Short-range Robotic Navigation and Exploration Tasks via Deep Q-Networks for Biomedical Applications

Joel Dzidzorvi Kwame Disu[1], Clinton Elian Gandana[2], Hongzhi Xie[3], Lixu Gu[4]
[1,2,4]*Institute of Medical Robotics, School of Biomedical Engineering, Shanghai Jiao Tong University*, Shanghai, China
[3]*Department of Cardiology, Peking Union Medical College Hospital*, Peking, China
[1]joel-disu90@sjtu.edu.cn, [2]clintonelian@sjtu.edu.cn, [3]drxiehz@163.com, [4]gulixu@sjtu.edu.cn

***Abstract*—This research is focused on the performance of a Deep Reinforcement Learning method on an agent (mobile robot) in a simulated virtual environment (Operating Room) for medical applications. The purpose of this research is to compare suitable decisive actions taken by the agent to achieve its goal target. Executing this goal requires the implementation of a reward-penalty system for observation and analysis. The agent's accumulated reward is based on the best-navigated decision to avoid collisions; solely generating an intelligent agent system. We reviewed previous works on the impact of Deep Reinforcement Learning algorithms on an agent in areas of navigation and exploration. Adopting a Deep Reinforcement Learning method and a physical simulator, we trained and tested the agent using existing environments and our modeled operating room, respectively. Measuring the positive reward output of the experiment with different parameters of the algorithm such as the learning rate, maximum Q-value and the average time to attain its goal position, we presented our work with plots of the experiment and compared it with a widely known traditional method. Our experimental results indicated that the agent achieved a high positive reward of 3800 in our operating room environment with a learning rate of 0.5. Our research aimed at training an agent to make intelligent decisions in achieving its goal destination without prior experience and input data. Reinforcement Learning provides a structure for robotics to function effectively; utilizing and engaging a robot to navigate and explore in any given environment.**

***Index Terms*—Reinforcement Learning, Deep Q-Networks, Navigation, Exploration, SLAM, Operating room, Biomedical application.**

I. Introduction

The use of artificial intelligence has become relatively common for many biomedical applications, including patient monitoring and evaluation, medical supplies delivery, and assisting healthcare professionals. The assistance of medical robots grants a better-quality care, exactness of its workflow, a robust and resilient healthcare system; alleviating trained operators from tough and tedious tasks [1]. Machine learning is an essential component of artificial intelligence (AI) and spans through all fields of AI. Although the concept of machine learning (reinforcement learning) and robotics has effectively employed non-mobile medical robots in the medical field for healthcare and surgical procedures, navigation and exploration continue to remain as two of the biggest challenges in medical robotics [2]. For example, the frustration of patient overpopulation when doctors and nurses are worn-out. This factor is as a result of healthcare professionals working around the clock to check vitals for the outbreak of a severe novel respiratory disease [3].

In Reinforcement Learning, an agent (robot) chooses an action based on its current decision, receives feedback and the next state from its given environment. The agent makes better action selections in future states through repeated learning, as it continues to explore and exploit the environment. There are three main components of an RL system; the state, action, and reward. The agent receives a state in a state space, decides to select an action from an action space, gets a scalar reward, and moves to the next state to complete the cycle according to the environment dynamics [4].

The mobile agent aims to discover an optimal policy that provides the highest positive reward over time. Previous dynamic programming (DP) methods such as Simultaneous Localization Autonomous Mapping (SLAM) employs an algorithm for path planning and navigation. A mobile robot can simultaneously formulate a map of any given environment and derive it's the location through SLAM [5]. However, this method is time-consuming as the algorithm needs to first map its environment and create a path to follow through DP planning [6]. The use of RL methods with deep neural networks operates the mapping and pathfinding as end-to-end so that the progress works more structurally and efficiently.

In this work, the discussion of the implementation of Deep Q-learning (DQN) on an agent (robot) is our primary task, and we conducted experiments on a modeled simulated virtual environment through the communication between the Robot Operating System and a virtual simulator. The mobile robot engages its environment based on a reward and penalty system to avoid repetitive exploration and ensure time efficiency. Employing this method in healthcare reduces the burden of medical professionals and facilitates smooth workflow in the hospital. Fig. 1 describes our workflow, and its criteria has been used as a guideline in this paper. Section II reviews the previews works on this topic, and Section III explains the Reinforcement Learning method. Section IV explains the agent and the operating room environment used for this paper, and Section V provides details about its methodology. Section VI describes the results and Section VII is the conclusion.

