

## Article

# Fast Path Planning for Long-Range Planetary Roving Based on a Hierarchical Framework and Deep Reinforcement Learning

Ruijun Hu <sup>1,\*</sup>  and Yulin Zhang <sup>2</sup> 

<sup>1</sup> College of Aerospace Science and Engineering, National University of Defense Technology, Changsha 410073, China

<sup>2</sup> College of Control Science and Engineering, Zhejiang University, Hangzhou 310000, China; zhangyulin@zju.edu.cn

\* Correspondence: huruijun17\_nudt@163.com

**Abstract:** The global path planning of planetary surface rovers is crucial for optimizing exploration benefits and system safety. For the cases of long-range roving or obstacle constraints that are time-varied, there is an urgent need to improve the computational efficiency of path planning. This paper proposes a learning-based global path planning method that outperforms conventional searching and sampling-based methods in terms of planning speed. First, a distinguishable feature map is constructed through a traversability analysis of the extraterrestrial digital elevation model. Then, considering planning efficiency and adaptability, a hierarchical framework consisting of step iteration and block iteration is designed. For the planning of each step, an end-to-end step planner named SP-ResNet is proposed that is based on deep reinforcement learning. This step planner employs a double-branch residual network for action value estimation, and is trained over a simulated DEM map collection. Comparative analyses with baselines demonstrate the prominent advantage of our method in terms of planning speed. Finally, the method is verified to be effective on real lunar terrains using CE2TMap2015.



**Citation:** Hu, R.; Zhang, Y. Fast Path Planning for Long-Range Planetary Roving Based on a Hierarchical Framework and Deep Reinforcement Learning. *Aerospace* **2022**, *9*, 101. <https://doi.org/10.3390/aerospace9020101>

Academic Editor: Gokhan Inalhan

Received: 22 December 2021

Accepted: 10 February 2022

Published: 14 February 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Keywords:** planetary rovers; global path planning; deep reinforcement learning; hierarchical framework; computational efficiency

## 1. Introduction

The path planning of planetary rovers is crucial for improving exploration benefits and system safety in extraterrestrial exploration. The existing lunar and Mars rovers mainly rely on man-involved local planning for path determination due to the limited local perception ability and surface mobility. For instance, the Yutu and Yutu-2 rovers move towards an object of interest for exploration through a series of waypoints and segmented paths that are terrestrially planned [1,2]. The present path planning systems seem to be reliable, but they are less efficient, and they can only plan as far as a rover can see. Orbital imageries of extraterrestrial planets can be provided at an effectively unlimited distance, which provides the possibility, as well as the basis, for long-range global path planning for extraterrestrial surface robotic activities [3,4]. High-precision orbital images can be further processed into digital elevation models (DEMs) carrying vivid three-dimensional topographical data, such as the CE2TMap2015 [5,6] for the Moon and the HiRISE [7] data for Mars.

Generally, global path planning for planetary roving is divided into two stages, namely, map construction and path searching. First, the original orbital images are interpreted into computable configuration spaces in which free space and obstacle space are identified. Then, the right path can be identified by some path searching algorithms, such as the A\* [8,9], the rapidly exploring random trees (RRT) [10,11], the ant colony algorithm [12], and the genetic algorithm [13], etc. These methods are operable in most instances, but may suffer from the exponential explosion of computation time when it comes

to wide-range or high-resolution maps. Learning methods, such as deep learning and deep reinforcement learning (DRL), have been applied to the global path planning of planetary rovers in pursuit of improving computing efficiency [14] and reducing dependence on human experience [15–17]. However, there are still some questions that remain unsolved in learning-based methods. For the artificial neural network (ANN)-based planners, one very important issue is the difficulty of adapting to maps with various sizes due to their specific network structures.

In this paper, we propose a novel learning-based planning method for the global path planning of planetary roving, which can significantly improve the planning speed for long-range roving and which, meanwhile, can adapt to arbitrarily sized maps. First, a kind of binary feature map is obtained from the original extraterrestrial DEM through topography and illumination analysis. The feature map is designed to provide more distinguishable information for learning and planning. Then, a hierarchical planning framework is designed to resolve the issue into two hierarchies, namely, step iteration and block iteration. For the planning within step iteration, an end-to-end step planner named SP-ResNet is constructed using DRL. The planner employs double branches of residual networks for feature reasoning and action value estimation, and is trained over a simulated extraterrestrial DEM collection. In the results, comparative large-range planning tests with A\*, RRT, and goal-biasing RRT (Gb-RRT) [18] are conducted, demonstrating the prominent advantage of the proposed method in terms of planning efficiency. Furthermore, the effectiveness and the feasibility of the method are validated on real lunar terrains from CE2TMap2015.

The main contributions of this paper can be summarized as follows:

- (1) A novel learning-based method combining a hierarchical planning framework and an end-to-end step planner improves the computational efficiency for long-range path planning.
- (2) The step planner learns and conducts path planning on binary feature maps without relying on any label supervision or imitation.
- (3) The hierarchical planning framework solves the problem that a single ANN-based planner only adapts to specifically sized maps.

