





Deep deterministic policy gradient reinforcement learning for collision-free navigation of mobile robots in unknown environments

Mobil robotların bilinmeyen ortamlarda çarpışmasız gezinmesi için derin deterministik politika gradyan takviyeli öğrenme

Taner YILMAZ^{1*} , Omur AYDOGMUS² 

^{1,2}Department of Mechatronics Engineering, Faculty of Technology, Firat University, Elazig, Turkey.

¹t.yilmaz@firat.edu.tr, ²oaydogmus@firat.edu.tr

Received: 07.04.2023
Accepted: 15.05.2023

Revision: 05.05.2023

doi: 10.5505/fujece.2023.85047
Research Article

Abstract

Learning how to navigate in unfamiliar environments is a critical skill for AI-powered mobile robots. Traditional methods for robot navigation typically involve three key steps: positioning, mapping, and route planning. However, in unknown environments, these methods can become outdated because route planning requires an obstacle map. Moreover, classical approaches may become trapped at a local maximum as the environment becomes more complex, which can negatively impact the system's success. Therefore, it is crucial to address collision avoidance in autonomous navigation, both in static and dynamic environments, to ensure that the robot reaches the target safely without any collisions. In recent years, heuristic approaches have gained importance due to their proximity to human behavioral learning. In this paper, we will examine the advantages and disadvantages of using Deep Deterministic Policy Gradient Reinforcement learning methods to guide the robot from the starting position to the target position without colliding with static and dynamic obstacles. By using a reinforcement learning method, the robot can learn from its experiences, make informed decisions, and adapt to changes in the environment. We will explore the efficacy of this method and compare it to traditional approaches to determine its potential for real-world applications. Ultimately, this paper aims to develop a robust and efficient navigation system for mobile robots, which can successfully navigate in unknown and dynamic environments. For this purpose, the system was tested for 100 episodes and the results showed a success rate of over 80%.

Keywords: Mobile robot, machine learning, reinforcement learning, obstacle avoidance robots

Özet

Bilinmeyen ortamlarda navigasyon öğrenmek, yapay zekâ destekli mobil robotlar için önemli bir beceridir. Robot navigasyonunda geleneksel yöntemler genellikle konumlandırma, haritalama ve rota planlama olmak üzere üç ana adım içerir. Ancak, bilinmeyen ortamlarda rota planlama engel haritası gerektirdiğinden bu yöntemler geçerliliğini koruyamamaktadır. Ayrıca, çevre karmaşık hale geldikçe klasik yaklaşımlar yerel maksimumda sıkışabilir ve sistem başarısını olumsuz etkileyebilir. Bu nedenle, otonom navigasyonda çarpışma önlemeye özellikle statik ve dinamik ortamlarda dikkat etmek, robotun hedefe güvenli bir şekilde çarpışmadan ulaşmasını sağlamak önemlidir. Son yıllarda, heuristik yaklaşımlar insan davranışsal öğrenime yakınlığı nedeniyle önem kazanmıştır. Bu çalışmada, Derin Belirgin Politika Gradyan Takviyeli Öğrenimini kullanarak robotun, başlangıç pozisyonundan hedef pozisyonuna çarpışmadan ilerlemesi için yönlendirilmesinin avantajları ve dezavantajları incelenecektir. Pekiştirmeli öğrenim yöntemi kullanılarak, robot deneyimlerinden öğrenebilir, bilinçli kararlar alabilir ve çevredeki değişikliklere uyum sağlayabilir. Bu yöntemin etkililiği araştırılacak ve gerçek dünya uygulamaları için potansiyeli, geleneksel yöntemlerle karşılaştırılarak belirlenecektir. Sonuç olarak, bu çalışma, bilinmeyen ve dinamik ortamlarda başarılı bir şekilde gezinebilen, sağlam ve verimli bir mobil robot navigasyon sistemi geliştirmeyi amaçlamaktadır. Bu amaçla alınan sonuçlara göre sistem 100 bölüm için test edilmiş ve sonuçlar %80'in üzerinde bir başarı oranı göstermiştir.