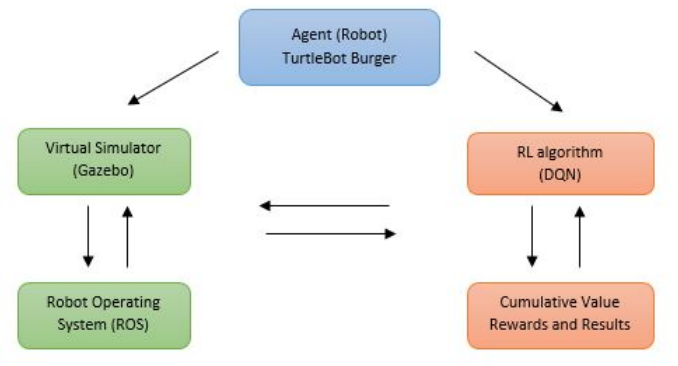


Fig. 1. Simple Architecture Diagram.

Our principal focus of research is the utilization of a Deep Reinforcement Learning method to minimize the workload of medical operators in the hospital by providing them with mobile robots that are equipped with a high-quality navigation and exploration system to perform both simple and complex tasks.

II. Previous Works

The application of Reinforcement Learning for navigation and exploration is evident in different fields. For example, Faust et al. [7] involved the integration of a path planning algorithm with Deep Deterministic Policy Gradient for individual navigation tasks. This hierarchical method tested its robustness on two navigation tasks: a complex indoor navigational system using a differential robot and aerial cargo delivery in central cities. Their approach successfully improved task completion by the RL agent used for the experiment.

Furthermore, Deep Reinforcement Learning in [8] trained a nonholonomic mobile robot to navigate in both invisible virtual and real environments with the help of a mapless motion planner. The robot reached its goal destination without colliding with obstacles. The operation of the two model-free algorithms (Q and DQN) in [9] provided the best actions for the robot to be self-balanced in a virtual environment. The environment was built in Gazebo, a virtual simulator, and combined with the Robot Operating System (ROS) to perform experiments efficiently at low cost.

Classical methods such as SLAM facilitate a mobile robot to build consistent maps when placed at an unknown location in an unfamiliar environment, thereby creating an autonomous system and providing computational efficiency [10]. Wang et al. [11] proposed a low-cost, independent mobile robot system for localization, navigation, and detection and avoidance of obstacles in an environment. With minimized complexity, the robot performed various tasks in indoor environments. However, SLAM strategies assume proper data association, which leads to uncertainties. The use of different filters initially improved this problem, but they are computationally expensive when dealing with high dimensional maps [12].

To solve this uncertainty, we proposed a Reinforcement Learning approach applied to our modeled operating room using deep neural networks to process data within any level of complexity.

III. Theory and Algorithms

Deep Reinforcement Learning combines the concept of deep learning and reinforcement learning. This method allows the simplification of an ample sample space in complex environments through deep neural networks, which is essential for a mobile robot tasked with navigation and exploration. Therefore this method is a potential solution [8]. The model-free algorithm expresses the idea that the agent (robot) learns by trial-and-error from experience explicitly, exploits, and employs the current best action to maximize rewards.

A. Deep Q Networks

A typical Reinforcement Learning setup consists of an action, state, reward, policy, and action value. The key to solving RL problems are either estimating the expected return when starting in a state or searching for an optimal strategy [13]. Deep Q Network is a branch of Q-learning. The concept of the Q-learning algorithm is an off-policy system based on the Bellman Equation, and the goal is to maximize the Q-value, which is the cumulative reward, as shown in Equation 1.

$$Q\left(s,a\right) = r\left(s,a\right) + \gamma \cdot maxQ\left(s^{'},a\right) \tag{1}$$

Gamma γ here ranges from the values of 0 to 1 and represents the discount factor that maximizes the priority of future rewards. The equation above expresses the Q-value resulting from its initial state s_t and performing action a is the current reward $r\left(s,a\right)$ including the likelihood of the highest Q-value from the next state s_{t+1}. This recursive equation allows hypothesis for all Q-values with prior knowledge and convergence to the optimal policy. Equation 2 represents this principal factor.

