A Survey of Research and Applications of Optimal Path Planning Based on Deep Reinforcement Learning

*Zifan Song**

University College London, Electrical and Electronic Engineering Department, WC1E 6BT, United Kingdom

Abstract. This study focuses on the comparison between conventional path planning techniques and Deep Reinforcement Learning-based path planning technology. Traditional algorithms including the A* algorithm, Dijkstra algorithm, and random sampling techniques, which perform well in static contexts, become inefficient in dynamic and complicated high-dimensional settings due to high computing costs. In contrast to DRL, a robot is charged with detecting sensor data linked with optimum path-planning using MDP (Markov Decision Process). Such robots independently study their surroundings while choosing the proper path. This implies it gets better at adapting to new settings and changes. This research looks at the real-time performance of three standard DRL algorithms: Proximal Policy Optimization, Deep Deterministic Policy Gradient, and Deep Q-Network in real-world settings. This work also discusses the drawbacks of DRL in path planning, i.e., high processing requirements, extended training times, weak generalization capabilities, and so on. Future initiatives include creating efficient training algorithms, mechanisms that increase model generalization, and optimizing them using classical techniques.

1 Introduction

Because of the fast advancements in robotics and artificial intelligence, there is a rising demand for improved control systems, and path planning, which is one of the fundamental challenges for autonomous mobile robots, has received a lot of attention. Optimal path planning thus entails not only determining the shortest distance between two points, but also accounting for energy time and dynamic impediments in the environment [1]. Conventional path planning approaches [2].

Recent advancements in Reinforcement Learning (RL) and Deep Learning (DL) have paved the way for novel approaches to path planning. To be able to do so in an unknown, highly dynamic environment without an instructor, deep learning (feature extraction) is combined with reinforcement learning (decision making) to create the wonderful world of Deep Reinforcement Learning, or DRL. This increases the robustness and versatility of the path planning algorithm [3][4].

* corresponding author: Zceeonb@ucl.ac.uk

Deep reinforcement learning (DRL) has achieved advancements in a variety of domains. For example, Mnih and his colleagues used DeepQ networks (DQN) in gaming AI to exhibit great learning [5]. The Deep Deterministic Policy Gradient prompted DQN to include continuous action space in its list, since RL can now perform well in fields like as autonomous driving and robot control [4].

Deep reinforcement learning algorithms are making inroads into route planning. Robots can automatically determine the best path planning approach for avoiding environmental contact by modeling the problem as a Markov Decision Process and then applying MDP to solve it. These approaches can swiftly service applications that require real-time adaptation while simultaneously adjusting to environmental uncertainties [6]. DRL approaches are more generic and scalable. DRL techniques significantly outperform traditional systems in managing high-dimensional status and action data.

However, despite its excellent path planning performance, DRL remains distant from practical implementations due to several obstacles. Path planning challenges are complicated and multidimensional, which contributes to this. However, deep reinforcement learning is computationally costly and time-consuming to train. Second, we still need an urgent answer on how to discover a meaningful incentive for teaching the model a suitable path planning strategy [7][8]. Ultimately, more work needs to be done to enhance the deep reinforcement learning model's capacity to generalize in a real, complicated world [9].

In summary, the research aims to give a broad overview of the advanced deep reinforcement learning-based path planning methods, explore their latest developments across a range of application domains, and evaluate prospective directions for development in the future.

2 Path planning Existing path planning methods

Autonomous systems have long utilized established path planning techniques. These techniques primarily consist of random sampling techniques, algorithms based on geometric programming, and techniques based on graph searches. When dealing with high-dimensional dynamic situations, they progressively show their limitations, although they perform very well in offline planning and static contexts.

2.1 Utilizing Graph Search for Path Planning

Graph search is one of the simplest and most often used path planning techniques. Finding the best route from the starting node to the destination node and identifying the nodes within the specified state space are its two main goals. These methods usually involve building a topological graph and conducting searches over it. The most popular graph search algorithms are listed below:

- **Dijkstra algorithm**: It is among the oldest graph search methods for finding the shortest routes in weighted graphs. To make sure that the selected path is always locally optimal, it expands the nodes layer by layer until it reaches the global optimal path [10]. For instance, in Figure 1, A, B, E, and D is the most efficient route from A to D. The Dijkstra algorithm is inefficient when working with large-scale graphs because, despite guaranteeing an optimal solution, its computational cost is $O(V^2)$.
- **A* algorithm**: It's a tweaked version of Dijkstra's method that uses heuristic estimations like Manhattan or Euclidean distance to expedite the search[2]. It is extensively employed in numerous real-world applications and has the ability to greatly shrink the search space. However, the heuristic function selection has a significant impact on how well A*

performs. When dealing with intricate settings and imprecise heuristic functions, its effectiveness could be significantly diminished.

