electronics

MDPI

Article

Deep Reinforcement Learning-Based 2.5D Multi-Objective Path Planning for Ground Vehicles: Considering Distance and Energy Consumption

Xiru Wu *,†, Shuqiao Huang † and Guoming Huang *

School of Electronic Engineering and Automation, Guilin University of Electronic Technology, Guilin 541004, China; 21082304036@mails.guet.edu.cn

* Correspondence: xiruwu@guet.edu.cn (X.W.); huangguoming@guet.edu.cn (G.H.)

† These authors contributed equally to this work.

Abstract: Due to the vastly different energy consumption between up-slope and down-slope, a path with the shortest length in a complex off-road terrain environment (2.5D map) is not always the path with the least energy consumption. For any energy-sensitive vehicle, realizing a good trade-off between distance and energy consumption in 2.5D path planning is significantly meaningful. In this paper, we propose a deep reinforcement learning-based 2.5D multi-objective path planning method (DMOP). The DMOP can efficiently find the desired path in three steps: (1) transform the high-resolution 2.5D map into a small-size map, (2) use a trained deep Q network (DQN) to find the desired path on the small-size map, and (3) build the planned path to the original high-resolution map using a path-enhanced method. In addition, the hybrid exploration strategy and reward-shaping theory are applied to train the DQN. The reward function is constructed with the information of terrain, distance, and border. The simulation results show that the proposed method can finish the multi-objective 2.5D path planning task with significantly high efficiency and quality. Also, simulations prove that the method has powerful reasoning capability that enables it to perform arbitrary untrained planning tasks.

Keywords: 2.5D path planning; deep reinforcement learning; deep Q learning; reward shaping; ground vehicles

Citation: Wu , X.; Huang, S.; Huang, G. Deep Reinforcement Learning-Based 2.5D Multi-Objective Path Planning for Ground Vehicles: Considering Distance and Energy Consumption. *Electronics* **2023**, *12*, 3840. https://doi.org/10.3390/electronics12183840

Academic Editors: Stéphane Caro, Guanglei Wu, Ming Shen and Ricardo Soto

Received: 13 August 2023
Revised: 7 September 2023
Accepted: 8 September 2023
Published: 11 September 2023

1. Introduction

The use of energy is a important topic of interest in the field of ground vehicles (GVs) [1]. GVs such as exploration robots and space probes mostly operate in the complex off-road terrain environment (2.5D environment), and energy consumption can become a problem that restricts their working range and time. So far, scholars have studied battery technology [2], motor control technology [3], and gas engine technology [4] in order to improve energy efficiency.

In this paper, an energy-saving method from the perspective of path planning for GVs is studied. A GV going uphill always consumes a lot of energy to overcome its gravity, which means a path with less up-slope but a bit longer may save more energy than that of the shortest path, which covers many up-slopes. Hence, there will always be some energy-saving paths on the 2.5D map. Meanwhile, some of these paths with longer distances may significantly increase the traveling time, which is also unacceptable. Overall, this yields a multi-objective path planning problem: to find paths on a terrain environment with a good trade-off between energy consumption and distance.

The traditional methods such as violent search [5], heuristic-based search [6] and probabilistic search [7], etc., can solve this multi-objective path planning problem to some extent. However, all of them encounter some shortcomings on this task. The violent search methods have to search a large area with low efficiency. The heuristic-based methods

suffer the difficulty of modeling the heuristic function for 2.5D path planning (it is hard to estimate the cost of a path on the terrain map). The probabilistic search methods show unstable performance in the off-road environment.

Lately, deep reinforcement learning (DRL) has emerged as a highly promising approach for tackling path planning problems [8]. Therefore, in this paper, we employ DRL to establish a multi-objective 2.5D path planning method (DMOP) that focuses on improving planning efficiency while ensuring a good balance between energy consumption and the distance of the planned path. The contribution of this work is listed as below:

- Novel topic: A deep reinforcement learning-based multi-objective 2.5D path planning method (DMOP) is proposed for GVs, realizing a good trade-off on distance and energy consumption on off-road terrain environments.
- Useful findings: (1) The DMOP plans paths with significant high efficiency in comparison with other methods. (2) The DMOP has powerful reasoning capability that enables it to perform arbitrary untrained planning tasks.
- Solved two technical problems: (1) The convergence problem of DQN in high state space is solved by using a hybrid exploration strategy and reward-shaping theory. (2) To tackle the multi-objective 2.5D path planning problem using DRL, an exquisite reward function is designed, incorporating terrain, distance, and border information for modeling.

The remainder of this paper is structured as follows: Section 2 provides a review of the current research status on 2.5D path planning algorithms. The preliminaries of the 2.5D path planning task are given in Section 3. Then, the detailed description of the DMOP is introduced in Section 4. Section 5 describes the experiments conducted to verify the effectiveness of DMOP and exhibits and analyzes the experimental results. Finally, Section 6 summarizes this paper and outlines future research directions.

2. Related Works

Path planning methods that consider slope (on 2.5D map) have been studied for years, and researchers are concentrating on finding suitable paths for GVs in complex terrain environments [9–11]. In early research, most studies focused on single-object optimization problems, such as traversability or avoiding obstacles. Therefore, scholars are concerned about vehicle dynamics to ensure motion stability [7,12,13]. For instance, a probabilistic road map (PRM)-based path planning method considering traversability is proposed for extreme-terrain rappelling rovers [7]. Linhui et al. [12] present an obstacle avoidance path-planning algorithm based on a stereo vision system for intelligent vehicles; such an algorithm uses stereo matching techniques to recognize obstacles. Shin et al. [13] propose a path planning method for autonomous ground vehicles (AGVs) operating in rough terrain dynamic environments, employing a passivity-based model predictive control. Obviously, these planning methods place less emphasis on the global nature of the path.

As research on 2.5D path planning deepens, scholars are prone to create multi-objective path planning methods that take complex characteristics of 2.5D terrain into account, especially the issues of operational efficiency, energy consumption, and the difficulty of mechanical operation. These works aim to find paths that are energy efficient and easy to pass [14,15]. Ma et al. [16] propose an energy-efficient path planning method built upon an enhanced A* algorithm and a particle swarm algorithm for ground robots in complex outdoor scenarios. This method considers the slope parameter and friction coefficient, which can effectively assist ground robots in traversing areas with various terrain features and landcover types. Based on the hybrid A* method, Ganganath et al. [17] present a algorithm by incorporating a heuristic energy-cost function, which enables the robots to find energy-efficient paths in steep terrains. Even though A*-based methods are likely to find the optimal or near-optimal path, their performance in terms of search time tends to be poor due to the search mechanism. Moreover, Jiang et al. [18] investigate a reliable and robust rapidly exploring random tree (R2-RRT*) algorithm for the multi-objective planning mission of AGVs under uncertain terrain environment. However, this kind of method has an unavoidable weaknesses, local minimum, which means it cannot guarantee the quality

of the path. Despite the fact that existing research has made notable achievements, the 2.5D path planning method considering energy consumption and the distance of the global path has not been widely studied.

Regarding this topic, our team has previously proposed a heuristic-based method (H3DM) [19]. In this method, a novel weight decay model is introduced to solve the modeling problem of the heuristic function, and then the path planning with a better trade-off of distance and energy consumption is realized. Nonetheless, the solutions obtained by this method are not good enough regarding the path's global nature and the search efficiency. Thus, the multi-objective path planning method, which considers the path's global nature and time efficiency, should be further researched.