$$Q\left(S_t,A_t\right) \leftarrow Q\left(S_t,A_t\right) + \alpha\left[R_{t+1} + \gamma maxQ\left(S_{t+1},a\right) - Q\left(S_t,A_t\right)\right] \tag{2}$$

The learning rate is the step size denoted by alpha α. This hyper-parameter simply determines the update transfer from old to new information for intelligent decision-making. Although Q-learning is an efficient algorithm, it lacks generality and is unable to estimate the value of the unseen state. Also, the amount of memory required to save and update the Q-table increases with high state values; thus, a considerable amount of time needed to explore each state. Introducing neural networks resolves this issue.

Deep Q Network is a model-free algorithm responsible for approximating the Q-value function. The input for the neural architecture is the current state, while the output is the corresponding Q-value for each of the actions. Training samples in Reinforcement Learning are usually highly correlated and less-data efficient, leading to non-convergence for the network.

DQN uses two key ideas; experience replay and an iterative update. According to [14], the principle of experience replay stores the past experiences S_t, A_t, R_t, S_{t+1} of an agent at each time step. This feature merges many episodes into a replay memory and randomly selects from the "transition pool" to update the knowledge (Q-table). This pool is the iterative update. The Loss function represents the Q-learning update at iteration in Equation 3.

$$L_i\left(\theta_i\right) = \mathbb{E}_{\left(s,a,r,s'\right)} \sim U(D) \left[\left(r + \gamma maxQ\left(s^{'}, a^{'}; \theta_i^{-}\right) - Q\left(s, a; \theta_i\right)\right)^2\right] \quad (3)$$

L is the loss function, and theta i represents the specifications of the network structure at iteration i while theta $-i$ refer to the target computations at iteration i. This model of a convolutional neural network trained with Q-learning existed in seven Atari 2600 games, and the results from the model outperformed a human expert on three occasions [15]. In DQN, optimization and convergence are two essential characteristics, and therefore we employed a gradient descent step [16].

IV. Experimental Setup

A. Agent (TurtleBot 3 Burger)

As shown in Fig. 2 , the robot model TurtleBot 3 Burger is the agent for this project in a simulated operating room environment in Gazebo. TurtleBot is a Robot Operating System standard platform robot used for education, high-quality research, and product prototyping. Due to the high cost of real-world robotic simulation, the TurtleBot 3 family provides a reduction in the size of the platform with similar functionality and quality. It consists of three versions; TurtleBot 3 burger, waffle, and waffle pi:- all built with a robust embedded system that is equipped with a 360-degree distance sensor (LIDAR) used for reflecting nearby obstacles. The Raspberry Pi, which is below the LIDAR senses and reads data, and further transfers information to a 'master pc' for real calculations [17].

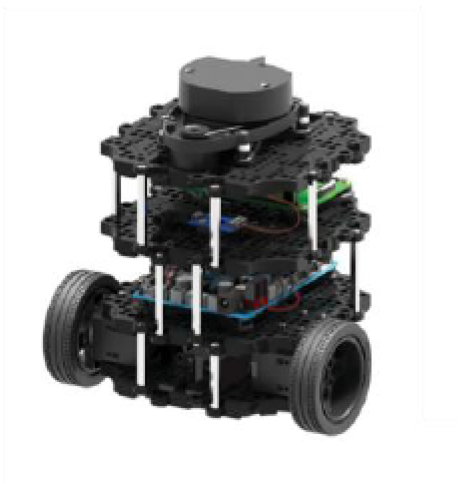


Fig. 2. Virtual TurtleBot 3 Burger.

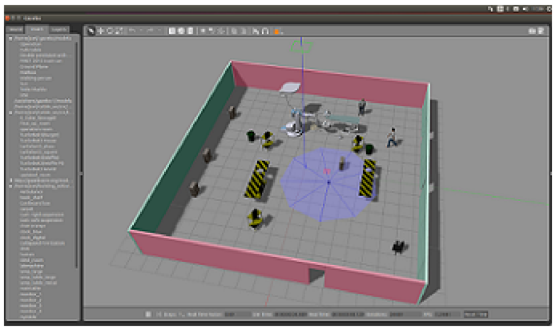


Fig. 3. Structure of Operating Room in Gazebo.