Anahtar kelimeler: Mobil robotlar, makine öğrenmesi, takviyeli öğrenme, engelden sakınan robotlar

*Corresponding author

1. Introduction

Robot technology is advancing rapidly, enabling machines to mimic and perform human-like tasks. This has always been an exciting research topic for scientists. Initially, researchers believed that by enhancing computers' abilities to think, hear, speak, recognize, and see, they could help them perceive the outside world like humans and intelligently solve problems. Machine learning plays a vital role in enabling robots to fulfill tasks in accordance with users needs. One such machine learning technique is reinforcement learning, a hybrid model that encompasses both supervised and unsupervised learning techniques[1]. It is now widely utilized in various applications, including health, logistics, industrial, military, and other areas, to improve human life. Autonomous vehicles are another groundbreaking development in robot technology. They can operate without human intervention through automatic control system hardware and software, thanks to advances in artificial intelligence, the Internet of Things, and engineering. These vehicles can perform tasks such as reaching a destination and avoiding obstacles through their automatic control system.

Safety is a significant factor in autonomous robot applications, especially when it comes to collision issues that are challenging to solve, particularly in complex environments with obstacles. As a result, trajectory planning and real-time collision avoidance are essential. However, the available strategies for achieving this are often computationally heavy and complex. Although path planning with model-free controllers is light in computation, it is conservative[2]. In contrast, model-based approaches such as Model Predictive Control integrate global planning with local avoidance[3], but their implementation through motion primitive search requires substantial computation[4]. Additionally, such methods have a limited scope as they rely on specific model dynamics and sensor-based obstacle position estimation[5]. To overcome these challenges, this study proposes a novel learning algorithm (depicted in Figure 1) that directly generates control commands from sensor inputs.

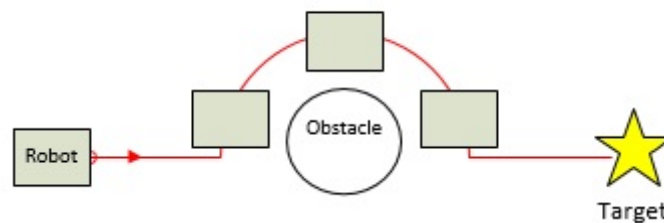


Figure 1. Mobile robot movement

The policy shown in Figure 1 is utilized to learn collision avoidance and path-following maneuvers (indicated in red) and is able to generalize to previously unseen scenarios while avoiding obstacles to reach the goal. In this structure, control is generated directly from the sensors, eliminating the need to estimate obstacle locations. Furthermore, neural networks are more efficient compared to traditional methods, making them the preferred choice[6]. Learning can also be integrated with movement planning. Modelless reinforcement learning, a type of method that learns control policy through environmental interaction, is the most commonly used approach for learning a control policy[7].

In this study, the application of Reinforcement learning techniques to guide a mobile robot from a desired starting position to a desired target position in an environment with static and dynamic obstacles is examined with a robot simulator called CoppeliaSim. The environment is considered a Markov Decision Process and has states and rewards corresponding to each situation. The agent or mobile robot tries to move from a starting location to a target location, collecting maximum rewards and also trying to avoid obstacles. The environmental states observed by the robot are its target position, its own position, linear and angular velocities, as well as its distance from obstacles. The robot gets a small negative reward for every move it takes so it has to move on to the next state, if it hits obstacles it gets a big negative reward. If he hits the goal, he gets a big positive reward. A reward component of its speed can also be used to make the robot move faster.

2. Materials and Method

2.1. Q learning

Q-learning is a model-independent algorithm for reinforcement learning, utilized for learning a policy that guides the agent on the actions to execute based on specific conditions. This algorithm can solve problems with stochastic transitions and rewards without any adaptation and does not require an environment model. The main goal of Q-learning is to find the optimal policy for a finite Markov decision process by maximizing the expected value. This is achieved by calculating the percentage of the total reward over all consecutive steps starting from the current state. Q-learning calculates the action-choice policy for any FMDP given an optimal infinite discovery time and a partially random policy. The reward-returning function used to provide reinforcement is named "Q," which represents quality of an action performed in a particular situation. The weight of a transition from state δ_t to the future is determined by the expression γ^{δ_t} , where the discount factor γ is a value between 0 and 1, and it assigns a higher significance to the rewards obtained earlier in comparison to the rewards obtained later. The discount factor γ can also be interpreted as the probability of success at every δ_t . The algorithm can be approximated as shown in equation (1).

$$Q: S \times A \rightarrow \mathbb{R} \quad (1)$$

The basis of this equation is a Bellman equation, which is a simple value iteration update using the weighted average of the old value and the new information as shown in the equation below[8].

