[7] Meganathan RR, Kasi AA, and Jagannath S, "*Computer vision based novel steering angle calculation for autonomous vehicles*", In: IEEE international conference on robotic computing, Laguna Hills, CA, USA, 31 January–2 February, 2018.

[8] Gupta S, Tolani V, Davidson J, et al, "*Cognitive mapping and planning for visual navigation*", In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 7272–7281.

[9] Zhu Y, Mottaghi R, Kolve E, et al, "*Target-driven visual navigation in indoor scenes using deep reinforcement learning*", In: 2017 IEEE international conference on robotics and automation (ICRA), Stockholm, 16–21 March 2016, pp. 3357–3364.

[10] S. Amarjyoti, "*Deep reinforcement learning for robotic manipulation-the state of the art*", *Bull. Transilv. Univ. Braşov*, vol. 10, no. 2, 2017.

[11] A. V. Bernstein, E. Burnaev, and O. Kachan, "*Reinforcement learning for computer vision and robot navigation*", in Proc. International Conference on Machine Learning and Data Mining in Pattern Recognition, 2018, pp. 258-272: Springer.

[12] V. Matt and N. Aran, "*Deep reinforcement learning approach to autonomous driving*", ed: arXiv, 2017.

[13] X. Da and J. Grizzle, "*Combining trajectory optimization, supervised machine learning, and model structure for mitigating the curse of dimensionality in the control of bipedal robots*", Int. J. Rob. Res., vol. 38, no. 9, pp. 1063–1097, 2019.

[14] I. Zamora, N. G. Lopez, V. M. Vilches, and A. H. Cordero, "*Extending the openai gym for robotics: A toolkit for reinforcement learning using ros and gazebo*", arXiv preprint arXiv:1608.05742, 2016.

[15] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "*Socially compliant mobile robot navigation via inverse reinforcement learning*", The International Journal of Robotics Research, vol. 35, no. 11, pp. 1289-1307, 2016.

[16] L. Tai and M. Liu, "*A robot exploration strategy based on qlearning network*", in Proc. 2016 IEEE International Conference on Real-time Computing and Robotics (RCAR), 2016, pp. 57-62.

[17] L. Tai, G. Paolo, and M. Liu, "*Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation*", in Proc. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 31-36.

[18] Mnih. V, Kavukcuoglu. K, Silver. D, Rusu. A.A, Veness. J, Bellemare. M.G, Graves. A, Riedmiller. M,

Fidjeland. A.K, Ostrovski. G, et al, "*Human-level control through deep reinforcement learning*", Nature 2015, pp. 518-529.

[19] Van Hasselt. H, Guez. A, Silver. D, "*Deep Reinforcement Learning with Double Q-Learning*", AAAI: Phoenix, AZ, USA, 2016; Volume 2, p. 5.

[20] Wang. Z, Schaul. T, Hessel. M, Van Hasselt. H, Lanctot. M, De Freitas. N, "*Dueling network architectures for deep reinforcement learning*" arXiv 2015 arXiv:1511.06581. Avaliable online: https://arxiv.org/pdf/1511. 06581.pdf (accessed on 12 September 2018).

[21] Diederik P, Kingma and Jimmy Ba, "*Adam: A method for stochastic optimization*", *CoRR*, abs/1412.6980, 2015.

[22] Dr.V.V.Narendra Kumar, T.Satish Kumar, "Smarter Artificial Intelligence with Deep Learning" SSRG International Journal of Computer Science and Engineering Vol-5,Iss-6,2018.