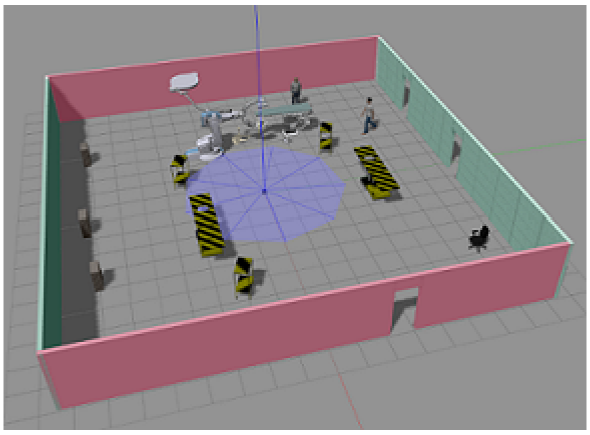


Fig. 4. Our Operating Room model with Turtlebot 3 burger.

B. Environment (Operating Room)

Using the Gazebo tool, we modeled our customized environment and launched the TurtleBot 3 burger. Our environment depicts that of a real-world operating room. Fig. 3 and Fig. 4 provide details of our operating room. It follows the main principles of working room management and efficiency. These factors include ensuring patient safety and optimal patient outcome; providing surgeons with appropriate access to the operating room; maximizing the effectiveness of the operating room utilization; staff and materials to reduce costs; decreasing patient delays; and enhancing satisfaction among patients; staff, and medical professionals. Medical equipment and human models for the environment were sourced and later modified from 3D Warehouse (a large dataset of simulated 3D objects) and [18].

C. ROS and Gazebo

Robot Operating System (ROS) uses tools and libraries necessary for writing software for robots and used in robotics research and industry. The framework operates through a publisher-subscriber interface and provides functionality like hardware abstraction, localization, navigation, and visualization. ROS works with Gazebo (our virtual simulator) through plugins that provide the correct interface for messages, services, and dynamic reconfigure.

We conducted experiments in the modeled operating room virtual environment with the various obstacles such as medical equipment and humans. The mobile agent motioned into this virtual environment to find the right navigation towards its goal box. Codes were written and modified in Python Language with Kinetic Kame as our Robot Operating System and Gazebo 7.0 as our virtual simulator with Linux Ubuntu 16.04 Operating System.

V. Methodology

Using the agent, ROS, and Gazebo, we first trained and further tested the agent with our modeled environment. This technique provided the mobile robot with some learning and explorative experience. Our methodology is as follows;

A. Training the Agent and Setting Hyper-parameters

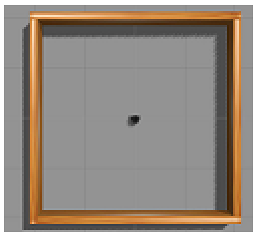
We conducted experiments for the DQN algorithm with different parameters but primarily the state, action, reward, step size, and the discount factor. We trained the agent using the source code package DQN from the ROBOTIS company and its simulated environments. We trained the agent with pre-modeled simulated environments for four different environment scenarios. The first environment had no obstacles, the second had four cylinders of static obstacles, the third had four cylinders of moving barriers, and finally, the fourth had a combination of collision walls and two moving obstacles. The environments for training in Fig. 5. We also contributed some code alterations to cater for the reward and penalty value system in the Gazebo simulation.

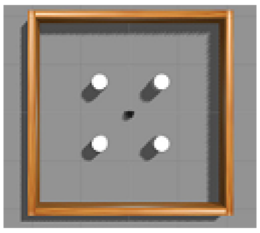
We projected our results with a Q-value and cumulative reward graph. The total reward graph represents the overall reward value for each state when the mobile robot explores and navigates in any of the four simulated environments. When the reward value is below 0, it means the agent has collided with a wall or medical object and therefore produces a negative reward (penalty). A positive reward indicates successful navigation towards its goal target within a short period. The Q-value graph presents the best value action for each episode of navigation and exploration in the operating room.