$$Q^{new}(s_t, a_t) \leftarrow Q^{old}(s_t, a_t) + \alpha \left(r_t + \gamma \max_a Q^{optimal}(s_{t+1}, a) - Q^{old}(s_t, a_t) \right) \quad (2)$$

The Q-learning algorithm terminates at the final state s_{t+1} , marking the end of an episode. Q-learning is not limited to episodic tasks and can continue learning iteratively. As an iterative algorithm, Q-learning assumes an initial condition prior to the first update. By using high initial values known as "optimistic initial conditions," the update process progresses based on the initial condition, independent of the chosen action. Subsequently, the initial conditions can be reset using the first reward, r .

Q-learning has been observed to work well for problems with discrete action spaces and a small number of states. However, it is not suitable for problems such as navigation or balancing an inverted pendulum on a car with continuous motion space. This is because finding the optimal policy in continuous spaces requires action optimization at every time step, which is too slow to be practical with large, unconstrained function approximators and complex action spaces. This is one of the drawbacks of the Q-learning algorithm.

2.2. Deep Q learning

Q-learning, a simple yet powerful algorithm, creates a "cheat sheet" for the agent to guide it towards the optimal action. However, the experiments conducted in this study show that deriving the Q-value of new states solely from previously discovered states is not feasible. This leads to two problems: first, as the number of states increases, the table size and frequency of updates required also increase. Second, exploring and building the necessary Q-table takes an unrealistic amount of time in most scenarios.

To address these challenges, deep Q-learning utilizes a neural network. The network takes all possible actions as inputs and generates corresponding Q-values as outputs for each action[9]. The algorithm proceeds through the following steps in order:

- All past experiences are stored in memory by the user.
- The next action is determined by the output of the maximum value owned by the Q network.
- The loss function used is the mean square error between the predicted Q-value and the target Q-value, Q^* .

This essentially amounts to a regression problem. However, in the context of reinforcement learning, we do not have access to the actual target value. To address this, the Q-value update equation is derived from the Bellman equation.

$$\left(r_{t+1} + \gamma \max_a Q^{optimal}(s_{t+1}, a) \right) \quad (3)$$

Equation 3 depicts the target value. Although we can attempt to estimate its value, the unbiased true reward r enables the network to update its gradient using backpropagation and eventually converge to the correct value. Deep Q-Learning is shown in the figure below:

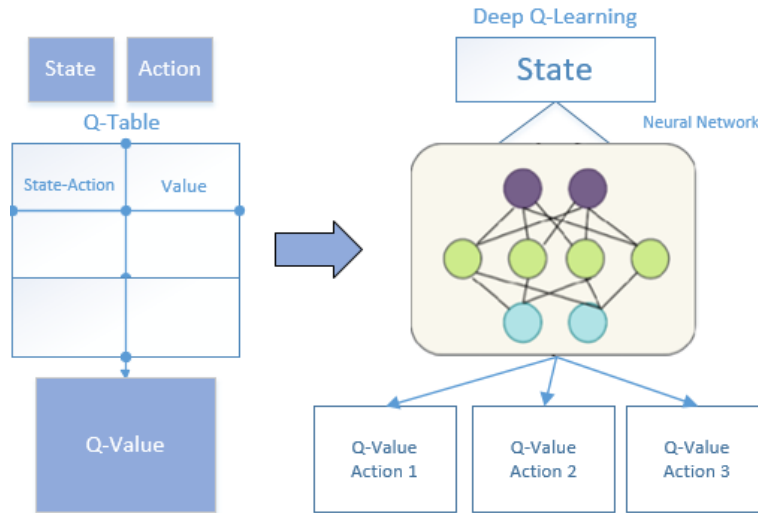


Figure 2. Representation of Deep Q-Learning

2.3. Deep deterministic policy gradient

Among the plethora of available reinforcement learning algorithms, the Deep Deterministic Policy Gradient (DDPG) is highly regarded for its proficiency in performing tasks in continuous action domains.

