2023 35th Chinese Control and Decision Conference (CCDC) | 979-8-3503-3472-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/CCDC58219.2023.10327188

A Novel Route Planning Approach Based on Energy-Based Action Sample Reinforcement Learning

Yuxiang Li
Department of Automation
Shanghai Jiao Tong University
Shanghai, China
ct_yuxiangli@sjtu.edu.cn

Jun-Guo Lu
Department of Automation
Shanghai Jiao Tong University
Shanghai, China
Corresponding auther:
jglu@sjtu.edu.cn

Zhen Zhu
Department of Automation
Shanghai Jiao Tong University
Shanghai, China
dyzz0928@sjtu.edu.cn

Qing-Hao Zhang
Department of Automation
Shanghai Jiao Tong University
Shanghai, China
zhangqinghao@sjtu.edu.cn

***Abstract*—Route planning for driverless vehicles in given environment has been widely studied. In this paper, a novel route path planning approach is proposed for self-driving based on energy-based action sample reinforcement learning. Firstly, based on the established graph, Q-learning is adopted in agent training with an novel considering steering penalty into reward function. Secondly, an innovative energy-based action sampling method is used to choose the most valuable action. Thirdly, the agent interacts with environment to plan a global path from the start to the goal point. As the results shown in experiments, this approach can complete the route planning task, and perform better in path length and turns (i.e. evaluation indicators).**

***Index Terms*—Reinforcement learning, self-driving vehicle, deep Q network, energy-based sampling.**

I. Introduction

Driverless vehicles are the most commendable evolution in transportation field [1]. For the sake of the tremendous progress made in computer vision technology and on-board sensors, driverless vehicles have the possibility to be driven on the road. An driverless driving system includes detection, localization, planning, decision-making and control. In [2], a survey of motion planning and control techniques, traditional route planning approaches, such as geometry methods, sampling-based methods and graph search strategies, are summaried comprehensively. They can complete the route planning tasks from the starting point to the destination in a established graph. However, in some extreme scenarios, these methods are prone to planning failure or huge computation.

Best first search (BFS) [3] is a heuristic path planning algorithm which find the closest path to the goal node. But the path BFS finded is probablely not the shortest. Dijkstra algorithm [4] is proposed which is a extraordinary method that selects node closest to the last choozen node. As Dijkstra focused on the shortest path length from the source point to all other points, it costs more computing resources than A* [5] which is depth first heuristic search algorithm, mainly calculating the shortest direct distance of nodes. The rapidly-exploring random trees (RRT) is an algorithm for expanding search through sampling in a fully known environment. it plays an essential role in the planning of multi-degree of freedom robots [6]. However, autonomous robots with dynamic constraints is hard to followed the RRT paths.

Traditional methods for resolving the route planning have its advantage in calculating speed, which is benefit to the little constraints of the task. But when the scenery becomes complex or the limits occur, these algorithms are doomed to huge computation or failing to plan a path.

Owing to the brisk development of computer processing efficiency, reinforcement learning has ushered in its spring [7]. It is widely used in various dynamic interactive environments. The self-driving route planning task with a constructed map is highly compatible with the applicable environment of reinforcement learning. There are also many researchers who use reinforcement learning to implement route planning while there are also a few down-to-earth problems [8] [9] [10] [11] [12]. Paths that all the above works planned are prone to not shortest, that is a lot of space for optimization. None of them take steering penalty into consideration, which is the vital element in factual driving scenery.

In [8], a algorithm was proposed that made it become a reality that using Q-network to train a critic to assist the agent to find a rational path to the goal in grid map. They adopted the strategy of experience replaying. The input of Q-network is random sampling in replay buffer. However, the maps that they used are too small and certainty. In [9], a algorithm was utilized accepted-rejected sampling to build its graph. The Q status was got by k-nearest search algorithm to train model. But it did not take avoiding obstacles into consideration. In [10], a model was trained by sampling expert demonstrations to accelerate convergence of algorithm. Nevertheless, the diversity of agent training data cannot be guaranteed. In [13], entropy was used to measure the status of sampled data. For instance, data buffer tends to update model parameters with a batch of data with larger variance rather than smaller one. In [11], reinforcement learning route planning has been completed on improved DQN algorithm that greatly reduce convergence time.

All these learning-based methods mentioned above have succeed in completing the task of realizing route planning in reinforcement learning, but few of them have considered the steering of vehicles.Besides, applying ε-greedy strategy in [8] [9] [10] [11] [14] [15], grows the risk of enrolling into local optim.