With the merit of solving complex programming problems, DRL has achieved much success in Go [20], video games [21], internet recommendation systems [22], etc. Scholars have also proved that DRL has the potential to solve N-P hard problems [23], such as the traveling salesman problem (TSP) [24] and the postman problem [25]. Since the path planning problem is also a kind of N-P hard problem [26,27], researchers have begun to apply DRL theory to solve path planning problems. Ren et al. [28] create a two-stage DRL method, which enables the mobile robot to navigate from any initial point in the continuous workspace to a specific goal location along an optimal path. Based on the double deep Q network, Chu et al. [29] propose a path planning method for autonomous underwater vehicles under unknown environments with disturbances from ocean currents. These methods have powerful reasoning capabilities and show the great potential of DRL in the field of path planning.

In order to develop a computationally efficient method with a better trade-off of distance and energy consumption, we present the DMOP method using the DRL theory in this paper. The DMOP is composed of a deep Q network (DQN), which can finish the planning task with higher efficiency in comparison with other methods.

3. Preliminaries

3.1. Environment for Path Planning

The planning task is regarded as an Atari-like game, and we have built an environment, namely, "Mario Looks for Mushroom". The goal of this game is to control Mario to find the path from an initial position to the target where the mushroom is located. In this game, the controller is given eight actions corresponding to eight directions (see Figure 1). Then, this planning task is understood as making the agent learn to find the mushroom with less energy consumption and travel time. In addition, 2.5D off-road maps are used in this paper. All of them are digital elevation maps (DEMs) in which the Z-axis denotes the altitude and the XOY plane represents the horizontal plane. DEMs assume that each sampling pixel (x, y, z) can be reached in the planning process.

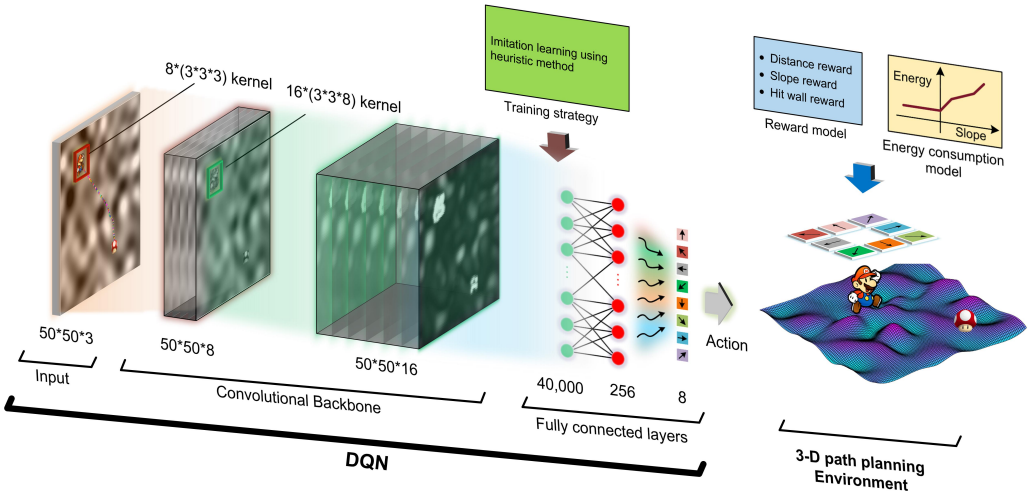

Figure 1. DQN for 2.5D path planning task.

3.2. Distance Calculation Model

In this paper, the distance D from point to point is calculated via the straight line distance equation,

$$D(P_n,P_{n+1}) = \sqrt{(x_n - x_{n+1})^2 + (y_n - y_{n+1})^2 + (z_n - z_{n+1})^2} \tag{1}$$

where (x_n, y_n, z_n) is the 3D coordinate of P_n and $(x_{n+1}, y_{n+1}, z_{n+1})$ is the 3D coordinate of P_{n+1}.

Hence, the distance of a specified path can be acquired as:

$$D_{total} = \sum_{i=1}^{d-1} D(P_i, P_{i+1}) \tag{2}$$

where d denotes the number of sampling points in a path.

3.3. Energy Consumption Calculation Model

Based on the above distance calculation model, the energy consumption between two points is modeled as:

$$E(P_n, P_{n+1}) = \begin{cases} D(P_n, P_{n+1}) \cdot E_{us}, z_n \leq z_{n+1} \\ D(P_n, P_{n+1}) \cdot E_{ds}, z_n \geq z_{n+1} \end{cases} \tag{3}$$

where E_{us} and E_{ds} are the up-slope and down-slope energy consumption estimation [19].

Then, these two variables can be modeled as:

$$E_{us} = \beta / \frac{\pi}{4} \cdot \eta_{us} \tag{4}$$

$$E_{ds} = 1 - \beta / \frac{\pi}{4} \cdot \eta_{ds} \tag{5}$$

where η_{us} and η_{ds} denote the up-slope and down-slope estimated energy consumption per meter when $\beta = 30^\circ$, respectively. β is calculated as:

$$\beta = \arctan\left(\frac{z_n - z_{n+1}}{\sqrt{(x_n - x_{n+1})^2 + (y_n - y_{n+1})^2}}\right) \tag{6}$$

where (x_n, y_n, z_n) and $(x_{n+1}, y_{n+1}, z_{n+1})$ are the co-ordinates of two reference point P_n and P_{n+1}. In addition, the unit of E_{us} and E_{ds} is set as u in this paper. The unit of u is Joule, and its value can be determined via test for a specific GV. Here, η_{us} and η_{ds} are set as 25 u/m and 0.25 u/m, respectively.

Based on this model, the energy consumption of a path is calculated as:

$$E_{total} = \sum_{i=1}^{d-1} E(P_i, P_{i+1}) \tag{7}$$

4. Dqn-Based Multi-Objective 2.5D Path Planning

In this section, we first present the problem statement, followed by the design of DQN, including the structure of the convolutional neural network (CNN), the modeling of the reward function, and the training strategy, as well as the path-enhanced method. The DMOP method built from these components ultimately solves the multi-objectives 2.5D path planning problem.

4.1. Problem Statement

Though the DQN has the great potential to solve the path planning problem, two technical problems should be solved here: (1) the convergence problem of DQN in high state space; (2) the multi-objective path planning problem.

- The convergence problem of DQN in high state space presents a challenge. The construction of the 2.5D map at a relatively high resolution results in a large map size, further exacerbating the issue as the DQN struggles to converge due to the extensive state space. Even for a size of 20*20, the DQN may fail to converge sometimes.
- The multi-objective path planning problem involves the goal of finding not only a path from a given start point to a target but also a globally optimal path with a short length and low energy consumption.

4.2. Background and Structure of DQN

DQN is a reinforcement learning technique that combines deep neural networks with Q-learning to tackle complex decision-making tasks [21]. It leverages deep neural networks to estimate the Q-values of different actions in various states [30].

As shown in Figure 1, the adopted DQN is a convolutional neural network that is composed of three parts: the input layer, the convolutional backbone, and the fully connected layers.

The input layer is the observation state of the DQN, while the state is a 50*50*3 image (transforming the 100*100*3 map into a 50*50*3 image using the pixel skipping technique) in which Mario, mushroom, and the 2.5D map are drawn on the different channels, respectively.