- **B* Algorithm**: Path planning in an unknown environment is possible with this method, which is an improved version of A*. Because B prepares the route dynamically by updating the map data online, it performs better while navigating partially visible or dynamic environments [11].

Consequently, the inability to adjust to changes in dynamic contexts and their high computational cost are the drawbacks of graph search-based techniques. Furthermore, these approaches often work well in a known, static context and are not very effective in unpredictable real-world circumstances.

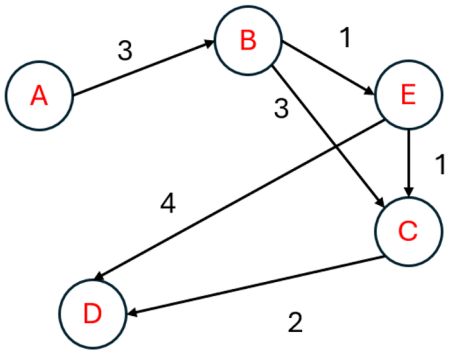

Fig.1. Example of using Dijkstra algorithm

2.2 A geometric programming-based approach to path design

Geometric programming techniques are especially useful for locating smooth paths in continuous space since they employ geometric information for path planning.

- **Voronoi diagram**: Using a voronoi diagram, the environment is divided into zones where impediments have an impact in order to design a path from a starting point to a goal. This approach has the benefit of automatically maintaining a wide safety gap, making the path more cautious and appropriate in situations requiring a high level of security [12]. Further optimization is necessary since Voronoi diagrams typically result in longer paths that are not the shortest paths.
- **Gradient Descent**: One continuous space-based optimization method that may be used to problems with optimum path planning is gradient descent. By gradually altering the path to shorten the path or use less energy, it reaches the local optimum of the path [13]. Nevertheless, gradient descent frequently leads to local optimal solutions; therefore, in order to prevent this issue, additional techniques, like simulated annealing, must be used in conjunction with gradient descent.
- **Artificial Potential Field (APF)**: APF is a well-liked real-time path planning method that directs the robot toward the objective by using "force fields" to represent target attraction and obstacle repulsion [14]. In complicated settings, particularly small spaces or U-traps, this approach is prone to falling into local minima, while being computationally efficient and appropriate for real-time applications.

As can be seen, the geometric programming approach produces smooth routes and exhibits strong continuity, making it appropriate for scenarios requiring a high degree of path quality. Nevertheless, these approaches are ineffective in high-dimensional complex settings because of their local optimization features, which make it simple for them to enter local optima.

2.3. Random sampling method

Probabilistic Roadmap and Rapidly-exploring Random Tree are two random sampling techniques that have demonstrated significant promise in resolving high-dimensional path planning issues. These techniques randomly sample a continuous space to gradually build viable pathways.

- **Rapidly-expanding Random Tree**: RRT is a sample-based technique that randomly develops the tree to explore a path in a high-dimensional environment. Each expansion, until the goal is identified or enough tree structure has been developed, is determined by a random direction from the current node [15]. RRT is excellent at dealing with complicated, high-dimensional spaces, but it frequently produces coarse paths that need further smoothing.
- **Probabilistic Path Graph**: In order to carry out path planning, PRM creates a graph linking randomly selected locations in the configuration space [16]. PRM is often used for offline path planning, and it is suited for usage in known contexts since the resulting path graph may be multiplexed in numerous jobs. On the other hand, PRM struggles in dynamic settings or highly unpredictable situations.

Compared with the path planning method based on graph search and the path planning method based on geometric programming, random sampling methods (such as RRT and PRM) can quickly find feasible paths in high-dimensional space and can cope with more complex environments, but the paths generated by them are often not smooth and require subsequent path optimization.

3 Deep Reinforcement Acquiring knowledge and using it in route planning

The three main algorithms for deep reinforcement learning theory are Proximal Policy Optimization, Deep Deterministic Policy Gradient, and Deep Q-Network.