The motivation of this paper is to figure out these two problems: One is traditional path planning methods might fail to get a path. The other is the existing learning-based method might not consider steering penalty and using ε-greedy strategy is easy to enroll into local optim.

This paper is organized as follows: In Sec. II, the route planning problem is described including predefined symbols and preliminaries. In Sec III, the novel route path planning approach is fully described. Moreover, experiments and results are shown in Sec. IV. Besides, the analyse of results is displayed by charts and images. Last but not least, the conclusions of this paper are given in Sec V.

II. Problem Description

A. Environment Description

Considering a 4-direction grid map with obstacles, the white pixels are traversable region while the black are obstacles. A pair of start point and end point is random given in the grid map. The agent initial setting in the start point aims to find a shortest and smoothest path to the end point. A field of grid map with the position of agent and its travelled path are defined as environment state s_t at time moment t. Based on the state, the agent will take a certain action a. Synchronously, the environment will change to next state s_{t+1}.

A reward r is given by environment based on its own rule. The aciton pace of agent π is $\mathcal{A} = \{up,\ down,\ left,\ right\}$ (i.e. $a \in \mathcal{A}$). When the agent interacts with the environment achieving a certain number of steps or it has reached the goal, an episode comes to an end. The sum of all the rewards in an episode is the total reward, which is the maximum goal for the task. An action sequence in an episode is called Markov Decision Process (MDP) [16], with a property that the next state of environment is only depended on the current state.

B. Task Description

The task is defined as that an agent need to find a path P that is adjacent traversable from given start position s to the given goal position g, $P = P(s, g) = (s, next(s), next(next(s)), \ldots, g)$. Here $next(x)$ refers to the position that can be reached one step from position x. In this work, we expect the path that agent found is shorter and less turns, which closes to the real driverless scene.

The position of the agent at time t as $p_t = (x_t, y_t)$ is denoted. When an certainty action $a \in \mathcal{A}$ has been done, the influence of action is changing the position of agent: $p_t \rightarrow p_{t+1}$. If a collision happens, the status can not be changed, but a negetive reward is given to agent.

When agent reaches the goal or has operated reaching max steps, an episode is done. Unlike the game "highway-v0" in openai gym [17] ending the episode by collision occurring. This strategy can guarantee the variance and diversity of data generated by agent.

C. Q-learning and Deep Q Network

Q-learing is a value based algorithm of the reinforcement learning. It provides agents with the capability of learning to act optimally in Markovian domains by experiencing the consequences of actions, without requiring them to build maps of the domains [18].

Given π a policy of a task, if an agent observes the environment s, the $\pi(a|s)$ represents the probability of taking action a. The true value of an action a in state s is shown in eq (1):

$$Q_\pi(s, a) \equiv E[r_1 + \gamma r_2 + \ldots | \pi(s|a)]. \tag{1}$$

where $\gamma \in [0, 1]$ is a discount factor which preferred immediate rewards to later rewards. The optimal value is then $Q^*(s, a) = \underset{\pi}{max}\, Q_\pi(s, a)$. It is easy to learn a optimal policy by choosing the highest-value action in each state. The gradient descent algorithm is, well-known, temporary difference algorithm. Most interesting problems are too large to learn all action values in all states separately. Instead, we can learn parameterized value function $Q(s, a; \theta_t)$. Generally, Q-learning update the parameters after taking action a_t in state s_t and observing the immediate reward s_{t+1} and resulting state s_{t+1} is then in eq (2):

$$\theta_{t+1} = \theta_t + \alpha(Y_t^Q - Q(S_t, A_t; \theta_t))\nabla_{\theta_t} Q(S_t, A_t; \theta_t). \tag{2}$$

where α is a scalar step size and the target Y_t^Q is defined as eq (3):

$$Y_t^Q \equiv R_{t+1} + \gamma \underset{\theta}{max}\, Q(S_{t+1}, a; \theta_t). \tag{3}$$

This update resembles stochastic gradient descent, updating the current value $Q(S_t, A_t; \theta_t)$ towards a target value Y_t^Q [19].

A deep Q network (DQN) is a neural network with several layers which the input is the environment state s and the output is a vector containing action values $Q(s, \cdot; \theta)$, where θ are the parameters of neural network. In 2016, [19] propose the target network and experience replay. These two method have great increased the efficience of training process. The target used by DQN is shown by eq (4):

$$Y_t^{DQN} \equiv R_{t+1} + \gamma \underset{a}{max}\, Q(S_{t+1}, a; \theta_t^-), \tag{4}$$

where θ_t^- represents target network.