The convolutional backbone is built with 2 convolutional layers: the size of the convolutional kernel for the first layer is 8*(3*3*3), and the second is 16*(3*3*8). The activation function is ReLu. It should be mentioned that Mario and mushroom are used to replace the single pixel of the current and target point, to allow the network to more easily recognize the current location of agent and target, thus accelerating the learning process. In addition, no pooling layers are used here. According to numerous experiments, using the pooling layer in this structure can lead to a severe convergence problem. The reason may be that too much information is lost after the pooling process.

The fully connected layers consist of three layers. The first layer connects to the output of the last convolutional layer. The second one has 256 nodes; both this layer and the former layer also use ReLu as the activation function. The last layer has 8 nodes that represent the Q value of actions, and it takes a linear function as the activation function.

4.3. Design of Reward Function

The reward function plays a key role in training the DQN. When the size of the map becomes bigger and bigger, the DQN will be more difficult to converge. If the agent only gets the reward at the target, the DQN will hardly converge through random exploration, let alone realize the multi-objective planning. To solve this problem, an exquisite reward function is designed for the 2.5D multi-objective path planning task. Based on the reward-shaping theory [21], this function is built from the following points:

- The DQN aims to find a policy that maximizes the discounted accumulated reward (score), and the goal of this game is to find a path with a good trade-off between distance and energy consumption; hence, the reward for Mario's step should be negative (penalty), otherwise the DQN will tend to plan a path that maximizes the reward, even if it means making the path as long as possible.
- To guide Mario to find the mushroom, a suitable distance-based reward should be provided. In addition, the agent can receive a bigger reward as it gets closer to the target.
- Considering the energy consumption in path planning, a slope-based penalty should be given.

- To prevent the agent from cheating to obtain a higher score by hitting the wall, the hit-wall penalty should be relatively strong.
- A relatively big reward should be given to encourage the agent to remember the experience when reaching the target.

Based on the above ideas, the reward function is modeled as:

$$r = r_d + r_s + r_w + r_t \tag{8}$$

where r_d is the distance penalty, r_s is the slope penalty, r_w denotes the hit-wall penalty, and r_t stands for the reached reward.

4.3.1. Distance Penalty Model

The model of the distance penalty includes two parts:

$$r_d = r_{d1} + r_{d2} \tag{9}$$

where r_d is a global distance penalty, and r_{d1} is a step distance penalty that is modeled as:

$$r_{d1} = -D(P_n, P_{n+1})/D_{b1} \tag{10}$$

where $D(P_n, P_{n+1})$ is a step distance in 2.5D aspect, D_{b1} is a factor of the intensity, and $D_{b1} = 10$ according to a bunch of tests.

r_{d1} alone is not enough as it neglects the terrain information, so the second part r_{d2} is introduced to compensate for the penalty. r_{d2} heuristically penalizes the action that leads the next current position too far away from the destination in the 2D aspect. It is modeled as:

$$r_{d2} = -D_2(P_{n+1}, P_d)/D_{b2} \tag{11}$$

where $D_2(P_{n+1}, P_d)$ means the 2D distance between the next current position and the destination, and D_{b2} is an intensive factor (its value is set according to the size of the map). Here, $D_{b2} = 50$.

4.3.2. Slope Penalty Model

The slope penalty can be calculated based on Equation (3). Here, the slope penalty is defined as:

$$r_s = -E(P_{n-1}, P_n) \cdot k \tag{12}$$

where k is the intensity factor. k is set as 5 according to many training experiment. It is easy to understand that the penalty is proportional to energy consumption.

Here, a case is given to illustrate the slope penalty term. For a path depicted in Figure 2a, the up-slope motion consumes lots of energy (see Figure 2b). Moreover, we can control the intensity of the slope penalty with a different factor k (see Figure 2c). From Figure 2d, we can see that when k becomes bigger, the planned path is more sensitive to the slope and tries to avoid going up-slope. Ultimately, we choose k = 5 as it produces a balanced trade-off between energy consumption and distance in the planned paths.

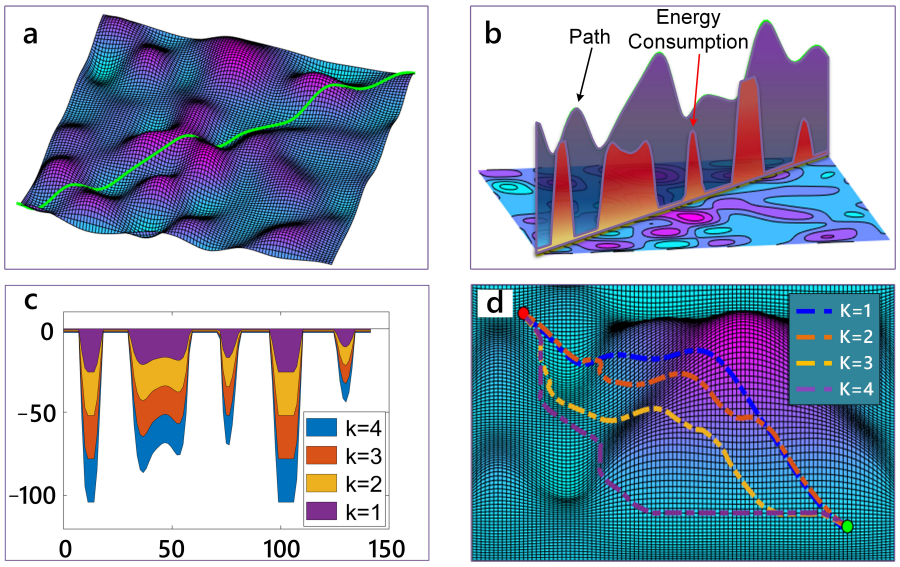

Figure 2. The function of slope penalty term. (**a**) depicts a path for demonstration. (**b**) shows the curves of energy consumption compared with the path. (**c**) gives the energy consumption with different factor k. (**d**) is the corresponding planned results.

4.3.3. Hit-Wall Penalty Model

The wall penalty term r_w is used to prevent the agent from hitting the boundary of the maps for the higher reward and thus from cheating in game. r_w is modeled as:

$$r_w = \min((r_{wx} + r_{wy}), b) \cdot \gamma; \tag{13}$$

where b is the upper bound, and $b = 6$. This parameter is set to prevent excessive penalties when the agent is positioned in the four corners of the map, namely, the top-left, bottom-left, top-right, and bottom-right corners. r_{wx} and r_{wy} are corresponding penalties to the x and y axes, respectively. γ denotes the penalty intensity, and $\gamma = 5$. r_{wx} and r_{wy} are defined in a similar way:

$$r_{wx} = \begin{cases} (l_1 - P_x) \cdot 2, P_x \leq l_1 \\ (P_x - l_2) \cdot 2, P_x \geq l_2 \\ 0, \quad others \end{cases} \tag{14}$$

$$r_{wy} = \begin{cases} (l_1 - P_y) \cdot 2, P_y \leq l_1 \\ (P_y - l_2) \cdot 2, P_y \geq l_2 \\ 0, \quad others \end{cases} \tag{15}$$

where l_1 and l_2 are the specific lines that trigger the wall penalty, and $l_1 = 3, l_2 = 97$. P_x and P_y are the corresponding coordinates on the XOY plate.