3.1 Deep Q-Network, DQN

The deep neural network (DNN) technique aims to approximate the Q-value function. The predicted cumulative reward when action a is carried out in a specific states is indicated by the Q-value function $Q(s,a)$. The fundamental concept of DQN is approximating Q-values using deep neural networks $Q(s,a;\theta)$ and updating the network weights using the Bellman equation. The formula for updates is:

$$\boldsymbol{Q(s,a;\theta) \leftarrow Q(s,a) + \alpha[r + \gamma max Q(s',a';\ \theta^{-}) -\ Q(s,a;\theta)]} \quad (1)$$

$\boldsymbol{\gamma}$ is the discount factor. $\boldsymbol{\theta^{-}}$ is a fixed target network parameter, and $\boldsymbol{\alpha}$ is the learning rate.

Path planning is modelled by DQN as a Markov Decision Process (MDP), in which the robot senses the condition of the environment and decides on the best course of action to take in order to achieve the objective position. To be more precise, the state s often contains information on the robot's current location in relation to its surroundings as well as obstacles and objectives. The robot's present condition allows it to do certain actions, including turning left or right or moving ahead. This is known as the action a. With the use of interactions with the environment—specifically, incentives earned during the robot's movement and state transitions. Similar to how Mnih and his team [5] successfully used DQN to Atari games, where the robot picks up on the controls and uses pixel input to find its way around the game area in order to maximize the score.

DQN has been widely applied to robot route planning in practice. Zhu and colleagues [17], for example, have made significant progress in the area of robot vision navigation by

creating a unique way to apply the Siamese Actor-Critic network model to enhance the robot's ability to generalize to different targets. This approach, in my opinion, effectively addresses the issue that the conventional deep reinforcement learning model must be retrained in front of a new target in order to improve the robot's performance on navigation tasks in intricate interior settings. Nevertheless, the model's performance may be impacted by the complexity of the environment and sensor noise in real-world applications because this experiment still depends on excellent simulated settings (like the AI2-THOR framework) for training. Subsequent investigations may delve deeper into the functionality and enhancement in actual settings.

Furthermore, Mnih et al. [5] demonstrated how DQN may be used to make Atari function; an autonomous DRL agent interacts with game screen pixels and learns navigation in a simulated 3D world through score maximization. Similarly, Mirowski et al. and Eriksson et al. Instead, Zhu [18] integrated auxiliary tasks of loop closure and depth prediction into the asynchronous dominant Actor-Critic (A3C) algorithm to assist UAV navigation in a difficult 3D labyrinth. It improves navigation performance by better comprehending how routes operate and the presence of barriers, resulting in a significant boost in learning efficiency and navigation success. This not only verifies our hypothesis that deep reinforcement learning can function in high-dimensional, dynamic contexts, but it also shows more potential than utilizing it for static benchmarks.

3.2 Proximal Policy Optimization, PPO

The policy gradient-based reinforcement learning approach Proximal Policy Optimization (PPO) restricts policy update change range to stabilize policy optimization. In reinforcement learning, policy gradient approaches directly maximize the policy function $\pi(a|s;\theta)$ to raise cumulative rewards. On the other hand, an excessively large update step size during policy updates might cause the policy to collapse or degrade performance. To guarantee that each policy update does not diverge excessively and preserve training stability, PPO provides the policy update limit method. The objective function of PPO is its fundamental formula:

$$\boldsymbol{L^{CLIP}(\theta) = E_t[min\left(r_t(\theta)\widehat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_t\right]} \quad (2)$$

$\boldsymbol{r_t(\theta)} = \left(\frac{\pi_\theta(\boldsymbol{a_t}|\boldsymbol{s_t})}{\pi_{\theta_{old}}(\boldsymbol{a_t}|\boldsymbol{s_t})}\right)$ is the old and new policy probability ratio that stands for the probability ratio of carrying out the same action $\boldsymbol{a_t}$ under the old policy parameters $\boldsymbol{\theta_{old}}$ and the new policy parameters $\boldsymbol{\theta}$. $\boldsymbol{clip(r_t(\theta), 1-\epsilon, 1+\epsilon)}$ is Clipping function that restricts the level of the policy update so that $\boldsymbol{r_t(\theta)}$ can be in the range of $\boldsymbol{[1-\epsilon, 1+\epsilon]}$. For tiny policy changes, $\boldsymbol{r_t(\theta)\widehat{A}_t}$ is directly optimized. To prevent instability in the training process, the clipping function suppresses changes in policy when they become too significant. The advancing function, denoted as $\widehat{\boldsymbol{A}}_{\boldsymbol{t}}$, indicates how much better action $\boldsymbol{a_t}$ is in state $\boldsymbol{s_t}$. when relative to average performance. The policy is updated in a more advantageous manner using the advantage function.