DDPG employs a unique learning process where it learns both a Q-function and a policy simultaneously. This is achieved by utilizing the Bellman equation and non-policy data to learn the Q-function, and subsequently using the Q-function to learn the policy[10]. This approach is strongly associated with and motivated by Q-learning. Specifically, given the optimal action-value function $Q^*(s,a)$, the optimal action $a^*(s)$ for any given state can be determined by solving the equation.

$$a^*(s) = \arg \max_a Q^*(s, a) \quad (4)$$

DDPG is a reinforcement learning algorithm designed to address the challenge of continuous action spaces. It achieves this by combining two types of networks, namely actor and critic networks. The actor network takes environment states as input and predicts the actions that the agent will take in its next step. Meanwhile, the critic network evaluates the policy function predicted by the actor based on the temporal difference error. In environments with discrete action spaces, evaluating the Q-values for each action is straightforward. However, in continuous action spaces, evaluating the space exhaustively is not feasible, and finding the maximum Q-value for a given state can be computationally expensive. DDPG solves this problem by assuming that the function $Q^*(s, a)$ is differentiable with respect to the action argument[11]. This allows an efficient, gradient-based learning rule for the policy $\mu(s)$, enabling the approximation of the maximum Q-value with $\max_a Q(s, a) \approx Q(s, \mu(s))$ [12]. By leveraging the actor and critic

networks, along with experience repetition, DDPG enables efficient learning in environments with continuous action spaces.

DDPG consists of two types of networks called actor and critic networks. The actor network takes environment states as input and predicts the actions that the agent will take in its next step. The critical network is used to evaluate the policy function predicted by the actor according to the temporal difference (TD) error. This combination of actor network and critic network, together with experience repetition, creates the DDPG.

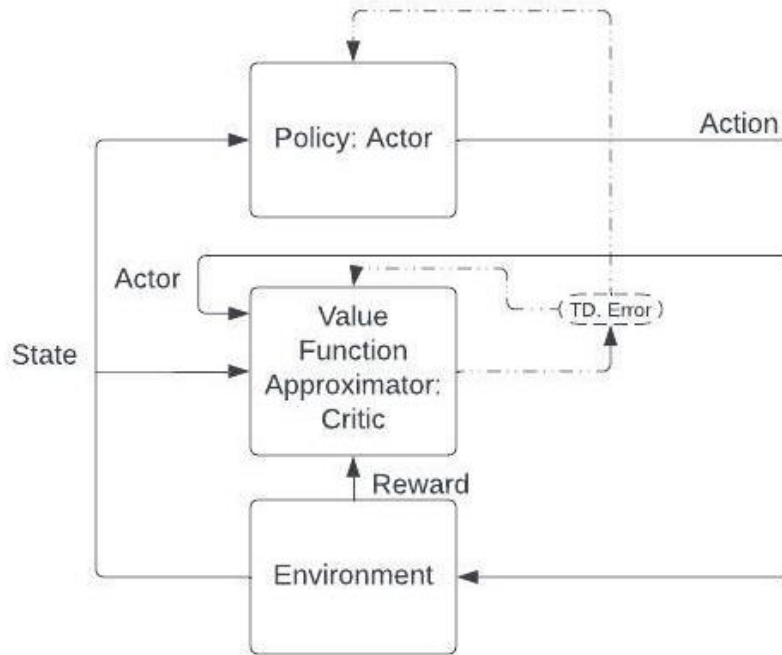


Figure 3. Representation of DDPG

It is clear from that the DDPG algorithm utilizes a parameterized actor function $\mu(s|\theta^\mu)$ to establish the current policy by deterministically mapping states to actions[13]. Similar to Q-learning, the critic $Q(s, a)$ is trained using the Bellman equation[14]. To update the actor, chain rule is applied to the expected return from the initial distribution J in relation to the actor parameters:

$$\nabla_{\theta^\mu} J = E_{st\rho\beta} [\nabla_a Q(s, a|\theta^Q) \Big|_{s=s_t, a=\mu_t} \times \nabla_{\theta^\mu} \mu(s|\theta^\mu) \Big|_{s=s_t}] \quad (5)$$

2.4. Deep deterministic policy gradient with hindsight experience replay

Experience replay is a technique introduced by Lin (1992) that involves training an agent using transitions sampled from a buffer of previously experienced transitions. A transition consists of four elements: the current state (s), the action taken (a), the reward received (r) after taking that action in the current state, and the resulting next state (s'). At each time step, the current transition is added to the playback buffer, and a subset of transitions are sampled from the buffer to train the agent. To address the challenges posed by sparse rewards and complex reward structures, the HER (Hindsight Experience Replay) algorithm was developed as an extension of experience replay[15]. HER uses a new parameter called the "goal" which includes both a desired goal and an achieved goal. In the context of the task, the desired goal refers to the objective that the agent is expected to achieve, while the achieved goal represents what the agent has accomplished thus far.