4.3.4. Reached Reward Model

The reached reward r_t is used to encourage the agent to remember the experiences it has encountered.

$$r_t = 5 \cdot k \tag{16}$$

where k is the same factor used in Equation (18). The coefficient "5" is tuned manually according to the training results.

4.4. Hybrid Exploration Strategy

Since the state space is too big, the DQN is difficult to converge through random exploration, even when a well-designed reward function is used. Here, a hybrid exploration strategy is applied to improve the training performance. We design a hybrid exploration strategy that uses the heuristic method to generate useful experience for training the DQN.

The traditional training strategy for DQN initially involves selecting actions with a certain probability, either through random exploration or using DQN's output, to guide the agent to successfully reach the target and gain valuable experiences for learning. However, in complex 2.5D path planning tasks, this method can lead to difficulties in reaching the target during the exploration process, resulting in a lack of successful experiences to learn from. To obtain more successful experiences and improve the learning efficiency, the exploration strategy has transitioned from pure random exploration to a blend of heuristic methods and random exploration. The heuristic method involves comparing the distances between each neighboring point and the target, with the action corresponding to the minimum distance being selected as the recommended action. In the early stages of training, there is a 60% probability of employing a hybrid exploration strategy, where 50% probability is assigned to selecting random actions and 10% probability is assigned to selecting heuristic actions. The remaining 40% probability is devoted to using actions generated by the DQN (see Figure 3). As the training goes further, the exploration probability will decrease to a relatively small value (the figure 60% will decrease to 1%, and 40% will increase to 99% at the end). Eventually, the agent acquires the knowledge of how to plan paths.

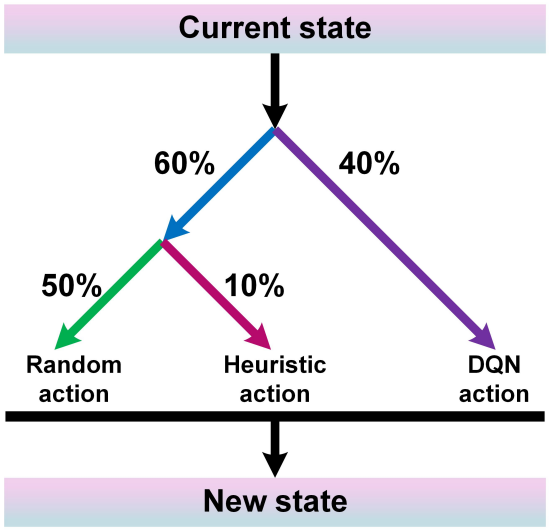

Figure 3. Action selection in exploration.

4.5. Path-Enhanced Method

The planned path on a lower resolution map includes many straight segments that should be curves on the higher resolution map; these segments can be recovered. Here, a path-enhancing method is proposed to transform a lower resolution path (50*50) into a higher one (100*100, or even higher).

The proposed method includes three steps: (1) map the planned path into the corresponding high resolution map, (2) extract the tuning points using the slope information of the path, and (3) connect the extracted points using the cubic spline interpolation method. Figure 4 shows an example of a smooth process using the path-enhanced method. It shows that the green curve becomes smoother than the red curve.

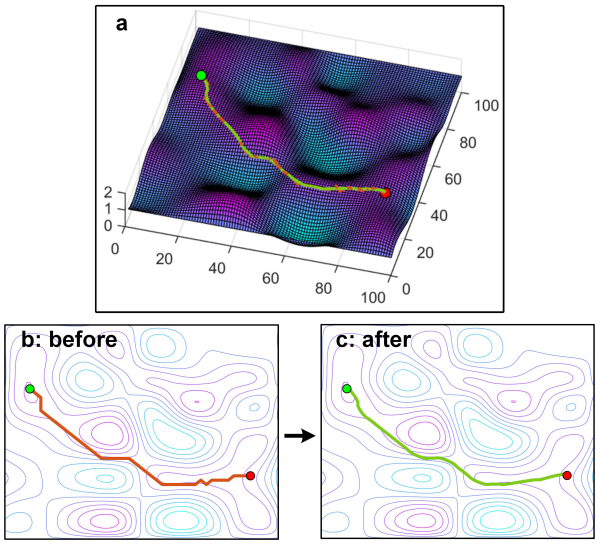

Figure 4. Example of path-enhanced. (**a**) is the comparison on the 2.5D perspective. (**b**,**c**) are, respectively, the original path and the processed path from the 2D perspective.

4.6. Hyperparameters Setting and Training Result

The DMOP method is trained with multitask, which means that the start point and the target are set randomly in different episodes. The hyperparameters are listed in Table 1.

Table 1. Hyperparameters.

Hyperparameters	Value
Learning rate	0.0001
Discount factor	0.99
Batch size	64
Batch number per training	20
Maximum episodes	6000
Maximum planning step per episode	150
Training interval	5
Initial probability of random action	0.5
Initial probability of heuristic action	0.1
Initial probability of DQN action	0.4
Energy factor	4

In Figure 5, the curves of reward in many cases are depicted. This result reveals that the convergence problem of DQN with high state space [21] can be solved via our method. When the DQN is converged, it learns how to recognize Mario and mushroom. In addition, we found an interesting phenomenon: that the bottom of valleys (see the yellow circuits in Figure 5b) is regarded as the peak by DQN, which means the agent takes as much energy to cross a valley as it does to cross a mountain.

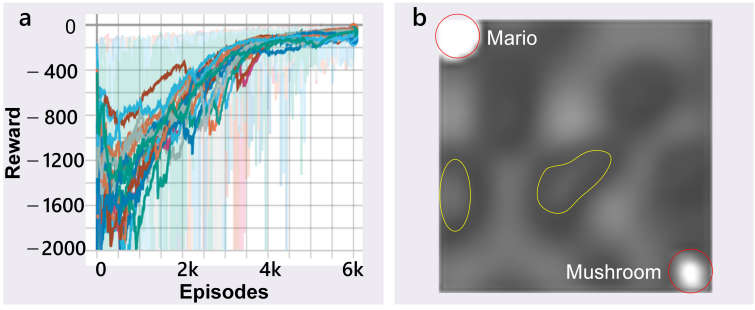

Figure 5. Result of training. (**a**) The curves of reward. (**b**) Feature map of CNN.

5. Simulation and Analysis

5.1. Setting and Results

The 2.5D map used in this section is set as: both of its x and y axes range in [1, 100], and its z-axis ranges in [0, 2]. This map can be regarded as it is zoomed from a 10 km * 10 km area with a maximum altitude of 200 m. Furthermore, the proposed method is compared with the H3DM method [19], an improved RRT method [31], the Dijkstra method [32], and the A^* method [33]. H3DM is a purely heuristic-based algorithm; at its core, it adds a distance weight decay model to the heuristic function to estimate energy consumption and distance. Based on the idea of H3DM, we apply the identical heuristic function model to the A* algorithm. The A* algorithm used in the 2.5D scenario employs a search mechanism the same as that in the 2D case. It evaluates and sorts nodes in the map based on the total cost function, selecting the node with the lowest cost to expand the search path. Dijkstra utilizes the same search mechanism as A*, with the only difference being that it lacks a heuristic function in its total cost function. As for the RRT algorithm, its search strategy is similar to that in 2D, and we enhance the guidance towards the target direction [34]. After planning the path, we calculate energy consumption data. All these methods are planning for multi-objective, where the energy consumption and path distance are using the same weights in comparison. All the methods run on a computer with NVIDIA GTX 3060 GPU, AMD Ryzen 5600 CPU, and 16G RAM, and the codes are implemented with python. It should be mentioned that the DQN in the DMOP method is trained with multitask, and the tested tasks have not all been used in the training process.