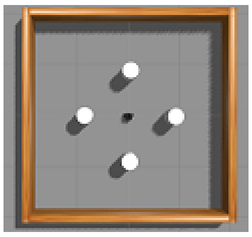
We performed training for 3000 episodes with a learning rate of 0.00025, an epsilon factor of 1.0, and a discount factor of 0.99. After training, we transferred the learned agent for testing. The Adam optimizer for the experiment accommodated sparser gradients and convergence with the mean square error, which also represents the loss function. This optimization tool also serves as a gradient descent approach by working with a single sample each time, and usually mini-batches.

B. Testing the Model in the Virtual Operating Room

We tested the training model in the virtual operating room environment. We used a high and low learning rate and discount factor values to measure the effectiveness of the agent's navigation towards the goal direction. The learning rate is the extent to which new data dominates recent data. A factor of 0 provides the agent with minimal exploration


(a) No obstacles


(b) Static obstacles


(c) Moving obstacles

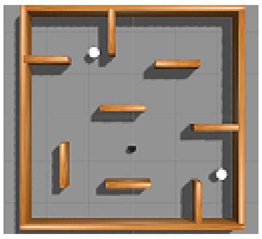
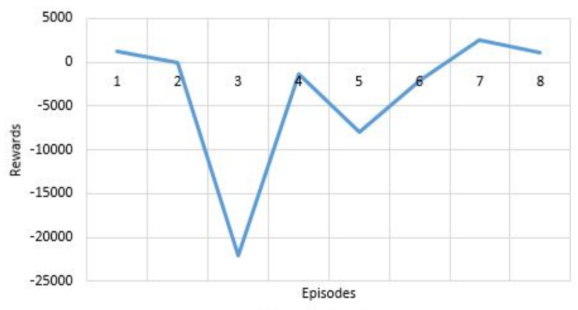
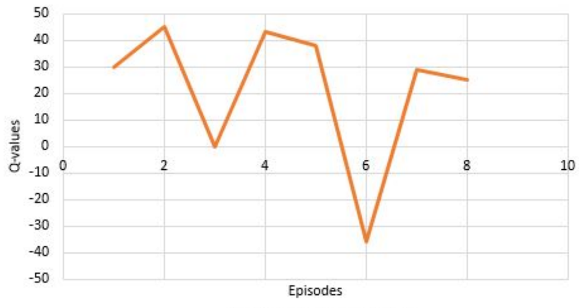

(d) Combination obstacle

Fig. 5. Simulated environments (a) - (d) for training Turtlebot 3 burger.



(a) Reward Value



(b) Average max Q-value

Fig. 6. Positive Reward value graph after 7 episodes.

techniques in the simulated environment, whiles a factor of 1 engages the agent to examine the updated information, which might not include prior-knowledge.

Adopting the two learning rates for the agent explores all possible actions for each state and finds the one that is the most rewarded for exploiting it in achieving its goal. The discount factor maximizes the priority of future rewards. An element value of 0 will consider only updated positive rewards, while a factor approaching the value of 1 will employ the agent in achieving a constant high positive reward [19]. These two parameters are necessary for a successful future workflow in the operating room. Using a Keras sequential model, we structured our neural network with two dense layers and a linear activation layer.

To carefully monitor the agent's progress, our graph displayed the reward and penalty value with the value of 1000 and -500 being the chosen highest reward and penalty respectively for each episode. The penalty value depicts the agent colliding against a human, medical equipment, or a wall, while a reward value expresses the agent achieving the success of reaching its targeted goal.

VI. Experimental Results and Discussion

The chart in Fig. 7 explains the performance of the DQN algorithm in our operating room environment. Fig. 6 accounts for our initial results. After seven episodes of testing, the agent achieved a positive reward in our environment when the learning rate was 0.00025 with DQN. Although the agent took more steps and a longer time towards its goal, it accumulated a positive reward of 1000 after 500 episodes with a learning rate of 0.5. This is attributed to the fact that the agent starts to learn a new environment with a higher number of steps at its initial stages and reduces as it achieves a positive reward. We noticed negative reward values at early episodes of testing, which accounts for exploration.