The central concept underlying HER involves transforming failed experiences into successful ones, allowing the agent to learn from them. Specifically, even if the desired goal is not achieved at the current time, if the agent has

completed a related goal, that achieved goal can replace the desired goal so that the agent can learn from the successful experience. With repeated learning iterations, the agent can eventually achieve the desired goal and complete the task with sparse rewards.

For example, if the agent attempts to reach the goal state G from the initial state S , but fails and ends up in some state S' , this trajectory is cached into the replay buffer[16]. With HER, the achieved goal of reaching state S' can be used as a substitute for the desired goal of reaching state G , allowing the agent to learn from this successful experience even though the original goal was not achieved.

$$(S_0, G, a_0, r_0, S_1), (S_1, G, a_1, r_1, S_2), \dots, (S_n, G, a_n, r_n, S') \quad (6)$$

In the HER approach, the reward received at step k of the episode is denoted as r subscript k , and the action taken at step k is represented as a subscript k . The central concept of HER involves considering the possibility that the intended goal was actually state S' all along. In this hypothetical scenario, the agent successfully reached the goal and obtained the associated positive reward. Thus, in addition to recording the actual trajectory, an alternative trajectory is also saved in the memory.

$$(S_0, S', a_0, r_0, S_1), (S_1, S', a_1, r_1, S_2), \dots, (S_n, S', a_n, r_n, S') \quad (7)$$

The aforementioned trajectory is fictitious and is driven by the cognitive capacity of humans to derive useful knowledge from unsuccessful trials. Additionally, it is important to observe that in the hypothetical trajectory, the reward obtained at the terminal stage of the episode represents a positive reward procured by accomplishing the assumed objective. The incorporation of the simulated trajectories into our replay buffer guarantees that irrespective of the performance of our policy, we are always presented with a corpus of favorable rewards from which to acquire knowledge.

This project involves the use of a simulation tool called CoppeliaSim (previously known as v-rep) by Coppelia Robotics, along with OpenAI's gym environment[17]. The Coppelia Sim provides a mobile robot and a scene in which the robot can move around. When coupled with the gym environment, we obtain the states and information about the environment without requiring any additional sensors. The environments look like the Fig:4.

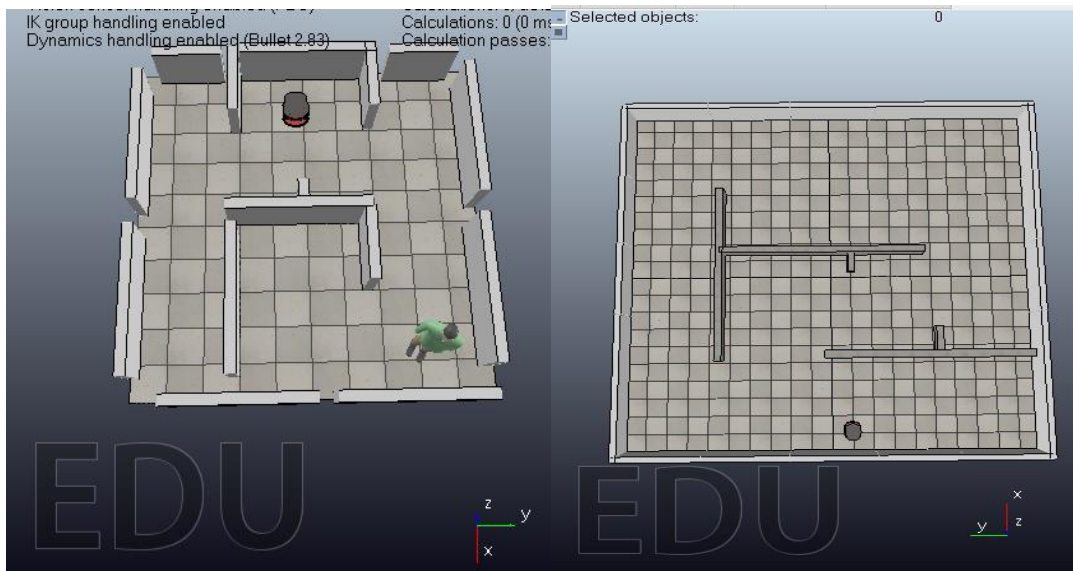


Figure 4. Examples of environment

To obtain information about the environment, we utilized "PyRep"[18], a toolkit designed for robot learning research that is built on top of CoppeliaSim and interfaces with OpenAI gym. Pyrep facilitates the testing of reinforcement learning algorithms developed in Python programming language by utilizing the integration of OpenAI Gym within