PPO is used in path planning activities to resolve high-dimensional control issues that are difficult, such route optimization for autonomous vehicles and UAV navigation. PPO functions effectively in both discrete and continuous action spaces when handling decisions made in real time. The instability issue that might arise with conventional policy gradient techniques is avoided by using the policy update approach, which edits to restrict the scope of policy changes. When unexpected impediments or changes in the target's position arise, the robot may swiftly modify its course and continue navigating effectively because to PPO's real-time optimization capacity [19].

Wang et al [20] employed PPO for end-to-end autonomous driving jobs in their research. In this application, the car is driven in a challenging urban area utilizing the PPO algorithm. The vehicle sensor data is sent into the PPO policy network, which generates the appropriate

control signals (e.g., acceleration, steering angle, etc.) to guarantee that the host car is driven safely in a complicated traffic situation. Furthermore, Berner et al. A collection of preprint papers from last autumn [21] reported that a daunting Mountain Dew-stockpiling project called OpenAI Five had managed to vault the low bar required to beat the best humans at Dota 2 after deploying Proximal Policy Optimization (PPO), Generalized Dominance Estimation (GAE), and continuous supervision training. This result indicates PPO's capacity to deal with difficult issues and policy reform.

3.3 Deep Deterministic Policy Gradient, DDPG

Deep Deterministic Policy Gradient (DDPG) is a reinforcement learning technique for continuous-action control. DDPG combines DPG and Q-learning with the actor-critic architecture, where the actor network outputs deterministic actions like policy gradients and the critic assesses them. The DDPG works particularly effectively in high-dimensional continuous control jobs since it just needs to output actual levels. DDPG principally updates two networks: the Actor network and the Critic network. The loss function that follows serves as the basis for updating the Critic network $\boldsymbol{Q}(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\theta^Q})$:

$$L(\boldsymbol{\theta^Q}) = \boldsymbol{E}[(\boldsymbol{Q}(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\theta^Q}) - \boldsymbol{y})^2] \quad (3)$$

Where, the target value y is defined by the following formula:

$$\boldsymbol{y} = \boldsymbol{r} + \boldsymbol{\gamma}\,\boldsymbol{Q}(\boldsymbol{s'},\boldsymbol{\mu}(\boldsymbol{s'}|\boldsymbol{\theta^\mu})\boldsymbol{\theta^Q}) \quad (4)$$

$\boldsymbol{r}$ represents the instantaneous reward earned in the action and state at hand in this formula. The discount factor, or $\boldsymbol{\gamma}$, quantifies how future rewards will affect the choice made today;$\boldsymbol{Q}(\boldsymbol{s'},\boldsymbol{\mu}(\boldsymbol{s'}|\boldsymbol{\theta^\mu})\boldsymbol{\theta^Q})$ is the future reward estimate that is acquired by the action produced by the Actor network $\boldsymbol{\mu}(\boldsymbol{s'}|\boldsymbol{\theta^\mu})$ in the next state $\boldsymbol{s'}$.

Complex control tasks, like robotic arm path planning and autonomous driving trajectory optimization, have been the focus of DDPG applications in path planning. With precise policy optimization, DDPG can manage high-dimensional and complicated path planning jobs. To provide precise control in complicated environments, DDPG can directly output specified action values in jobs with continuous action space [4].

Gu, et al. [22] employed a deep Deterministic Policy Gradient approach that was presented for mobile robot route planning that collects data concurrently and trains asynchronously. The robot with navigator driver can effectively avoid collision, the motion trajectory can still adjust to changing environments, and the original navigation path was effective from start to finish.

4. Deep Reinforcement Learning Path Planning and Conventional Path Planning Comparison

In terms of the path planning, Deep Reinforcement Learning (DRL)-based techniques clearly outperform standard methods in terms of handling dynamic situations, computational cost, generalization capacity, and path quality. I will provide a thorough analysis of the two strategies from a number of angles before delving into their benefits and drawbacks.

4.1 Computational Complexity and resource requirements

It is a waste of time to replan a route when you encounter dynamic changes, particularly if barriers often occur or the area is unfamiliar. Path planning algorithms that are best suited for a static environment, such as Dijkstra and A*, adhere to implicit static principles in their design [23]. DRL may interact with the environment to make real-time policy improvements [23]. This is because DRL outperforms standard methods in complicated dynamic contexts

by using a learning mechanism to optimize the policy in response to feedback in a changing environment. This benefit stems mostly from the fact that DRL models the path planning problem using MDP, enabling it to autonomously determine the best course of action in an uncharted area and react swiftly to abrupt environmental changes [24].