As presented in Fig. 8 and Fig. 9, the results exhibit that as learning rate increases in value, the reward increases and with increasing average Max-Q value. This result accounted for the success in training the agent at a lower learning rate to enable it to explore and exploit all actions in each state in the four simulated environments. Moving into a new environment (operating room) gives a lower chance of collisions and therefore maximizes its goal target in most given states and episodes based on its past learning experiences. A higher reward indicates a better-navigated decision, and consequently, the experiments show the higher reward at a learning rate of 0.5. Table I provides further details. The maximum Q-value is the level of intelligence of our neural architecture and the increase in the average Max-Q value is the cumulative reward for every increase in the episode. The instability of the cumulative reward values is the effect of the trade-off between exploration and exploitation.

Exploration occurs when the agent moves autonomously to gain experience by moving in different directions and taking actions that lead to its targeted goal. Exploitation exists when the robot initially attains a penalty value from its last episode and, based on its prior experience, would only aim at achieving a goal instead of navigating and learning the environment. The concept of Deep Reinforcement Learning depends on weight update transfer. When the robot makes a decision based on its sensors, the result of the information quickly transfers and updates the neural structure. The input layer of the framework contains weight decisions, and the output layer is the reward-penalty value system, which updates according to the current decision-making process.

Table II explains the comparison of the average time of DQN with a globally known traditional method. Comparing results with [20], the Deep Reinforcement learning method approaches its targeted goal faster than SLAM because it presents a faster learning update with deep neural networks. The intelligent structure modeling of our DQN algorithm facilitates the robotic structure to choose wise options for steering in our virtual world. This robust learning update engages the robot in making intelligent decisions without any data input as compared with traditional methods. Another reason is that DQN uses deep neural networks to account for data associated uncertainties, and therefore the time taken to build and learn maps is shorter than that in classical methods.

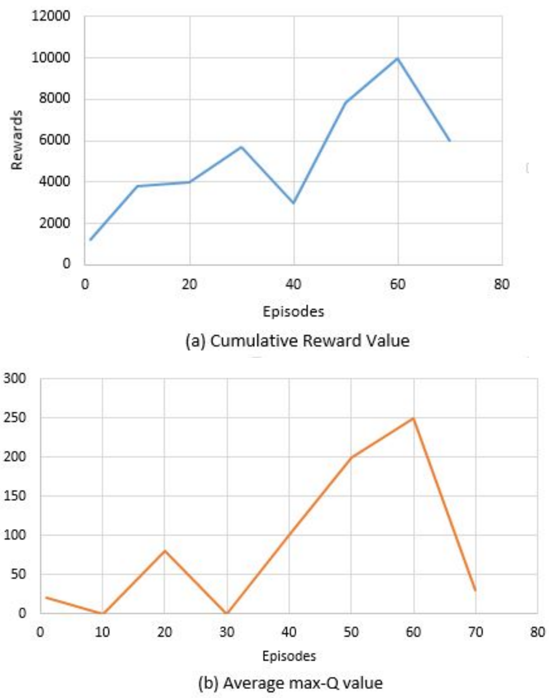


Fig. 7. Testing Results when learning rate is 0.00025.

TABLE I
Comparison of different learning rates for DQN algorithm

Algorithm	Parameters		
	Learning rate	*Discount factor*	*Highest Cumulative reward*
DQN	0.01	0.99	3000
	0.5	0.99	3800

TABLE II
Average time comparison for agent to reach the goal position

Algorithm	Average time (s)
DQN	75.3
SLAM	156.5

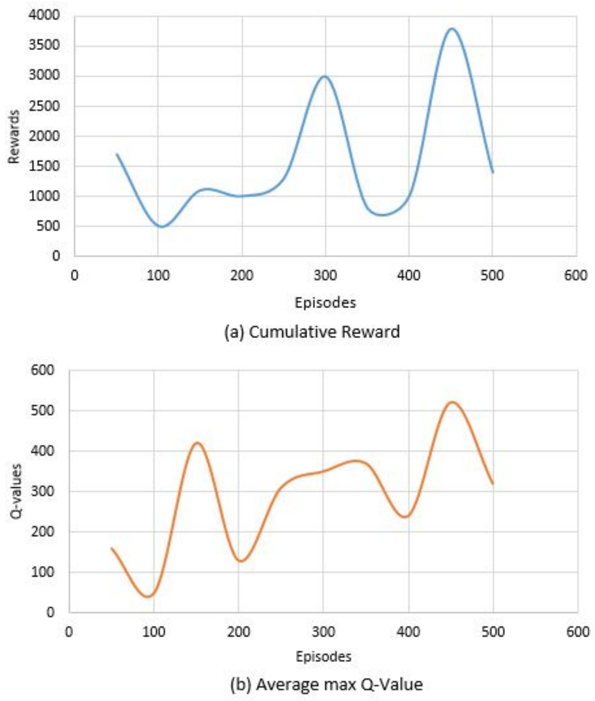


Fig. 8. Rewards and Q-value graph with learning rate as 0.5.