the CoppeliaSim platform, allowing for experimentation under various environmental conditions. Additionally, Pyrep enables real-time monitoring and potential 3D visualization of robot and object movements within the simulation environment. These capabilities provide robotics researchers and developers with the means to assess and enhance robot control algorithms by conducting tests and refinements using simulated scenarios that closely mimic real-world conditions[19]. Therefore, Pyrep is a preferred choice in the study. The visualization of Pyrep's working structure is shown in Figure 5.

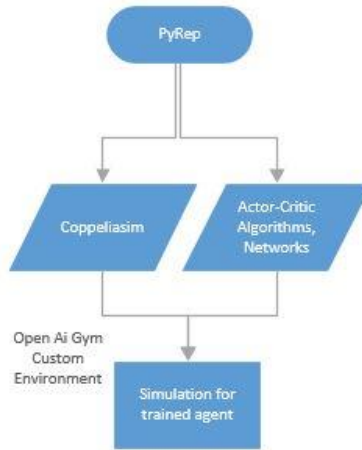


Figure 5. The visualization of Pyrep's working structure.

For the actor and critic neural networks, we opted to use TensorFlow due to its ease of use and robust training methods. Four networks were utilized in total: the actor, critic, target actor, and target critic. Shallow neural networks were used for all four, each with 2 hidden layers consisting of 200 and 100 neurons, respectively. The activation function employed was the relu function, which is known for its effectiveness in training. Adam optimizer was used to train both the actor and critic networks, with the objective of reducing the mean square error of the output between the critic and the target critic. To effectively train the agent, a well-designed reward function that considers the variables affecting the robot's performance is crucial. We formulated a reward function that takes into account all the critical factors, and the equation for this function is presented in equation(8).

$$R = \begin{cases} 1, & d < d_{th} \\ -1, & \exists D < d_{collision} \\ -0.1 & \exists D < d_{proxth} \\ -0.5 & V_L \cos(\theta) > 0.1 \end{cases} \quad (8)$$

V_L represents the linear velocity of the mobile robot, θ represents the heading angle, D refers to the vector of distance read from ultrasonic sensors, d denotes the distance between the robot and goal position, $d_{collision}$ is the distance at which a collision occurs, d_{th} is the threshold distance between the robot and goal position, and d_{proxth} is the safety distance threshold for regular operation.

3. Results

The CoppeliaSim robotic simulator was utilized in this study to facilitate the navigation of a robot within a 2-dimensional environment. The virtual robot utilized several ultrasonic sensors to detect obstacles in order to navigate.

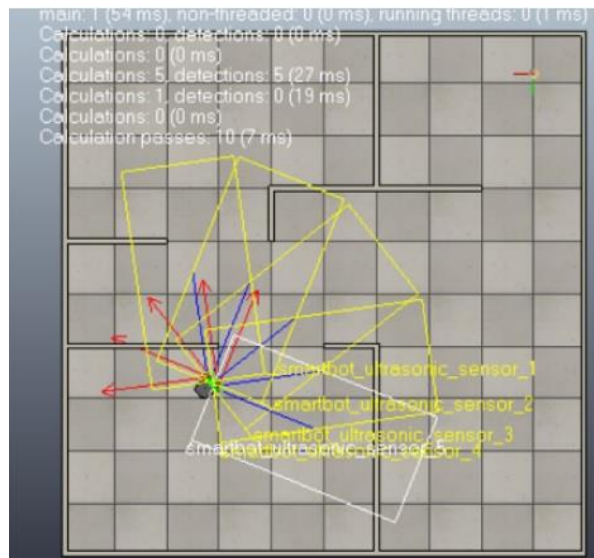


Figure 6. Example of illustrated gathering local information

The field of detection for these sensors is illustrated in Figure 5. The range of detection is sufficiently wide to enable the robot to detect multiple obstacles in its vicinity, which helps it avoid collisions. The red arrows in the figure indicate the presence of obstacles in close proximity, while the blue lines indicate a clear path for the robot to follow.