III. Main Works

A. Reward Function with Steering Penalty

Each step agent take will get reward is as follows in eq (5):

$$r_t = \begin{cases} r_{goal} & , p_t \rightarrow g \\ r_{colli} & , p_t \rightarrow 0 \\ r_{dist} & , p_t \rightarrow 1 \\ r_{turn} & , a_t \neq a_{t+1}, \end{cases} \tag{5}$$

where r_t refers to the reward agent obtained at time t. The agent obtains reward from the following rules. As it reaches

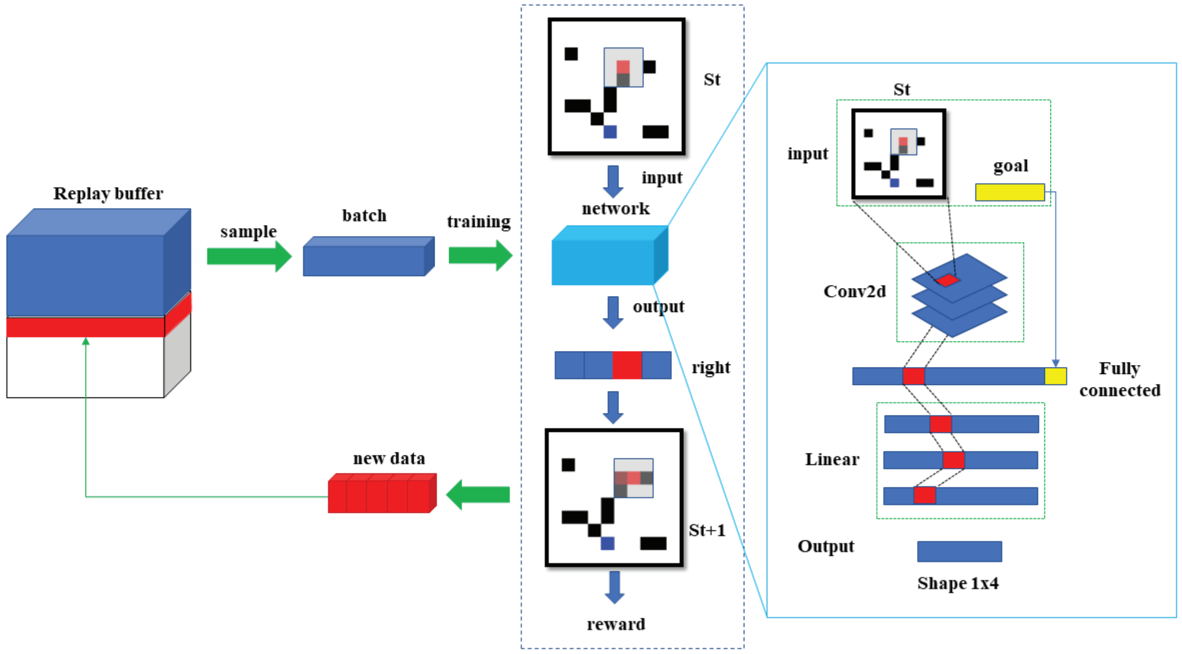

Fig. 1. Main pipline of interating with environment and training.

the goal, it will gain a large reward r_{goal}. Moreover, if the collision happens, a punishment r_{colli} (a negetive scalar reward) is returned by the environment. Last, when it reaches a position neither obstcles nor goal, a reward r_{dist} depends on distance and number of changing of direction is given by environment.

Compared with r_{colli} and r_{dist}, r_{goal} is about ten times of them. The reward value setting in this way is to prevent the reward value of repeated back and forth movement near the goal from being greater than that of direct movement to the goal. r_{colli} is also larger than r_{dist}, as the collision is not expected.

When agent reaches the goal or has operated reaching max steps, an episode is done. Unlike the game "highway-v0" in openai gym [17], it ends the episode by collision occurring. This strategy can guarantee the variance and diversity of data.