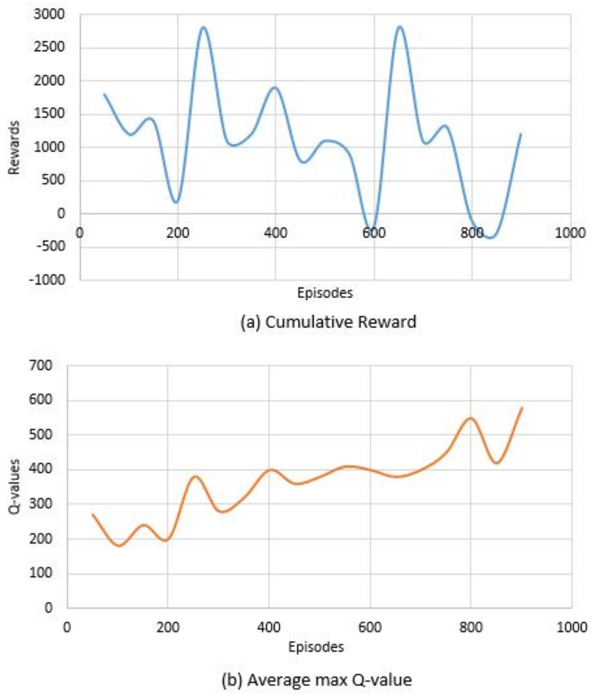


Fig. 9. Rewards and Q-value graph with 0.01 learning rate.

VII. Conclusion and Future Work

In this paper, we successfully deployed the DQN algorithm in an operating room environment. We have discussed the algorithm with the task of finding the best-navigated action performed by the agent towards a positive reward in a simulated operating room. One challenge of Reinforcement Learning is the trial-and-error interaction with the environment, but the system of Deep Reinforcement Learning uses its optimal policy to resolve this issue. The long-range time dependency is another challenge of RL. Our algorithm deals with this problem by introducing a time-out plan that provides a lesser penalty value after excessive training for one episode. The work of Deep Reinforcement Learning achieves input information through a deep neural architecture, and its middle and output layers provide decision-making actions for the agent to deploy and consequently gain positive rewards. The agent obtains a penalty when it does not achieve its targeted goal. The large memory of the architecture continually updates to include current poses and experiences while the layers calculate the weight decisions to ensure a maximum cumulative reward. The application of this concept with our modeled simulated virtual operating room proves its advantages over classical methods.

In our research, we used model-free algorithms to facilitate efficient learning by the agent. We used the TurtleBot burger 3 for our research and analysis. This multifunctional agent has a package system, used by a wide variety of companies for education and research. We chose this agent because it is user friendly and affordable. The system of the agent fits the model of the software (Gazebo and ROS) that provides a connection between them to carry out relevant research. The agent initially trained in four different environments. In the first scenario, although it stumbled on most collision elements, it achieved a high reward after several episodes. The second and third environments almost achieved similar results as the complexity of the environment increased, and in this case, our neural architecture updated decision weights continually to create a policy for our agent. The fourth environment mimicked our operating room, and we trained the TurtleBot 3 burger twice for 3000 episodes each to provide the agent with enough training experience for validation and testing.

During experimental work, we compared the optimal parameters, time rate, and efficiency with traditional methods. The fastest convergence for our algorithm was with a learning rate of 0.5 and a discount factor of 0.99. Workload minimized by the agent will increase workflow in the operating room and facilitate efficiency. A promising way for this work would be to run more sophisticated experiments with other different learning rates on the agent in the operating room environment. Future work should also be on the performance of other different RL algorithms on the environment. The application of this work extends to solving pandemics in both developed and developing countries to combat epidemics that prevent the movement of locomotion. An example is the outbreak of an infectious disease (COVID-19).