Multiple experiments were conducted using various start and end points. 15 pairs were used and trained for 5000 episodes each. Initially, simple reward functions were utilized that penalized the robot for collisions and rewarded it for reaching the goal. Shown in figure 6.



Figure 7. Example of final robot position

However, this proved to be an inadequate reward formulation, and a more comprehensive reward system was developed, as mentioned above. After training for 75000 episodes, the system was tested for 100 episodes, and the results, as shown in Figure 7 and Figure 8, demonstrated a success rate well above 80%.

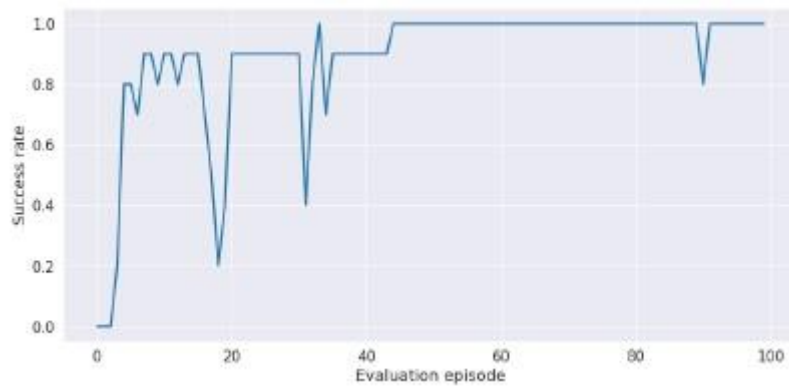


Figure 7. Example success rate of algorithm

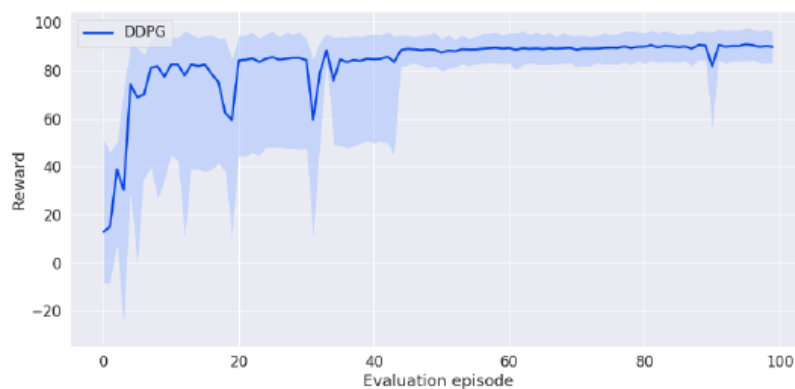


Figure 8. Reward-Evaluation Episode

Based on the outcomes of this study, it can be inferred that Deep Deterministic Policy Gradient with HER performs effectively within a simulated environment. Despite initial evaluation failures, the algorithm rapidly recovers and attains a higher success rate in subsequent episodes.

4. Conclusion

As a result of this project, it can be concluded that the Deep Deterministic Policy Gradient performs quite well in a simulation environment. Although it may initially fail in evaluation episodes, it can quickly recover and achieve a higher success rate in later episodes. As an extension of this project, incorporating the advantages of Hindsight Experience Replay to learn from previous transition experiences is recommended. Additionally, it is possible to improve navigation strategies by better understanding the environment through the incorporation of vision sensors for improved perception by the robot.

5. Discussion

According to the results, it has been observed that the algorithm used is successful. Especially due to its ease of programmability and achieving a success rate of 90% or higher in relatively short scene numbers compared to the studies found in the literature up to a certain level of complexity, it has been observed that the algorithm used has an advantage. However, as the uncertainty level of the environment increases (such as extremely complex mazes, narrow passages, etc. beyond the examples of the environment used in the study), there are deviations in the robot's success. In this context, it has been determined that this situation creates a disadvantage for the algorithm used. To eliminate this situation, it is anticipated that complex algorithms such as TD3, PPO, A3C, etc. supported by curious learning module could be suitable for implementation in environments with higher uncertainty and complexity in future studies.

6. Acknowledgments

We would like to thank Firat University for their contribution to the studies in this research.