4.2 Computational Complexity and Scalability

Conventional path planning algorithms, like Dijkstra, have an $O(V^2)$ complexity. The computing time increases rapidly as the issue scale increases when working with large-scale graphs or high-dimensional spaces [2]. On the other hand, DRL techniques, like DDPG, use deep neural networks to deal with high-dimensional state and action Spaces. These networks can manage path planning issues in intricate settings and are particularly well-suited for real-time and large-scale data processing applications [25]. DRL can efficiently scale to high-dimensional space and difficult tasks, while having a high training phase computational resource cost and a low execution phase computational complexity[26].

4.3 Paths' optimality and smoothness

Even though pathways may be created fast using random sampling techniques like RRT, these paths are often coarse and need further smoothing processing [27]. Geometric programming may provide reasonably smooth routes, but it cannot ensure the global optimal solution and is prone to local optima [28]. On the other hand, by optimizing the reward function, DRLs like DDPG may create pathways that take into account path length, energy consumption, and smoothness all at once [29]. This is because DRL may directly optimize pathways in the continuous action space; smooth, almost global optimal paths are formed as a result [29].

4.4 Ability to make decisions in real time

Conventional path planning techniques, like APF, depend on preset rules that are hard to adjust to abrupt changes in dynamic settings, particularly when barriers are present. They also have a tendency to become stuck in local optima [30]. DRL is able to swiftly modify the path in response to changes in the environment by dynamically modifying the policy network.[31] By limiting its scope, PPO and other algorithms ensure that the policy update stays on the best path, which significantly enhances the policy update's capacity to respond in real time in complicated contexts. Because of this, DRL performs exceptionally well in real-time path planning tasks like UAV navigation and autonomous driving [31]. Simplified Comparison of Traditional Path Planning vs DRL is shown in the table 1 below.

Table 1. Simplified Comparison of Traditional Path Planning vs DRL

Aspect	Traditional Path Planning	Deep Reinforcement Learning (DRL)
Computational Complexity	High complexity (e.g., Dijkstra $O(V^2)$), inefficient for dynamic changes.	High training cost but efficient in execution, adapts to dynamic environments.

Scalability	Limited scalability, performance degrades with problem size.	Highly scalable, handles high-dimensional spaces efficiently.
Path Smoothness & Optimality	Paths are often rough (e.g., RRT), need smoothing; prone to local optima.	Smooth, near-optimal paths directly optimized in continuous spaces.
Real-Time Decision Making	Struggles with sudden changes, relies on static rules (e.g., APF).	Quick adaptation to environmental changes, real-time adjustments (e.g., PPO).

5 Conclusion

This study addresses both deep reinforcement learning-based path planning approaches and conventional path planning methods. Conventional techniques like Dijkstra and A* work well in static settings but are inefficient and computationally difficult in dynamic ones. Although, the DRL approach is able to learn optimization rules and enough to deal with complex and dynamic scenarios on its own. However, DRL algorithms such as DQN, PPO and DDPG are more suited to high-dimensional, dynamic environments. These are especially suited to real-time applications like autonomous driving and UAV navigation. However, training time and hardware overheads along with generalizability are still an issue with DRL.

Deep Reinforcement Learning has enormous potential for further improvement in the realm of route planning, and its future expansion will most likely focus on at least four key areas. The initial line of study is to enable efficient training approaches (e.g., distributed computing and model compression) as the task's complexity and the size of the environment increase, in order to reduce the amount of time and computer resources necessary for training. Second, the emphasis shifts to the DRL model's generalizability. In the future, the use of transfer learning, meta-learning, and multi-task learning approaches will increase the model's robustness and adaptability in predicted scenarios. Finally, combining this with additional solvers like (A* or Dijkstra) to blend reinforcement learning and path planning algorithms would maximize real-time path planning in dynamic contexts. Finally, with the fast advancement of autonomous driving and intelligent robotics systems, there will be a broader and more efficient application of DRL-based works to real-world situations in the near future, where stability in harsh conditions and robustness to sounds are experienced.