B. Main Pipline

The main pipline of our work is shown by Fig 1. An simulation environment via grid map including multi-size is created. As is shown in Fig 1. S_t, the red pixel is the start point while the blue is the goal point. The pixels centered on red grid are limited perception range of the vehicle. When a status of environment S_t as an input to the Q-network, Q-network's output is an 1×4 vector represents Q value (the expectation value of taking corresponding actions when the task come to an end). Then the max value's action will be operated to get the status S_{t+1}. At the same time, the environment will return a reward according to reward function eq (5). Each time that agent acts with environment is be record as a data $d = \{S_t,\ a_t,\ r_t,\ S_{t+1},\ end\ flag\}$. The data newly generated is added to the replay buffer $D = \{d_1, d_2, \ldots, d_n\}$, where n is the max length of buffer.

When training, a batch of datas is sampled randomly according to the method in [20]. These datas as the input of Q-network to update its parameters. We update the parameters a certain times after acting each episode. The loss function of the training is shown in eq (6):

$$Loss(\theta_i) = E_{s,a\sim\rho(\cdot)}[(y - Q(s,a;\theta_i))^2], \tag{6}$$

where y_i is defined in eq (7):

$$y = r_i + \gamma \max_a Q(S_{t+1}, a; \theta_{i-1}). \tag{7}$$

The structure of Q-network was inspired by [8]. a few improvement has made into the structure. The input of the Q-network is the Field of View (FOV) according to the current position of agent and goal. The FOV processes 3 convolution layers with $kernel\ size = 3{\times}3$, $padding = 1$, $stride = 1$ with activate function ReLu. Then flatten the tensor and connect it with goal tensor. After that, the 1-dimention tensor go through 3 Linear layers. The first layer's activate function is ReLu, but the second is Tanh. The output of Q-network is an 1×4 vector, the values in the vector represent Q values corresponding to each action.

C. Agent's Energy-based Action Sampling Method

For the purpose of balance between with exploitation and exploration, the $\varepsilon - greedy$ strategy is implemented in the baseline [21] experiment. A novel energy-based method is proposed in this work: Firstly, when an agent is in a position, those actions which might cause collision are discarded. Secondly, after an $softmax$ function [22] shown in eq (8):

$$softmax(\boldsymbol{x}) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}, \tag{8}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$. the output of network is transfered into probability corresponding to each action. Thirdly, roulette-wheel algorithm [23] is used to choose the action via these probabilities. This method greatly prevents the agent from reaching an obstacles and the edge of environment, as a result, it can greatly raise the convergence rate and save computation resources. The details of algorithm is shown in **Algorithm I**.

Algorithm 1 Energy-based DQN for Route Planning Algorithm

```
1:  Initialize parameters;
2:  Generate map with start point and end point;
3:  Generate intial data on map;
4:  for $episode = 1$ to $maxepisode$ do
5:      Reset the environment
6:      while $true$ do
7:          $step+ = 1$;
8:          Select action based on novel energy-based sampling
            strategy;
9:          Add $d = \{S_t,\ a_t,\ r_t,\ S_{t+1},\ end\ flag\}$ to replay
            buffer;
10:         if $agent\ locate \neq end\ point$ then
11:             $Q(s,a) = r + \gamma \max_a Q(s,a)$
12:         else
13:             $Q(s,a) = r$
14:             $done = true$
15:         end if
16:         if $done$ then
17:             step=0;
18:             Break;
19:         end if
20:     end while
21:     for $i = 0$ to $4$ do
22:         Sample batch data from replay buffer;
23:         Q network reverse retransmission update parameters;
24:     end for
25: end for
```

IV. Experiments

A. Evaluation Indicators

As for real task that driverless vehicles plan a path, the length and smooth should be taken into consideration. We are aim to train a model that is good at find a shorter and smoother path from the start point to goal. The evaluation indicators are divided into two parts: path length and turns in eq (9).

$$score_{total} = score_{length} + score_{turns} \tag{9}$$

where $score_{length}$ is the number of steps that agent operated in an episode game, and $score_{turns}$ represents the times of it changing its action.

B. Training details

We prepare multi-size of grid maps including 10×10, 20×20, 50×50, with obstcles approximately 10% of the whole map. And given a start point and the goal for the agent. The maximum size of replay buffer is 1e6. Before training, 10 episode data will be add into the buffer. When an episode is done, a sequence of training data is append to the end of buffer until it is full (then the new data will replace the oldest's data). Concurrently, the parameters of model will be updated 5 times.

While the different map sizes, Various max steps of an episode and training batch sizes are defined. The relationship between map size, training batch and max step is shown in TABLE I. It is because the convergence speed is tremendous affected by max steps of an episode, which is various due to the size of map.