To compare the performance of the DMOP statistically, we conducted 200 path planning experiments with random starting points and destinations and then choose five representative cases to demonstrate the effectiveness of the method. The plots in Figure 6 illustrate the simulation result for each algorithm. In Figure 6a, the points represent the performance of various algorithms in terms of time, energy consumption, and path distance in the planning task. For ease of data visualization, we have taken the logarithm of time values, making the scatterplot distribution more concentrated. Furthermore, points closer to the lower-right corner indicate lower energy consumption, shorter path distance, and search time. In this three-dimensional graph, it is easy to observe that, overall, DMOP significantly reduces search time, exhibiting low energy consumption and short path length. In Figure 6b, the vertical axis represents the sum of path length and energy consumption, where a smaller value indicates higher path quality in the search. Due to its breadth-first search mechanism, Dijkstra's algorithm can find the global optimum path in this task. However, its search mechanism also results in much longer search time. In contrast, DMOP manages to significantly reduce search time while maintaining path quality similar to that of the Dijkstra's algorithm. From a large volume of numerical statistics, we can observe that the DMOP method demonstrates excellent stability and performance.

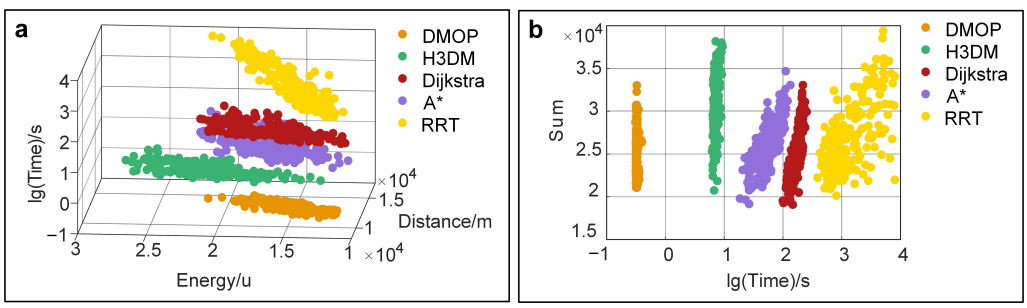

Figure 6. Numerical statistical graph. (**a**) Results show in three-dimensional graph. (**b**) Results show in two-dimensional graph.

The five cases are shown in Figures 7–11. Meanwhile, the quantified results are shown in Tables 2–6.

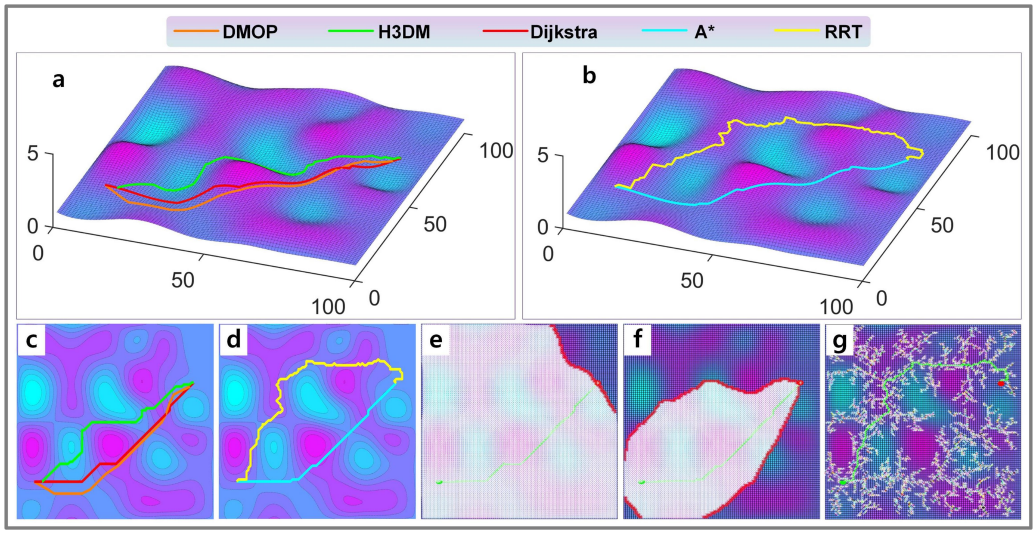

Figure 7. Case 1 of experiments. (**a**) Results of H3DM, Dijkstra and DMOP in 2.5D view. (**b**) Results of RRT and A* in 2.5D view. (**c**) Results of H3DM, Dijkstra and DMOP in top view. (**d**) Results of RRT and A* in top view. (**e**) Searching area of Dijkstra method. (**f**) Searching area of A* method. (**g**) Searching area of RRT method.

Table 2. Quantitative result of Case 1.

Method	Distance (m)	Energy (u)	Sum	Time (s)
DMOP	10,686.96	13,027.12	23,714.08	0.32
H3DM	10,956.94	13,832.42	24,789.36	10.49
Dijkstra	10,195.35	13,073.70	23,269.05	133.63
A*	10,195.08	13,076.01	23,271.09	53.28
RRT	15,337.99	18,897.25	34,235.25	2415.35

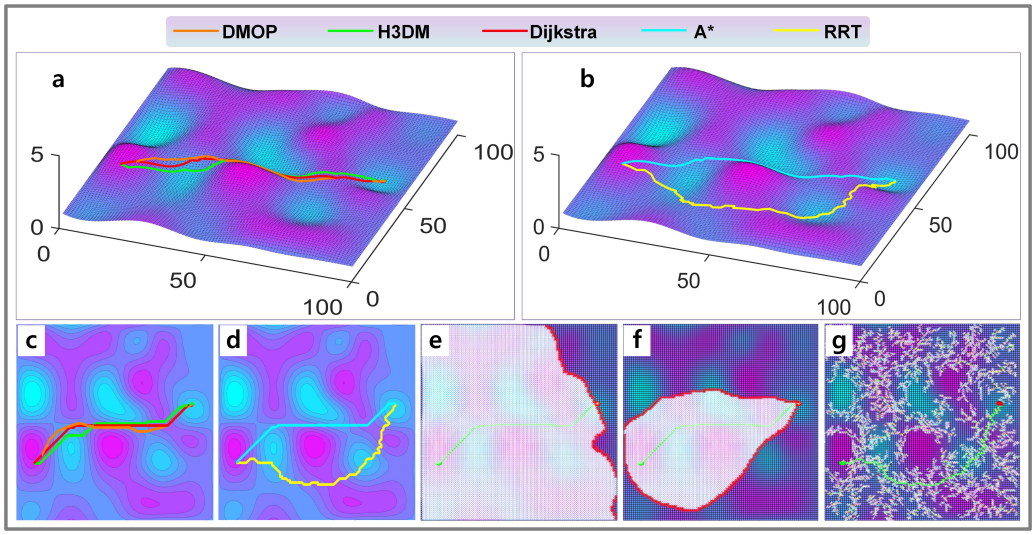

Figure 8. Case 2 of experiments. (**a**) Results of H3DM, Dijkstra and DMOP in 2.5D view. (**b**) Results of RRT and A* in 2.5D view. (**c**) Results of H3DM, Dijkstra and DMOP in top view. (**d**) Results of RRT and A* in top view. (**e**) Searching area of Dijkstra method. (**f**) Searching area of A* method. (**g**) Searching area of RRT method.