7. Author Contribution Statement

Taner Yılmaz played a role in designing the study and conducting the literature review, as well as contributing to analysis of the results. Ömür Aydoğmuş contributed to conceptualization of the article, as well as proofreading and editing content.

8. Ethics Committee Approval and Conflict of Interest

It is not necessary to obtain approval from an ethics committee for the article being prepared. Additionally, there are no conflicts of interest with any individuals or institutions in relation to the content of the article.

9. References

- [1] Tufenkci S, Alagoz BB Kavuran G, Yeroglu C, Herencsar N, Mahata S. "A theoretical demonstration for reinforcement learning of PI control dynamics for optimal speed control of dc motors by using twin delay deep deterministic policy gradient algorithm". *Expert Syst. Appl.*, 213, 119192, 2023.
- [2] Sampedro C, Rodriguez-Ramos A, Bavle H, Carrio A, de la Puente P, Campoy P. "A Fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques". *J. Intell. Robot. Syst. Theory Appl.*, 95(2), 601–627, 2019.
- [3] Alavizadeh H, Alavizadeh H, Jang-Jaccard J, "Deep Q-learning based reinforcement learning approach for network intrusion detection". *Computers*, 11(3), 1–19, 2022.
- [4] Uav F, Tan Z, Lyu Y, Lu H, Pan Q. "Motion primitives-based and Two-phase Motion Planning for". 2022.
- [5] Krishnan S, Boroujerdian B, Fu W, Faust A, Reddi VJ. "Air Learning: a deep reinforcement learning gym for autonomous aerial robot visual navigation". *Machine Learning*, 110(9), 2021.
- [6] Stevsic S, Nageli T, Alonso-Mora J, Hilliges O, "Sample Efficient learning of path following and obstacle avoidance behavior for quadrotors". *IEEE Robot. Autom. Lett.*, 3(4), 3852–3859, 2018.
- [7] Lockwood O, Si M. "A review of uncertainty for deep reinforcement learning". *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entertain*, 18(1), 155–162, 2022.
- [8] Clifton J, Laber E. "Q-Learning: Theory and Applications". *Annual Review of Statistics and Its Application*, 279–303, 2020.
- [9] Brockman G. *et al.*, "OpenAI Gym". 1–4, 2016, [Online]. Available: <http://arxiv.org/abs/1606.01540>.
- [10] Saglam B, Cicek DC, Mutlu FB, Kozat SS. "Off-Policy correction for actor-critic algorithms in deep reinforcement learning". 2022.
- [11] Tsai J, Chang CC, Ou YC, Sieh BH, Ooi YM. "Autonomous driving control based on the perception of a lidar sensor and odometer". *Appl. Sci.*, 12(15), 2022.
- [12] Luong NC. *et al.*, "Applications of deep reinforcement learning in communications and networking: a survey". *IEEE Commun. Surv. Tutorials*, 21(4), 3133–3174, 2019.
- [13] Xiang J, Li Q, Dong X, Ren Z, "Continuous control with deep reinforcement learning for mobile robot navigation". *Proc. - 2019 Chinese Autom. Congr. CAC 2019*, 1501–1506, 2019.
- [14] Tsingenopoulos I, Preuveneers D, Joosen W, "AutoAttacker: a reinforcement learning approach for black-box adversarial attacks". *Proc. - 4th IEEE Eur. Symp. Secur. Priv. Work. EUROS PW 2019* 229–237, 2019.
- [15] Andrychowicz M. *et al.*, "Hindsight Experience replay (279 cites)". *Adv. Neural Inf. Process. Syst.*, 2017-Decem, no. Nips, 5049–5059, 2017.
- [16] Lindner T, Milecki A, Wyrwał D, "Positioning of the robotic arm using different reinforcement learning algorithms". *Int. J. Control. Autom. Syst.*, 19(4), 1661–1676, 2021.
- [17] Lucchi M, Zindler F, Muhlbacher-Karrer S, Pichler H. "Robo-gym - an open source toolkit for distributed deep reinforcement learning on real and simulated robots". *IEEE Int. Conf. Intell. Robot. Syst.*, 5364–5371, 2020.
- [18] James S, Freese M, Davison AJ. "PyRep: bringing v-rep to deep robot learning". 1–4, 2019.
- [19] James S, Davison AJ, Johns E. "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task". *CoRL*, 1–10, 2017.