The remainder of this paper is organized as follows. The related work is reviewed in Section 2. Section 3 formulates the path planning problem in an end-to-end style. Section 4 presents the learning-based planning methodology in detail, including the feature map, the hierarchical planning framework, and the SP-ResNet planner. Section 5 explains the planning application of the method. Section 6 illustrates the numerical results and analysis, and this is followed by an overall conclusion in Section 7.

## 2. Related Work

Global path planning methods for long-range planetary roving can be generally divided into two categories, namely, the conventional searching-based methods as well as the learning-based methods.

Conventionally, given computable configuration spaces, such as the Voronoi diagrams or grid maps, the right path from the start point to the target can be solved by a wide range of searching algorithms, such as the A\* [8,9], the SD\*lite [1], the RRT [10,11], and the ant colony algorithm [12], etc. Various improvements have been made to these methods. For instance, surface constraints, such as the terrain slope, roughness, and illumination over the original DEM, were considered in the cost function of the A\* algorithm, which improved the feasibility and usability of the planned paths [19,20]. YU X Q et al. [21] introduced a safety heuristic function for the A\* algorithm to raise the overall security of the planned paths in large-range lunar areas. Sutoh M et al. [22] discussed the influence of rover motion forms and isolation conditions on the planned paths with Dijkstra's algorithm. However, there are some shortcomings in the conventional path planning methods. For one thing, the path planning performance depends on configuration space interpretation, which partly relies on human experience. For another, these path planning methods are

usually searching-based, meaning that the computation time increases exponentially with the increase in the size of the solution space and it, thus, cannot meet the requirements of long-range roving and complex obstacles, or when some time-varying factors, such as illumination, temperature, or communication, must be considered.

Learning-based methods have also been explored by the researchers for the global path planning of planetary roving. Wang J. et al. [14] trained a deep learning-based path predictor using label paths generated by A\* that could directly generate a probability distribution of the optimal path on an obstacle map. The distribution was used for the guidance of RRT-based final path planning, which significantly improved the path planning speed. In addition, the value iteration network (VIN) was applied to end-to-end path planning for the Mars remote images [15]. Furthermore, a soft action policy was applied to generate a soft VIN that outperformed the standard VIN in terms of the path planning accuracy [16]. Zhang J. et al. [17] proposed an end-to-end planner based on deep reinforcement learning (DRL) that could directly solve the sequenced nodes of a safety path for Mars remote sensing imageries through the iterative application of the planner. It can be concluded that learning-based methods are preferable alternatives that can not only provide end-to-end planning directly from pixel images without the need for artificial computable configuration space representation, but they can also get rid of the problem that the planning time increases exponentially with the problem space size. Nevertheless, the current learning-based methods mainly employ artificial neural networks (ANNs) to build the planning policy, and hence have difficulty adapting to planning tasks on maps with various sizes due to the limitations of the specific network structures. It is really not advisable to plan on reshaped maps, since this may lead to a serious loss of accuracy and the impracticability of the planned path.

This work is devoted to proposing a novel learning-based planning method, which can remarkably improve the computing speed of global path planning for long-range planetary roving and, meanwhile, improve the previous learning-based methods' poor adaptability to the maps of various sizes. An DRL-based path planner, employing double branches of residual networks, is built to directly plan on binary feature maps. Furthermore, a hierarchical planning framework is designed in pursuit of high planning speed and adaptability to arbitrarily sized maps. The method is able to plan faster than the baselines and is validated to be effective in planning tasks with real lunar terrain maps.

### 3. Problem Formulation

Generally, path planning is intended to solve a finite path from a start point to a target point on a map, with no collision with obstacles. For the development of a learning-based planning system, we further arranged the elements into the input and output of a step planner. For the global path planning based on the planetary DEM, the input consists of the current rover position, the target point, and the global map containing some form of environment information, which we call the feature map. The output refers to the best moving direction suggested by the planner at each step. For ease of understanding, some related elements should be further explicitly defined, as follows:

**Step planner:** An end-to-end planner designed based on ANNs, which suggests an optimal action for one step according to the rover position, the target position, and a specifically sized feature map.

**Feature map:**  $\mathbf{F}$ , an  $M \times M$  image representing distinguishable obstacle information.

**Block:**  $\mathbf{B}$ , a feature map with a specific size, e.g.,  $N \times N$ , which is intercepted or deformed from  $\mathbf{F}$ , to be directly fed to the step planner for planning. Due to the fixed network architecture of the planner, the planner can only plan on a specifically sized map, which is why the block is introduced. The next section addresses the role that the block plays in adapting the end-to-end planner to maps with arbitrary sizes.

**Start point:**  $(sx, sy)$ , the initial position of a planning task.

**Target point:**  $(tx, ty)$ , the target position of a planning task.

**Rover position:**  $(rx_k, ry_k)$ , rover position at the current step.  $k$  is the step index.

Action:  $a_k$ , one of the eight directions pointing to the surrounding grid points suggested by the step planner, based on which  $(rx_k, ry_k)$  is updated.

Path:  $\psi = \{\varphi_k | k = 0, 1, 2, \dots\} = \{(rx_k, ry_k) | k = 0, 1, 2, \dots\}$ , the sequenced nodes determined by the planner. If  $\mathbf{O}$  is used to represent the obstacle area of the map, a safe path is attributed to  $\forall \varphi_k \notin \mathbf{O}, k = 0, 1, 2, \dots$ .