Table 3. Quantitative result of Case 2.

Method	Distance (m)	Energy (u)	Sum	Time (s)
DMOP	9453.30	11,197.33	20,650.63	0.33
H3DM	9358.24	11,884.20	21,242.44	11.92
Dijkstra	9251.96	11,357.54	20,609.50	100.83
A*	9251.81	11,393.83	20,645.64	38.38
RRT	12,388.25	16,331.19	28,719.44	3271.31

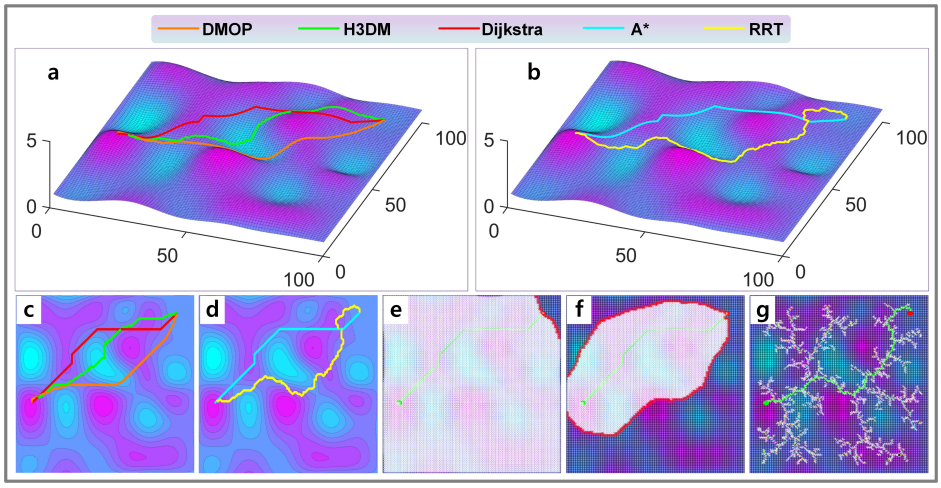

Figure 9. Case 3 of experiments. (**a**) Results of H3DM, Dijkstra and DMOP in 2.5D view. (**b**) Results of RRT and A* in 2.5D view. (**c**) Results of H3DM, Dijkstra and DMOP in top view. (**d**) Results of RRT and A* in top view. (**e**) Searching area of Dijkstra method. (**f**) Searching area of A* method. (**g**) Searching area of RRT method.

Table 4. Quantitative result of Case 3.

Method	Distance (m)	Energy (u)	Sum	Time (s)
DMOP	10,355.95	12,398.70	22,754.65	0.32
H3DM	10,786.18	14,104.28	24,890.46	9.77
Dijkstra	10,371.01	12,308.39	22,679.40	100.97
A*	10,371.01	12,308.39	22,679.40	53.48
RRT	14,490.30	19,435.67	33,925.97	958.51

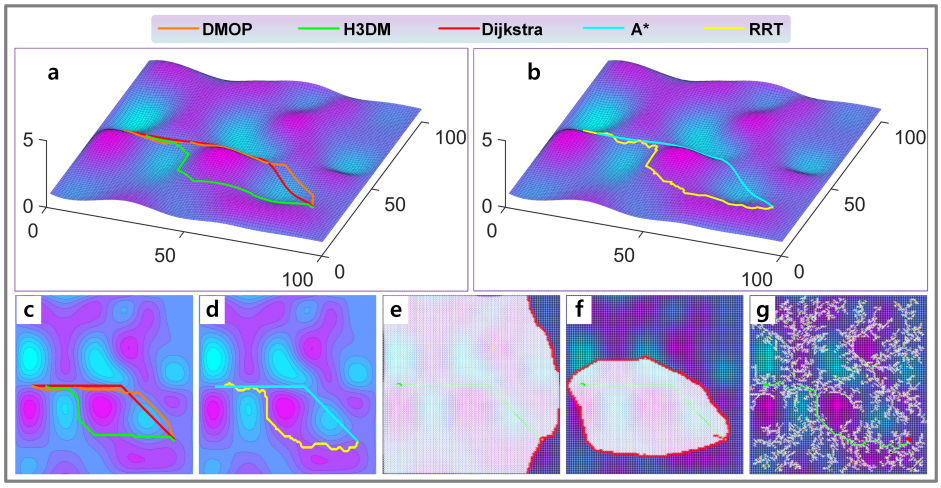

Figure 10. Case 4 of experiments. (**a**) Results of H3DM, Dijkstra and DMOP in 2.5D view. (**b**) Results of RRT and A* in 2.5D view. (**c**) Results of H3DM, Dijkstra and DMOP in top view. (**d**) Results of RRT and A* in top view. (**e**) Searching area of Dijkstra method. (**f**) Searching area of A* method. (**g**) Searching area of RRT method.

Table 5. Quantitative result of Case 4.

Method	Distance (m)	Energy (u)	Sum	Time (s)
DMOP	9500.19	11,754.78	21,254.97	0.34
H3DM	9890.28	12,049.37	21,939.66	12.97
Dijkstra	9246.16	11,444.45	20,690.61	100.97
A*	9246.16	11,444.45	20,690.61	40.37
RRT	11,378.68	14,074.97	25,453.65	2614.11

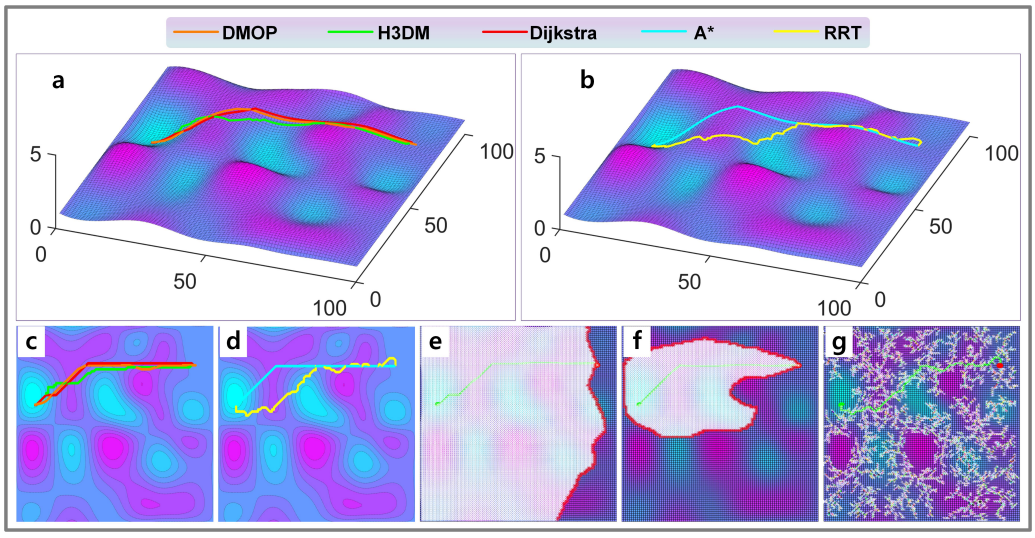

Figure 11. Case 5 of experiments. (**a**) Results of H3DM, Dijkstra and DMOP in 2.5D view. (**b**) Results of RRT and A* in 2.5D view. (**c**) Results of H3DM, Dijkstra and DMOP in top view. (**d**) Results of RRT and A* in top view. (**e**) Searching area of Dijkstra method. (**f**) Searching area of A* method. (**g**) Searching area of RRT method.