TABLE I
RELATIONSHIP BETWEEN MAP SIZE, MAX STEP AND BATCH SIZE

map size	*max step*	*batch size*
10×10	25	16
20×20	45	32
50×50	100	64
100×100	200	128

The reappearence of the work [8] has done in our experiment as baseline. The experiment is then divided into 2 steps. One is only applying the reward function which proposed in eq (5), called "steering-penalty" (SP). The another is added action sampling and steering penalty together, called "steering-penalty-action-sampling" (SPAS). All the 3 algorithms apply in 3 sizes of maps We compare the convergence speed of these tasks and the indicators present above between 3 algorithms. The results are displayed in next subsection.

C. Results

As shown in TABLE II, We created 10 different random maps and randomly selected start point and goal point. The data in TABLE II are the average of the results. The best data are bold in table. Compared the results of "baseline" (reappearence work in [8]) and SP, it is obvious that SP has a shorter path length and fewer turns than "baseline", which is more obvious when the map is larger. At the same time, compared with the effect of SPAS, the effect of SPAS is better

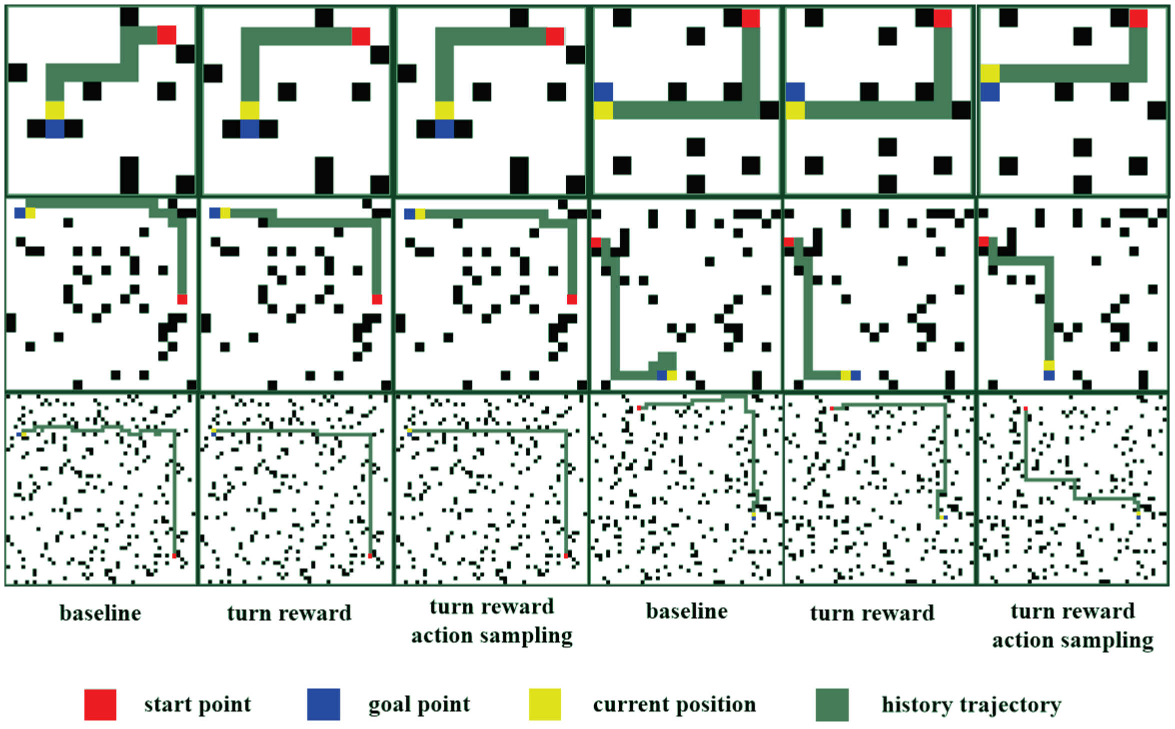

Fig. 2. results of route planning with different map sizes

TABLE II
RESULTS OF DIFFERENT MAP SIZES INDICATORS

map size	10 × 10			20 × 20			50 × 50		
indicators / **algorithms**	length	turns	score	length	turns	score	length	turns	score
baseline	12.6	3.2	15.8	28.8	7.4	36.2	69.2	16.4	85.6
SP	12.2	2.2	14.4	24	**2.6**	26.6	62	**5**	**67**
SPAS	**11.8**	**1.8**	**13.6**	**23.6**	2.8	**26.4**	**61.2**	6.6	67.8

than that of SP in terms of path length and turns, but it is slightly lower when the map is large.