## 4. Learning-Based Path Planning Methodology

### 4.1. Feature Map

The planetary DEM data include 3D elevation information for extraterrestrial surfaces that is not suitable for learning and planning with the ANNs. For feature reasoning and planning by the end-to-end planner, a kind of binary feature map with more distinguishable characteristics is constructed based on the local traversability analysis, including the terrain amplitude, slope, and illumination.

#### 4.1.1. Traversability Analysis

##### (1) Terrain amplitude

The terrain amplitude, essentially referring to the local relative elevation, directly reflects the locations and shapes of the craters and highlands in the DEM. The terrain amplitude can be calculated with  $|e(px, py) - e_0|$ .  $(px, py)$  represents the pixel coordinates,  $e$  represents the elevation, and  $e_0$  represents the local level. A terrain amplitude traversability map  $\mathbf{T}^{amplitude}$  can be filled by  $\mathbf{T}^{amplitude}(px, py) = 1$  if  $|e(px, py) - e_0| < \varepsilon^{amplitude}$ , and  $\mathbf{T}^{amplitude}(px, py) = 0$  if  $|e(px, py) - e_0| \geq \varepsilon^{amplitude}$ . On the traversability map, 1 represents the traversable area, and 0 represents the untraversable area.  $\varepsilon^{amplitude}$  represents the safety threshold of the terrain amplitude.

##### (2) Slope

It is preferable for planetary rovers to move on areas with gentle slopes. A  $3 \times 3$  grid rolling window around  $(px, py)$  is created to determine the local slope, which can be calculated with:

$$S(px, py) = \max \left\{ \arctan \left( \frac{|e^1 - e^i|}{d} \right) \middle| i = 2, 3, \dots, 8 \right\}, \quad (1)$$

where  $\arctan \left( \frac{|e^1 - e^i|}{d} \right)$  represents the slope along the eight directions around  $(px, py)$ , and  $e^1 = e(px, py)$ ,  $e^i = e(px^i, py^i)$ , and  $d = res \sqrt{(px - px^i)^2 + (py - py^i)^2}$ . A slope traversability map  $\mathbf{T}^{slope}$  can be constituted using  $\mathbf{T}^{slope}(px, py) = 0$  if  $S(px, py) \geq \varepsilon^{slope}$ , and  $\mathbf{T}^{slope}(px, py) = 1$  if  $S(px, py) < \varepsilon^{slope}$ .  $\varepsilon^{slope}$  is the safety threshold for the slopes, and  $res$  represents the DEM resolution.

##### (3) Illumination

Illumination is an essential condition for the rovers to maintain warmth and ample solar power. The accessibility of sunlight depends on the angle between the solar ray vector  $\vec{a}$  and the normal vector of the local surface  $\vec{n}$ . The angle can be quantitatively expressed with

$$\beta(px, py) = \arccos \left( \frac{\vec{n} \bullet \vec{a}}{|\vec{n}| |\vec{a}|} \right), \beta \in [0, \pi]. \quad (2)$$

The illumination is determined to be accessible only if  $\beta$  is larger than some threshold. An illumination traversability map  $\mathbf{T}^{illumination}$  can be filled using  $\mathbf{T}^{illumination}(px, py) = 0$  if  $\beta(px, py) < \varepsilon^{illumination}$ , and by  $\mathbf{T}^{illumination}(px, py) = 1$  if  $\beta(px, py) \geq \varepsilon^{illumination}$ .  $\varepsilon^{illumination}$  represents the safety threshold of  $\beta$ , and  $\varepsilon^{illumination} > \frac{\pi}{2}$ .  $\vec{n}$  can be calculated based on the  $3 \times 3$  grid rolling window with:

$$\vec{n} = (res, 0, e(px + 1, py) - e(px, py)) \otimes (0, -res, e(px, py - 1) - e(px, py)). \quad (3)$$

#### 4.1.2. Feature Map Calculation

Based on the traversability maps, operations are conducted with pixels, and a comprehensive feature map  $\mathbf{F}$  is constructed; that is,  $\mathbf{F}$  can be obtained with:

$$\mathbf{F} = \mathbf{T}^{amplitude} \cap \mathbf{T}^{slope} \cap \mathbf{T}^{illumination}. \quad (4)$$

Figure 1 shows an example of the traversability maps and the added comprehensive feature map, which is based on a  $10 \times 10 \text{ km}^2$  DEM at the bottom of a crater near the south lunar pole from the CE2TMap2015. The thresholds are set as:  $\epsilon^{amplitude} = 20(\text{m})$ ,  $\epsilon^{slope} = 15(\text{deg})$ ,  $\epsilon^{illumination} = 102(\text{deg})$ , and  $\vec{a} = (2, 2, -1)^T$ . It can be seen that the feature map, with consideration of the topography and illumination traversability, conveys the obstacle areas in a more distinguishable way than the original DEM.