References

1 S. M. LaValle, Planning Algorithms. (Cambridge University Press, London, 2006)
2 P. E Hart, N. J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, **4**, 100 (1968)
3 D. Silver, Mastering the game of Go with deep neural networks and tree search. Nature, **529**, 484 (2016)

4 T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning. ArXiv preprint ArXiv:1509.02971, (2015)
5 V. Mnih, Human-level control through deep reinforcement learning. Nature, **518**, 529 (2015).
6 H. Kretzschmar, Socially compliant mobile robot navigation via inverse reinforcement learning. The International Journal of Robotics Research, **35**, 1289 (2016)
7 J. Schulman, Trust region policy optimization. Proceedings of the 32nd International Conference on Machine Learning, 1889 (2015)
8 L. Pinto, A. Gupta, Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), (2015) pp. 3406-3413
9 C. Finn, Model-agnostic meta-learning for fast adaptation of deep networks. ArXiv preprint arXiv:1703.03400, (2017)
10 E. W. Dijkstra, A note on two problems in connexion with graphs. Numerische Mathematik, **1**, 269 (1959)
11 V. Lumelsky, A. Stepanov, Dynamic path planning for a mobile automaton with limited information on the environment. IEEE Transactions on Automatic Control, **31**, 1058 (1987)
12 Lozano-Perez, T. Spatial planning: A configuration space approach. IEEE Transactions on Computers, **100**, 108 (1983)
13 Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. The International Journal of Robotics Research, **5**, 90 (1986)
14 S. S Ge, Y. J. Cui, Dynamic motion planning for mobile robots using potential field method. Autonomous Robots, **13**, 207 (2002)
15 S. M. LaValle, J. J. Kuffner, Randomized kinodynamic planning, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), (2001)
16 L. E. Kavraki, P. Svestka, J. C. Latombe, M. H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation, **12**, 566 (1996)
17 Y Zhu, Target-driven visual navigation in indoor scenes using deep reinforcement learning. IEEE International Conference on Robotics and Automation (ICRA), (2017) pp. 3357-3364
18 P Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, Learning to navigate in complex environments. arXiv preprint arXiv:1611. (2016).
19 J. Schulman, Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347. (2017)
20 B. Wang, End-to-End Autonomous Driving with Deep Reinforcement Learning in Simulation Environments, Technical University of Dresden, (2024)
21 C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680. (2019)
22 S. Gu, Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, 2017 IEEE International Conference on Robotics and Automation (ICRA), (2016) pp. 3389-3396
23 D. Zhang, Z. Xuan, J. Yao, X. Li, Path planning of unmanned aerial vehicle in complex environments based on state-detection twin delayed deep deterministic policy gradient. Machines, **11**, 108. (2023)
24 D. Brunori, DAMIAN: A delay-aware multI-aerial navigation environment for cooperative DRL-based UAV systems. PhD Thesis, Università di Roma La Sapienza. (2024)

25 Z. Wang, W. Gao, G. Li, Z Wang, Path Planning for Unmanned Aerial Vehicle via Off-Policy Reinforcement Learning with Enhanced Exploration. IEEE Transactions on Intelligent Transportation Systems. June, June (2024), pp. 2625-2639
26 Z. Chu, Y. Wang, D. Zhu, Local 2-D Path Planning of Unmanned Underwater Vehicles in Continuous Action Space Based on the Twin-Delayed Deep Deterministic Policy Gradient. IEEE Transactions on Systems, Man, and Cybernetics: Systems. May , May (2024), pp. 2775-2785
27 C. Chronis, G. Anagnostopoulos, E. Politi, Dynamic navigation in unconstrained environments using reinforcement learning algorithms. IEEE Transactions on Vehicular Technology, **72**, 102 (2023)
28 P. Li, D. Chen, Y. Wang, L. Zhang, S. Zhao, Path Planning of Mobile Robot Based on Improved TD3 Algorithm in Dynamic Environment. Heliyon, **10**, e2405 (2024).
29 M. Quinones-Ramirez, J. Rios-Martinez, Robot path planning using deep reinforcement learning. arXiv preprint arXiv:2302.09120. (2023)
30 Y. Zhang, W. Zhao, J. Wang, Y. Yuan, Recent progress, challenges and future prospects of applied deep reinforcement learning: A practical perspective in path planning. Neurocomputing. **608**,128423 (2024)
31 Y. Xiaofei, S. Yilun, L. Wei, Y. Hui, Z. Weibo, Global path planning algorithm based on double DQN for multi-tasks amphibious unmanned surface vehicle. Ocean Engineering, **244**, 112345 (2022)