A few of route planning paths are shown in Fig. 2, each line serves as diverse map sizes: 10×10, 20×20, 50×50. The black pixels are the obstcles while white are traversable area. Other legends are displayed in Fig. 2. There are 6 maps in Fig. 2. Each map is test by 3 algorithms mentioned above. There is no doubt that in all 3 scenes, path that SPAS model planned is almost shortest and smoothest. It can even find some very perfect paths to reach the destination, such as the sub graph of the third row and the third column in Fig. 2. Note that the ability of the model trained by SP algorithm is no less than that of SPAS algorithm in some large scene maps.

V. Conclusions

In this work, the learning-based algorithm for self-driving vehicles route path planning has been proposed. Our approach has succeeded in completing the route planning task in given environment. The path length and agent steering times of the last two groups are less than the first one, which has demonstrate the following conclusions:

- Considering steering penalty does well in vehicle turns in route planning task.
- Using energy-based sample strategy in action decision make great progress in shorter path length.

- The algorithm proposed can succeed in more large and more complex scene.
- providing information of start point and goal to agent (i.e. take start point and goal as an input of network) does agent a favor to find the path to the end point.

Later, we will apply this algorithm to more complex and practical scenes. When constructing vehicle model, it is constraints of vehicle dynamic and kinematics that are not ignored. Then, taking the global path that the algorithm planned as the initial following trajectory of self-driving vehicles is the greatest value of this algorithm. On the other hand, this work also makes the connection between reinforcement learning and automatic driving closer, and contributes to the process of application of reinforcement learning in unmanned driving technology.

REFERENCES

[1] P. Liu, R. Yang, and Z. Xu, "How safe is safe enough for self-driving vehicles?" *Risk analysis*, vol. 39, no. 2, pp. 315–325, 2019.

[2] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[3] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of a," *Journal of the ACM (JACM)*, vol. 32, no. 3, pp. 505–536, 1985.

[4] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, 2022, pp. 287–290.

[5] G. Nannicini, D. Delling, L. Liberti, and D. Schultes, "Bidirectional a∗ search for time-dependent fast paths," in *International Workshop on Experimental and Efficient Algorithms*. Springer, 2008, pp. 334–346.

[6] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[7] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[8] A. I. Panov, K. S. Yakovlev, and R. Suvorov, "Grid path planning with deep reinforcement learning: Preliminary results," *Procedia computer science*, vol. 123, pp. 347–353, 2018.

[9] P. Gao, Z. Liu, Z. Wu, and D. Wang, "A global path planning algorithm for robots using reinforcement learning," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 1693–1698.

[10] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, "Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4423–4430, 2018.

[11] Y. Yang, L. Juntao, and P. Lingling, "Multi-robot path planning based on a deep reinforcement learning dqn algorithm," *CAAI Transactions on Intelligence Technology*, vol. 5, no. 3, pp. 177–183, 2020.

[12] H. Sang, Y. You, X. Sun, Y. Zhou, and F. Liu, "The hybrid path planning algorithm based on improved a* and artificial potential field for unmanned surface vehicle formations," *Ocean Engineering*, vol. 223, p. 108709, 2021.

[13] K. Lee, L. Smith, and P. Abbeel, "Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training," *arXiv preprint arXiv:2106.05091*, 2021.

[14] I. Sung, B. Choi, and P. Nielsen, "On the training of a neural network for online path planning with offline path planning algorithms," *International Journal of Information Management*, vol. 57, p. 102142, 2021.

[15] J. Li, Y. Chen, X. Zhao, and J. Huang, "An improved dqn path planning algorithm," *The Journal of Supercomputing*, vol. 78, no. 1, pp. 616–639, 2022.

[16] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[17] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[18] P. Dayan and C. Watkins, "Q-learning," *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.

[19] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[20] X. Qiu, Z. Yao, F. Tan, Z. Zhu, and J.-G. Lu, "One-to-one air-combat maneuver strategy based on improved td3 algorithm," in *2020 Chinese Automation Congress (CAC)*. IEEE, 2020, pp. 5719–5725.

[21] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. S. Kumar, S. Koenig, and H. Choset, "Primal: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2378–2385, 2019.

[22] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[23] A. Lipowski and D. Lipowska, "Roulette-wheel selection via stochastic acceptance," *Physica A: Statistical Mechanics and its Applications*, vol. 391, no. 6, pp. 2193–2196, 2012.