#### 4.2. Hierarchical Planning Framework

Trying to cope with a planning task on a feature map with an arbitrary size directly based on an end-to-end planner does not work due to the defects of fixed network architectures. Even if the feature map could be deformed to match the planner input, a serious loss of accuracy and feasibility might occur. A hierarchical planning framework is designed to break through this defect, which divides the issue into two levels of iteration, namely, step iteration and block iteration. It should be noted that step iteration is carried out on the block with a size of  $N \times N$ , which matches the input of the end-to-end planner, while the block iteration is carried out on the original feature map with an arbitrary shape of  $M \times M$ .

##### 4.2.1. Step Iteration

An end-to-end step planner is designed to solve the optimal moving action  $a_k$  for the rover at the current step according to the input rover position  $(rx_i, ry_i)$ , the target position  $(tx, ty)$ , and the block map  $\mathbf{B}$ . Through the iterative application of the step planner, a complete path from the start point to the target point can be worked out, which is known as step iteration, or SI for short.

##### 4.2.2. Block Iteration

Block iteration employs SI to plan a segmented path using blocks on the original map along a rough initial path; that is to say, the planning is divided into two phases, including the initial rough planning and the accurate block planning, as shown by  $\text{Si } 0$  and  $\text{Si } j$  in Figure 2, respectively. The final path on a feature map with an arbitrary size is usually obtained by splicing the segmented paths that are planned on multiple blocks.

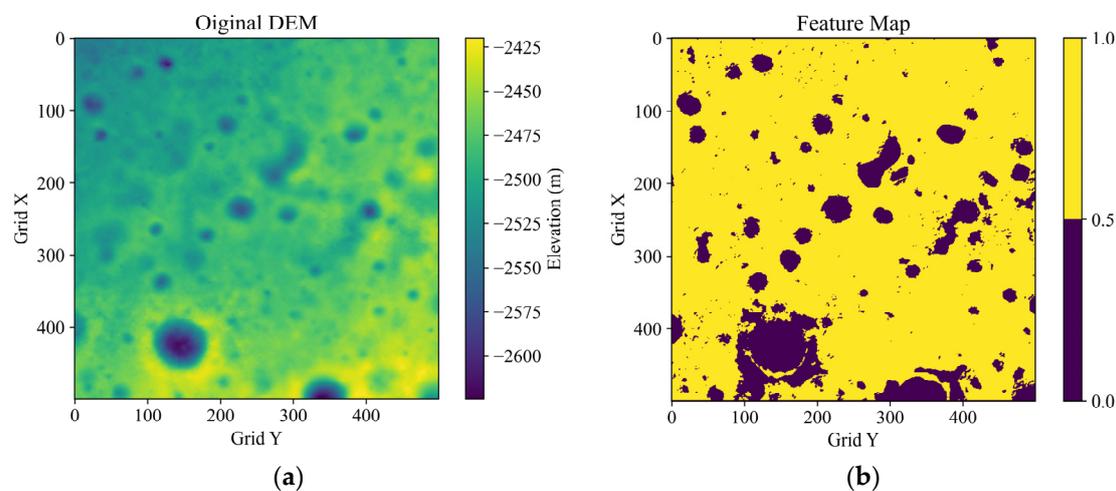
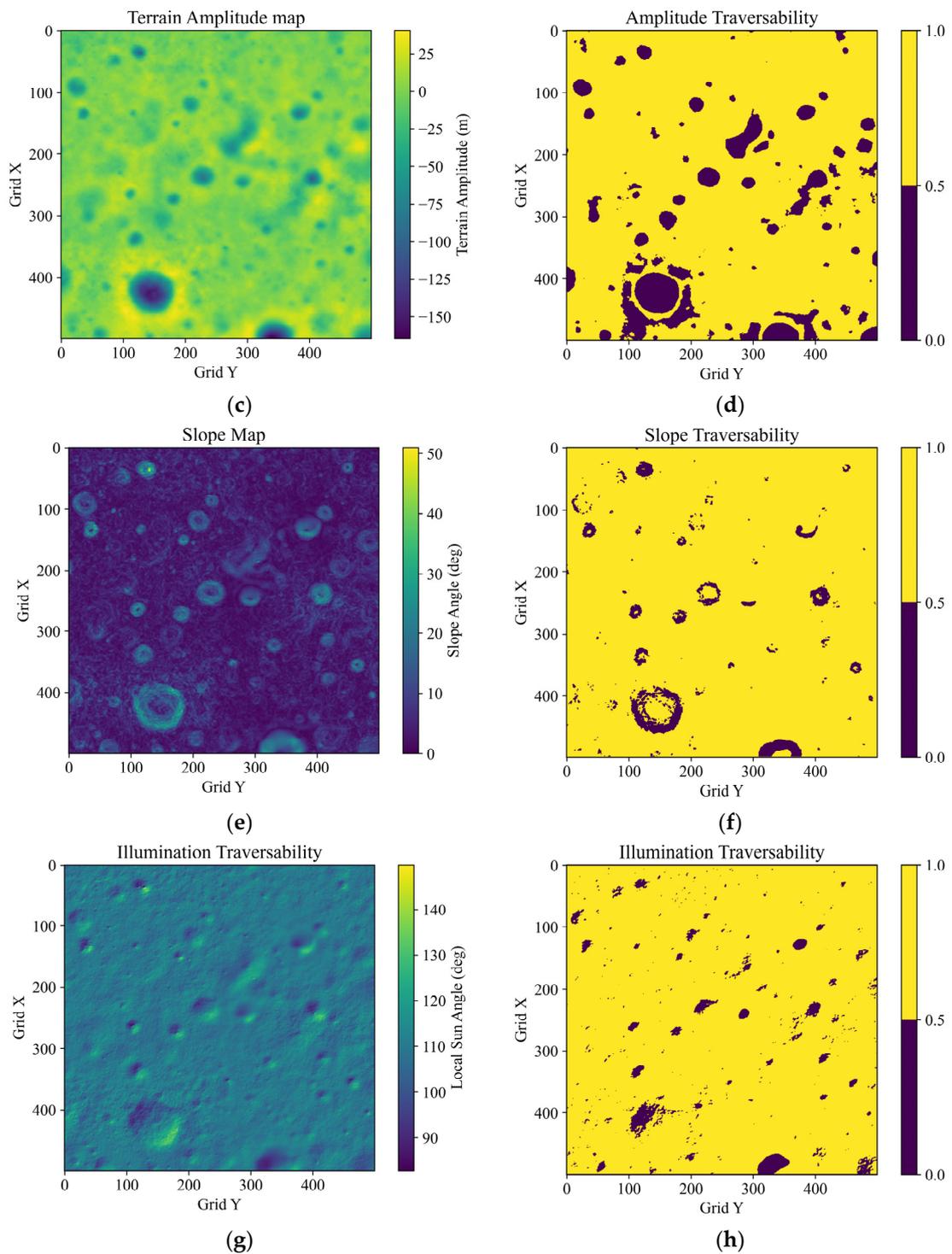
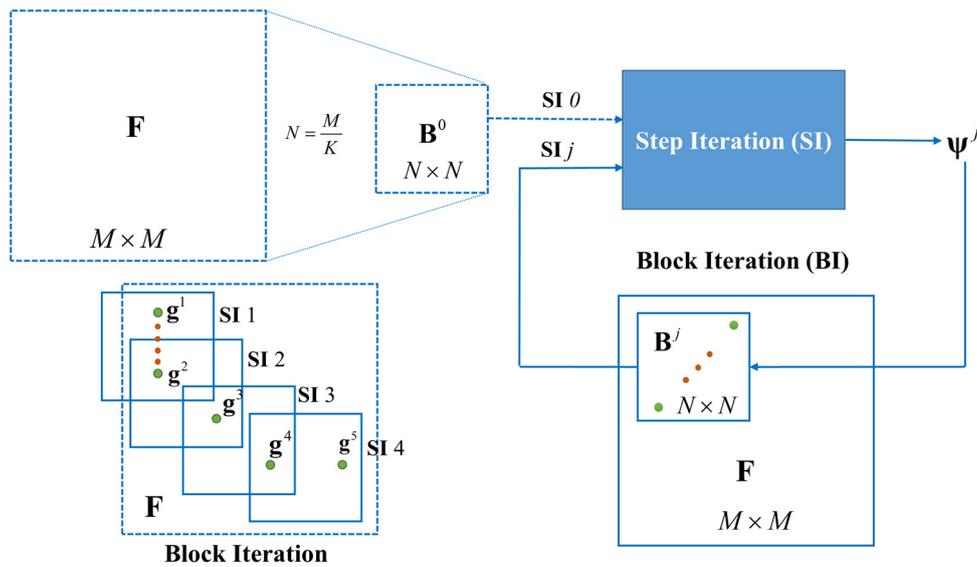