Table 6. Quantitative result of Case 5.

Method	Distance (m)	Energy (u)	Sum	Time (s)
DMOP	9017.33	12,892.16	21,909.49	0.29
H3DM	9116.10	13,127.90	22,244.23	10.48
Dijkstra	8917.89	12,470.11	21,387.99	97.07
A*	8835.11	12,629.49	21,464.60	39.91
RRT	11,244.09	16,680.88	27,924.98	2784.60

From the results, we can see that the planning speed of DMOP is extremely fast, which is around 0.32 s, and the quality of its results is very close to those of the Dijkstra method. The Dijkstra method can obtain the global optimal paths but needs about 100 s or more; thus, its efficiency is low. The A* method often takes dozens of seconds to complete the planning. The RRT is very slow and unstable (Case 2 takes 3271 s to find the target, while Case 3 takes 958 s), the global nature of the planned results is average, and the path smoothness is poor. The time required for H3DM to complete the planning task is around 10 s, and the global nature of the paths it plans is a bit worse than that of DMOP. Though DMOP does not find the global optimal solution, its path quality is close to that of the A* method. Considering the search speed and the path quality, the efficiency of DMOP is the best among all the methods.

Overall, if we assume that the paths planned by the DMOP, A*, and H3DM are all acceptable, then the DMOP is over 100 times faster than the A* method and about 30 times faster than the H3DM.

5.2. Discussion

The simulation results show that although the Dijkstra method can find the global optimal path, it always searches almost the whole map to obtain the result. The A* method is recognized as the most efficient method for the single-objective path planning task on 2D maps, which takes less time and has a high quality of solution because of the added heuristic function. In the 2.5D multi-objective path planning task, the A* method still maintains high efficiency. However, we made a special design of the heuristic function of the A* method according to the planning task of this paper to achieve relatively stable results. The special design is that we model its heuristic function with linear distance, curvilinear distance, and energy consumption estimation and set appropriate parameters. For the RRT algorithm, there are blindness and randomness in its search, which leads to its inefficiency and the poor smoothness of the planned paths. From the figures of the results, we can see that RRT searches fewer points compared to the Dijkstra method, but its search area basically covers the whole map.

Whether it is the Dijkstra method, A*, or RRT, they all need to search a certain area, causing the method to take a significant amount of time. The H3DM is built on the idea of pure heuristics, and the algorithm tends to be pure greedy search, with the least number of search points for path planning, so its efficiency is much higher than the previous methods, and the method takes only about 10 s. Due to the extreme compression of the search space, it sacrifices the quality of the solutions to some extent. However, in general, the solution obtained from H3DM does not undergo significant sacrifices, and the time required to plan the path is further reduced, thus indicating an improvement.

Compared with these previous methods, a large number of statistical data show that the DMOP method has a relatively high quality of solution and a very short search time. For the global nature of the solution of DMOP, during the training process DQN learns to judge the peaks and valleys, and it always sees the global view (the whole map as input), which allows it to achieve global path planning from another perspective. Furthermore, through many training tasks, DQN learns to assess the cost of each point in the planning process from a global perspective and thus plan paths that are better globally. For the search time, theoretically, the DMOP method searches the same region as the H3DM method does, and both methods perform greedy selection planning from neighbor points each time. However, due to the complexity of the heuristic function model in the H3DM, it takes a significant amount of time to estimate the cost of each neighbor point. On the other hand, the DMOP requires only simple forward operations, and the calculations inside the network are straightforward, which makes the method very fast.

Furthermore, DMOP is tested on the tasks that it has never seen before. Another important conclusion we can draw from the simulation is that this method possesses robust reasoning capabilities and adaptability to various planning tasks, allowing it to perform any untrained planning task on a given map. On this basis, for exploration robots working on a specific terrain area, the DMOP method can learn to plan paths on this area in advance and quickly generate the navigation path for robots while they are working.

6. Conclusions and Prospect

In this paper, a deep reinforcement learning-based multi-objective 2.5D path planning method is investigated for the energy-saving of ground vehicles. Such a method can efficiently plan paths that have a good trade-off for distance and energy consumption. In the implementation, we solved two technical problems: the convergence problem of DQN in high state space, and realizing multi-objective path planning. Unlike traditional algorithms, the DMOP method does not have a search area, and it directly generates the planned path after loading the map, which has high efficiency and path quality. Simulation shows the proposed method is over 100 times faster than the A* and about 30 times faster than the H3DM method. Also, simulation proves the method has powerful reasoning capability that enables it to perform arbitrary untrained planning tasks on the map where the ground vehicles are required to operate.

Future work will involve the following: (1) studying a new deep reinforcement learning method that converges more quickly than the DQN; and (2) exploring the application of deep reinforcement learning in solving other path planning problems.

Author Contributions: Conceptualization, X.W. and S.H.; methodology, S.H.; software, S.H.; validation, X.W., S.H., and G.H.; formal analysis, G.H.; investigation, G.H. and S.H.; resources, G.H.; data curation, G.H.; writing—original draft preparation, S.H.; writing—review and editing, G.H. and X.W.; visualization, S.H.; supervision, X.W. and G.H.; project administration, X.W.; and funding acquisition, X.W. and S.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China under grant 62263005, the Guangxi Natural Science Foundation under grant 2020GXNSFDA238029, the Key Laboratory of AI and Information Processing (Hechi University), the Education Department of Guangxi Zhuang Autonomous Region under grant 2022GXZDSY004, the Innovation Project of Guangxi Graduate Education under grant YCSW2023298, and the Innovation Project of GUET Graduate Education under grant 2023YCXS124.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclatures

P_n	The n_{th} point in the planned path
P_d	The destination (mushroom position)
$D(\cdot)$	Straight-line distance between two points
D_{total}	Distance for a 2.5D path
$E(\cdot)$	Energy consumption between two points
E_{total}	Energy consumption for a 2.5D path
s	Observation of DQN (agent)
a	Actions given by DQN (agent)
a_h	Actions given by heuristic method
r_w	Wall penalty term in the reward function
r_s	Slope penalty term in the reward function
r_t	Reward term when Mario obtains the mushroom
r_d	Distance penalty term in the reward function