There are abundant opportunities in intelligent navigation areas that require the field of robotics to undertake tasks [21]. We envision a complex navigational system for outdoor medical roles, especially for measuring vital signs and delivering medications to patients as part of our improvement for real-world simulation.

Acknowledgment

This research is partially supported by the National Key research and development program (2016YFC0106200), Beijing Municipal Natural Science Foundation (L192006) and the funding from IMR of SJTU as well as the 863 national research fund (2015AA043203).

References

[1] E. D. Momi, L. Kranendonk, M. Valenti, N. Enayati and G. Ferrigno, "A Neural Network-Based Approach for Trajectory Planning in Robot-Human Handover", *Frontiers in Robotics and AI*, 2016.

[2] G-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield, B. J. Nelson, B. Scassellati, M. Taddeo, R. Taylor, M. Veloso, Z. L. Wang and R. Wood, "The grand challenges of *Science Robotics*", *Science Robotics*, 2018, eaar7650.

[3] N. Zhu, D. Zhang, W. Wang, X. Li, B. Yang, J. Song, X. Zhao, B. Huang, W. Shi, R. Lu, P. Niu, F. Zhan, X. Ma, D. Wang, W. Xu, G. Wu, G. F. Gao and W. Tan, "A Novel Coronavirus from Patients with Pneumonia in China, 2019" *New England Journal of Medicine*, 2020.

[4] Y. Li, "Deep Reinforcement Learning: An Overview", 2017.

[5] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I", *IEEE Robotics & Automation Magazine*, 2006.

[6] T. T. Sung, C. Kim, K. Lee and Chae-Bong Sohn, "Exploring Navigation using Deep Reinforcement Learning", *International Journal of Applied Engineering Research*, 2018, pp. 14447-14450.

[7] A. Faust, O. Ramirez, M. Fiser, K. Oslund, A. Francis, J Davidson and L. Tapia, "PRM-RL: Long-range Robotic Navigation Tasks by Combining Reinforcement Learning and Sample-Based Planning", in *Proceedings - 2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

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

[9] M. D. M. Rahman, S. M. H. Rashid and M. M. Hossain, "Implementation of Q learning and deep Q network for controlling a self balancing robot model", *Robotics and Biomimetrics*, 2018.

[10] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping SLAM: part II", *IEEE Robotics & Automation Magazine*, 2006.

[11] S. Wang, Y. Li, Y. Sun, X. Li, N. Sun, X. Zhang, and N. Yu, "A localization and navigation method with ORB-SLAM for indoor service mobile robots", *2016 IEEE International Conference on Real-time Computing and Robotics RCAR*, 2016.

[12] A. Josep, P. Yvan, S. Joaquim and L. Xavier, "The SLAM problem: a survey", *Frontiers in Artificial Intelligence and Applications*, 2008.

[13] K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey", *(IEEE) Signal Processing Magazine,* 2017.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg and D. Hassabis, " Human-level control through deep reinforcement learning", *Nature*, 2015.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, " Playing Atari with Deep Reinforcement Learning", *NIPS*, 2013.

[16] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", 2014.

[17] R. Amsters and P. Slaets, "Turtlebot 3 as a Robotics Education Platform", Robotics in Education, Springer International Publishing, 2019.

[18] A. Rasouli and J.K. Tsotsos. "The Effect of Color Space Selection on Detectability and Discriminability of Colored Objects.", 2017.

[19] G. Ciaburro, "Keras Reinforcement Learning Projects", *Packt Publishing*, 2018.

[20] R. K Megalingam, C. R. Teja, S. Sreekanth and A. Raj, "ROS based Autonomous Indoor Navigation Simulation Using SLAM Algorithm", International Journal of Pure and Applied Mathematics, 2018, pp 199-205.

[21] G.-Z. Yang, B. J. Nelson, R. R. Murphy, H. Choset, H. Christensen, S. H. Collins, P. Dario, K. Goldberg, K. Ikuta, N. Jacobstein, D. Kragic, R. H. Taylor, M. McNutt, Combating COVID-19—The role of robotics in managing public health and infectious diseases. *Sci. Robot.* **5**, eabb5589 (2020).