Figure 1. Cont.



**Figure 1.** An example of traversability analysis and the feature map: (a) the original DEM, (b) the feature map, (c) the terrain amplitude map, (d) the amplitude traversability, (e) the slope map, (f) the slope traversability, (g) the illumination map, and (h) the illumination traversability.



**Figure 2.** The block iteration framework.

Given an original feature map  $F$  with a size of  $M \times M$ , the map is first deformed into an  $N \times N$  block  $B^0$ , for which an initial rough path is planned through  $SI_0$ . The initial path is denoted by  $\Psi^0 = \{\varphi_k^0 | k = 0, 1, 2, \dots\} = \{(rx_k^0, ry_k^0) | k = 0, 1, 2, \dots, L\}$ , which can be calculated with:

$$rx_k^0 = rx_k^{0*} \times K, ry_k^0 = ry_k^{0*} \times K, k = 1, 2, \dots, L, \quad (5)$$

where  $K = M/N$  represents the transformation ratio, and  $L$  indicates the initial path length. We use  $\bullet^j$  to indicate the variables solved by  $SI_j$ , and we use  $\bullet^*$  to distinguish the variable for the block coordinate system.

Along  $\Psi^0$ , a series of guide nodes  $(gx^j, gy^j), j = 1, 2, \dots, J$  are determined that serve as the endpoints of the precise block planning of the second phase. The guide nodes are illustrated by  $g^j$  in Figure 2.  $J$  is up to the length of the initial path by  $J = \frac{2LK}{N}$ . The guide nodes can be determined by:

$$gx^j = rx_{\frac{Nj}{2K}}^0, gy^j = ry_{\frac{Nj}{2K}}^0, j = 1, 2, \dots, J. \quad (6)$$

Each block  $B^j$  is located using a pair of adjacent guide nodes  $(gx^j, ry^j)$  and  $(gx^{j+1}, gy^{j+1})$  as follows:

$$\begin{cases} B_j = F(px_j - N/2 : px_j + N/2, px_j - N/2 : px_j + N/2) \\ px^j = \frac{gx^j + gx^{j+1}}{2} \\ py^j = \frac{gy^j + gy^{j+1}}{2} \end{cases}, \quad (7)$$

where  $(px_j, py_j)$  indicates the center position of  $B^j$ . The segmented path on the block is then planned with  $SI_j$ . The path coordinates for block  $B^j$  and the original feature map  $F$  can be converted using the following formula:

$$\begin{cases} rx_k^j = rx_k^{j*} + px^j - N/2 \\ ry_k^j = ry_k^{j*} + py^j - N/2 \end{cases}, k = 0, 1, 2, \dots \quad (8)$$

### 4.3. SP-ResNet Planner

#### 4.3.1. DRL Model

Step planning is based on DRL, which can be essentially modeled by the Markov decision process. At the  $k^{th}$  time step, the navigator observes a state  $\mathbf{s}_k$  and selects an action  $a_k$  from the action set  $\mathbf{A}$ . This decision is guided by the policy  $\pi(\theta_k)$ , in which  $\theta_k$  represents the weights in the Q-network [23]. Afterward, the agent reaches the next state  $\mathbf{s}_{k+1}$  and obtains an instant reward  $r_k$ . The state-action value  $q_\pi(\mathbf{s}, a)$  is introduced as  $q_\pi(\mathbf{s}_k, a_k; \theta_k) = E_\pi[G_k | \mathbf{s}_k, a_k]$  and used to evaluate the current policy  $\pi(\theta_k)$ , where  $G_k = \sum_{t=0}^{\infty} \gamma^t r_{k+t+1}$  represents the expected cumulative sum of rewards, and  $\gamma \in (0, 1)$  is a discount factor. The best action choice for the current policy is determined by  $q_\pi(\mathbf{s}, a)$ , according to:

$$a_k = \underset{a'}{\operatorname{argmax}} q_\pi(\mathbf{s}_k, a'). \tag{9}$$

As the agent learns from interacting experiences within the environment,  $q_\pi(\mathbf{s}_k, a_k; \theta_k)$  can be optimized using the Bellman equation:

$$q_\pi(\mathbf{s}_k, a_k; \theta_{k+1}) = q_\pi(\mathbf{s}_k, a_k; \theta_k) + \alpha \left[ r_k + \gamma \max_{a'} q_\pi(\mathbf{s}_{k+1}, a') - q_\pi(\mathbf{s}_k, a_k; \theta_k) \right]. \tag{10}$$

The Q-network can then be trained by minimizing the loss function, which can be expressed as follows:

$$\text{Loss}(\theta_k) = E[(r_k + \gamma \max_{a'} q_\pi(\mathbf{s}_{k+1}, a'; \theta_k^-) - q_\pi(\mathbf{s}_k, a_k; \theta_k))^2], \tag{11}$$

where  $\theta_k^-$  represents the weights in a target Q-network that are only updated for a specific number of steps [24].

#### 4.3.2. SP-ResNet Planner

The Q-network is the predominant learner as well as the planner for step planning, which determines features and holds the planning policy. We designed a novel Q-network consisting of double branches of residual networks [25] for global and local obstacle abstraction, as shown in Figure 3. The planner is named “the step planner based on ResNet”, or “SP-ResNet” for short.

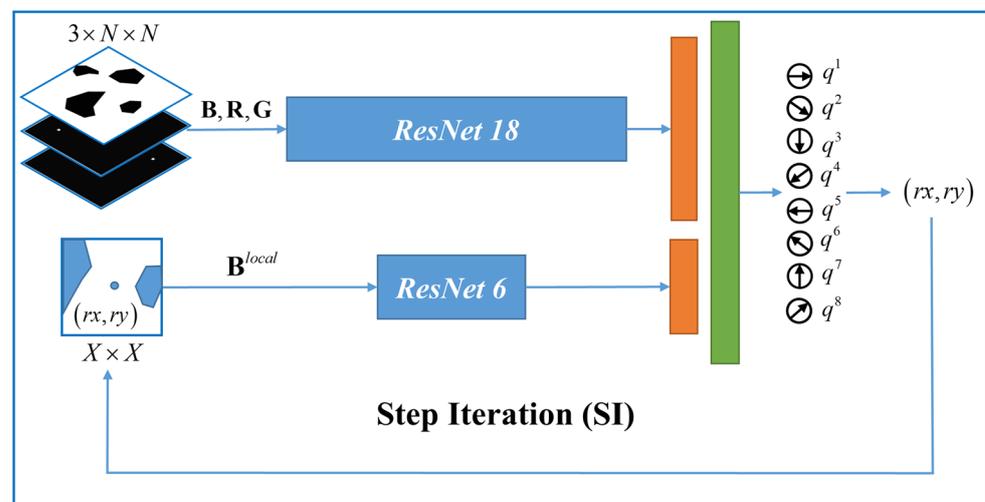


Figure 3. The SP-ResNet planner.