References

1. Farajpour, Y.; Chaoui, H.; Khayamy, M.; Kelouwani, S.; Alzayed, M. Novel Energy Management Strategy for Electric Vehicles to Improve Driving Range. *IEEE Trans. Veh. Technol.* **2022**, *72*, 1735–1747. [CrossRef]
2. Liu, W.; Placke, T.; Chau, K. Overview of batteries and battery management for electric vehicles. *Energy Rep.* **2022**, *8*, 4058–4084. [CrossRef]
3. Wang, J.; Geng, W.; Li, Q.; Li, L.; Zhang, Z. A new flux-concentrating rotor of permanent magnet motor for electric vehicle application. *IEEE Trans. Ind. Electron.* **2021**, *69*, 10882–10892. [CrossRef]
4. Stanton, D.W. Systematic development of highly efficient and clean engines to meet future commercial vehicle greenhouse gas regulations. *SAE Int. J. Engines* **2013**, *6*, 1395–1480. [CrossRef]
5. Brandao, M.; Hashimoto, K.; Santos-Victor, J.; Takanishi, A. Footstep Planning for Slippery and Slanted Terrain Using Human-Inspired Models. *IEEE Trans. Robot.* **2016**, *32*, 868–879. [CrossRef]
6. Chi, W.; Ding, Z.; Wang, J.; Chen, G.; Sun, L. A generalized Voronoi diagram-based efficient heuristic path planning method for RRTs in mobile robots. *IEEE Trans. Ind. Electron.* **2021**, *69*, 4926–4937. [CrossRef]
7. Paton, M.; Strub, M.P.; Brown, T.; Greene, R.J.; Lizewski, J.; Patel, V.; Gammell, J.D.; Nesnas, I.A.D. Navigation on the Line: Traversability Analysis and Path Planning for Extreme-Terrain Rappelling Rovers. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 7034–7041. . [CrossRef]
8. Hadi, B.; Khosravi, A.; Sarhadi, P. Deep reinforcement learning for adaptive path planning and control of an autonomous underwater vehicle. *Appl. Ocean. Res.* **2022**, *129*, 103326. [CrossRef]

9. Usami, R.; Kobashi, Y.; Onuma, T.; Maekawa, T. Two-Lane Path Planning of Autonomous Vehicles in 2.5D Environments. *IEEE Trans. Intell. Veh.* **2020**, *5*, 281–293. [CrossRef]
10. Huang, Z.; Li, D.; Wang, Q. Safe path planning of mobile robot in uneven terrain. *Control. Decis.* **2022**, *37*, 323–330.
11. Santos, L.; Santos, F.; Mendes, J.; Costa, P.; Lima, J.; Reis, R.; Shinde, P. Path planning aware of robot's center of mass for steep slope vineyards. *Robotica* **2020**, *38*, 684–698. [CrossRef]
12. Linhui, L.; Mingheng, Z.; Lie, G.; Yibing, Z. Stereo Vision Based Obstacle Avoidance Path-Planning for Cross-Country Intelligent Vehicle. In Proceedings of the 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery, Tianjin, China, 14–16 August 2009, Volume 5, pp. 463–467. [CrossRef]
13. Shin, J.; Kwak, D.; Kwak, K. Model predictive path planning for an autonomous ground vehicle in rough terrain. *Int. J. Control. Autom. Syst.* **2021**, *19*, 2224–2237. [CrossRef]
14. Inotsume, H.; Kubota, T.; Wettergreen, D. Robust Path Planning for Slope Traversing Under Uncertainty in Slip Prediction. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3390–3397. [CrossRef]
15. Kyaw, P.T.; Le, A.V.; Veerajagadheswar, P.; Elara, M.R.; Thu, T.T.; Nhan, N.H.K.; Van Duc, P.; Vu, M.B. Energy-efficient path planning of reconfigurable robots in complex environments. *IEEE Trans. Robot.* **2022**, *38*, 2481–2494. [CrossRef]
16. Ma, B.; Liu, Q.; Jiang, Z.; Che, D.; Qiu, K.; Shang, X. Energy-Efficient 3D Path Planning for Complex Field Scenes Using the Digital Model with Landcover and Terrain. *ISPRS Int. J. -Geo-Inf.* **2023**, *12*, 82. [CrossRef]
17. Ganganath, N.; Cheng, C.T.; Chi, K.T. A constraint-aware heuristic path planner for finding energy-efficient paths on uneven terrains. *IEEE Trans. Ind. Inform.* **2015**, *11*, 601–611. [CrossRef]
18. Jiang, C.; Hu, Z.; Mourelatos, Z.P.; Gorsich, D.; Jayakumar, P.; Fu, Y.; Majcher, M. R2-RRT*: Reliability-based robust mission planning of off-road autonomous ground vehicle under uncertain terrain environment. *IEEE Trans. Autom. Sci. Eng.* **2021**, *19*, 1030–1046. [CrossRef]
19. Huang, G.; Yuan, X.; Shi, K.; Liu, Z.; Wu, X. A 3-D Multi-Object Path Planning Method for Electric Vehicle Considering the Energy Consumption and Distance. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 7508–7520. [CrossRef]
20. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of go without human knowledge. *Nature* **2017**, *550*, 354–359. [CrossRef]
21. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
22. Fu, M.; Huang, L.; Rao, A.; Irissappane, A.A.; Zhang, J.; Qu, H. A Deep Reinforcement Learning Recommender System With Multiple Policies for Recommendations. *IEEE Trans. Ind. Inform.* **2022**, *19*, 2049–2061. [CrossRef]
23. Fan, C.; Zeng, L.; Sun, Y.; Liu, Y.Y. Finding key players in complex networks through deep reinforcement learning. *Nat. Mach. Intell.* **2020**, *2*, 317–324. [CrossRef]
24. Hu, Y.; Yao, Y.; Lee, W.S. A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs. *Knowl.-Based Syst.* **2020**, *204*, 106244. [CrossRef]
25. Keskin, M.E.; Yılmaz, M.; Triki, C. Solving the hierarchical windy postman problem with variable service costs using a math-heuristic algorithm. *Soft Comput.* **2023**, *27*, 8789–8805. [CrossRef]
26. Pan, G.; Xiang, Y.; Wang, X.; Yu, Z.; Zhou, X. Research on path planning algorithm of mobile robot based on reinforcement learning. *Soft Comput.* **2022**, *26*, 8961–8970. [CrossRef]
27. Sichkar, V.N. Reinforcement learning algorithms in global path planning for mobile robot. In Proceedings of the 2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), Sochi, Russia, 25–29 March 2019, pp. 1–5.
28. Ren, J.; Huang, X.; Huang, R.N. Efficient Deep Reinforcement Learning for Optimal Path Planning. *Electronics* **2022**, *11*, 3628. [CrossRef]
29. Chu, Z.; Wang, F.; Lei, T.; Luo, C. Path planning based on deep reinforcement learning for autonomous underwater vehicles under ocean current disturbance. *IEEE Trans. Intell. Veh.* **2022**, *8*, 108–120. [CrossRef]
30. Wu, K.; Esfahani, M.A.; Yuan, S.; Wang, H. TDPP-Net: Achieving three-dimensional path planning via a deep neural network architecture. *Neurocomputing* **2019**, *357*, 151–162. [CrossRef]
31. Ugur, D.; Bebek, O. Fast and Efficient Terrain-Aware Motion Planning for Exploration Rovers. In Proceedings of the 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), Lyon, France, 23–27 August 2021; pp. 1561–1567. [CrossRef]
32. Nie, Z.; Zhao, H. Research on Robot Path Planning Based on Dijkstra and Ant Colony Optimization. In Proceedings of the 2019 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), Shanghai, China, 21–24 November 2019; pp. 222–226. [CrossRef]
33. Erke, S.; Bin, D.; Yiming, N.; Qi, Z.; Liang, X.; Dawei, Z. An improved A-Star based path planning algorithm for autonomous land vehicles. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 1729881420962263. [CrossRef]
34. Kang, L.; Mao, L.C. Path Planning Based on Heuristic Rapidly-Exploring Random Tree for Nonholonomic Mobile Robot. *Appl. Mech. Mater.* **2014**, *494*, 1161–1164. [CrossRef]