The global branch employs ResNet18 to extract the global information, including  $(rx, ry)$ ,  $(tx, ty)$ , and  $\mathbf{B}$ . The block size is denoted by  $N \times N$ . To provide a more direct

spatial relationship for the rover position and the target position for the branch,  $(rx, ry)$  and  $(tx, ty)$  are marked on an  $N \times N$  blank layer as a bright pixel. The added layers are denoted as  $\mathbf{R}$  and  $\mathbf{G}$ , individually. Therefore, the input of the global branch is arranged as a three-layered state cube with dimensions of  $3 \times N \times N$ . The local branch employs ResNet6 to work on a local area around  $(rx, ry)$  on  $\mathbf{B}$ , which contains three BasicBlocks. The local area is a matrix denoted as  $\mathbf{B}^{local}$ , whose size is denoted by  $X \times X$ .  $\mathbf{B}^{local}$  can be obtained with:

$$\mathbf{B}^{local} = \mathbf{B}(rx - X/2 : rx + X/2, ry - X/2 : ry + X/2). \quad (12)$$

The outputs of the two branches are fused through two fully connected layers for the value estimation of eight actions  $a^i \in \mathbf{A}, i = 1, 2, \dots, 8$ , which denote the motions in the eight directions. The optimal motion at each step can be planned using Equation (9).

#### 4.3.3. Step Planning Iteration

It is important to specify how the SP-ResNet works in the step planning iteration. For each step,  $\mathbf{B}$ ,  $\mathbf{R}$ , and  $\mathbf{B}^{local}$  are updated according to the current  $(rx, ry)$ , while the action selection is achieved by the one-time reasoning of the Q-network. The operation flow of the step iteration is summarized in Table 1.

**Table 1.** The step iteration by the SP-ResNet planner.

Step Iteration (SI)	
a.	<b>Block input:</b> $\mathbf{B}, \mathbf{G}, sx, sy, tx, ty$
b.	<b>Block initialization:</b> $rx_0 = sx, ry_0 = sy, X$ <b>Step planning:</b>
c.	c.1. Obtain $\mathbf{R}$ and $\mathbf{B}^{local}$ according to $(rx_k, ry_k)$ c.2. Obtain $\mathbf{q}_k(\mathbf{s}_k, a)$ with Q-network reasoning c.3. Action: $a_k = \underset{a'}{\operatorname{argmax}} q_\pi(\mathbf{s}_k, a')$
d.	<b>Position update:</b> If $rx_k = tx$ and $ry_k = ty$ , output $\Psi = \{(rx_k, ry_k)   k = 1, 2, 3, \dots\}$ and quit; else, $rx_k = rx_{k+1}, ry_k = ry_{k+1}$ , go to <b>Step planning</b> .

## 5. Training and Planning

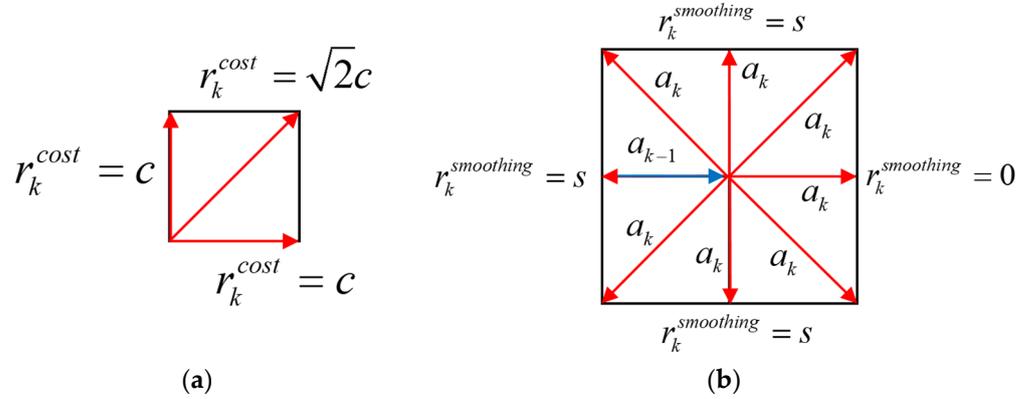
### 5.1. Rewards for Training

In fact, it is hard work to train an ANN to learn image-based path planning without supervision or imitation [18,21]. A comprehensive reward strategy is designed for our SP-ResNet planner to help it to efficiently learn the behaviors required to approach the target, avoid obstacles, optimize the accumulated length, and smooth the path.

The reward for arrival is denoted as  $r_k^{arrival}$ . If the rover arrives at the target position,  $r_k^{arrival} = 1$ , and the current episode is ended, while at other moments,  $r_k^{arrival} = 0$ . The reward for a hazard is denoted by  $r_k^{hazard}$ . If the rover enters any hazard area, a penalty of  $-1$  is given to  $r_k^{hazard}$ , and the current episode is ended; else,  $r_k^{hazard} = 0$ . The hazard area can be classified by  $\mathbf{T}(rx, ry) = 0$ . The reward for approaching  $r_k^{approach}$  is designed as a linear function of the distance variance between  $(rx_k, ry_k)$  and  $(tx, ty)$ , denoted as  $r_k^{approach} = \lambda(d_{k-1}^{target} - d_k^{target})$ . If the rover approaches the target,  $r_k^{approach}$  takes a positive value, while if the rover leaves the target, the value is negative. The reward for the energy cost  $r_k^{cost}$  is introduced for the path length optimization, whose value is based on  $a_k$ . If  $a_k$  goes along the grid edge,  $r_k^{cost} = c$ , but if  $a_k$  goes along the diagonal,  $r_k^{cost} = \sqrt{2}c$ , where  $c$  stands for the energy cost per unit length. The reward for smoothing  $r_k^{smoothing}$  can also be considered to be a turning cost for the rover whose value relies on the actions taken in adjacent moments. If  $a_k$  is along the same direction with  $a_{k-1}$ ,  $r_k^{smoothing} = 0$ ;

else,  $r_k^{smoothing} = s, s < 0$ . The schematics of  $r_k^{cost}$  and  $r_k^{smoothing}$  are illustrated in Figure 4. The final reward at the  $k^{th}$  step is obtained with the sum of the above rewards:

$$r_k = r_k^{arrival} + r_k^{hazard} + r_k^{approach} + r_k^{cost} + r_k^{smoothing}. \tag{13}$$



**Figure 4.** The schematic of the reward for the energy cost and the reward for path smoothing: (a) reward for energy cost and (b) reward for path smoothing.

### 5.2. Planning Application

This sub-section introduces the application of our hierarchical framework to the SP-ResNet planner for path planning according to the specific operation flow, which is presented in Table 2 in brief.

- Task initialization: The start position  $(sx, sy)$ , the target position  $(tx, ty)$ , the original DEM for planning, and the map resolution  $res$  are given;
- Feature map acquisition: The feature map  $F$  is constructed using traversability analysis;
- Deformation: The original feature map is deformed into  $B^0$ , and the transformation ratio can be determined with  $K = M/N$ ;
- Initial rough planning: The initial rough path on the size-reduced map is solved with SI 0, according to Equation (6);
- Guide node determination: Along the initial path, a series of guide nodes is determined by Equation (7);
- Precise planning: Precise block planning based on the guide nodes is conducted. First, the blocks are determined according to Equation (8); then, SI  $j$  is called to plan a fine path on  $B^j$  until  $j = J - 1$ . The input of SI  $j$  can be obtained with:

$$\begin{cases} sx^{j*} = gx^j - px^j + N/2 \\ sy^{j*} = gy^j - py^j + N/2 \\ tx^{j*} = gx^{j+1} - px^j + N/2 \\ ty^{j*} = gy^{j+1} - py^j + N/2 \end{cases} \tag{14}$$

**Table 2.** The application flow of the learning-based planning method.

Application Flow of Path Planning.	
a.	<b>Task initialization:</b> $DEM, (sx, sy), (gx, gy), res$
b.	<b>Feature map acquisition:</b> $DEM \rightarrow F$
<b>Phase 1: SI 0</b>	
c.	<b>Deformation:</b> $F \rightarrow B^0, sx^{0*}, sy^{0*}, gx^{0*}, gy^{0*}, K = M/N$
d.	<b>Initial rough planning:</b> $\psi^0 = KSI(B^*, sx^{0*}, sy^{0*}, gx^{0*}, gy^{0*})$
<b>Phase 2: SI <math>j, j = 1, 2, 3, \dots</math></b>	
e.	<b>Guide node determination:</b> $(gx^j, gy^j)$
f.	<b>Precise planning:</b>
f.1.	Block index initialization: $j = 1$

Table 2. Cont.

Application Flow of Path Planning.	
f.2.	Block input: $\mathbf{B}^j, sx^{j*}, sy^{j*}, gx^{j*}, gy^{j*}$
f.3.	If $j < J - 1$ , $\psi^j = \mathbf{SI}(\mathbf{B}^j, sx^{j*}, sy^{j*}, gx^{j*}, gy^{j*})$ , $j = j + 1$ . Else, go to Step g.
f.4.	Back-calculation: $\psi^j$
g.	Path splicing.

After this, the path on the block should be back-calculated into the original F coordinate system with Equation (8);

- Path splicing: The segmented paths  $\psi^j$ . are spliced in sequence to form the final path on the original map, which can be expressed as follows:

$$\psi = \{ \psi^j | j = 1, 2, \dots, J - 1 \}. \tag{15}$$

## 6. Numerical Results and Analysis

### 6.1. Planner Training and Evaluation

#### 6.1.1. Training Settings

Without loss of generality, an SP-ResNet planner is instantiated by  $N = 100$ . and  $X = 10$ , which means that the planner plans on the  $100 \times 100$  block and has a  $10 \times 10$  input for the local branch. To meet the application of block iteration, the planner needs to adapt to various scaled obstacles, although the block size is unified. The planner is trained on a map collection containing 10,000 frames of  $100 \times 100$  blocks, each of which contains obstacles with random quantities and various scales. Figure 5 illustrates the constitution of the training map collection based on the simulated DEMs of craters and highlands.

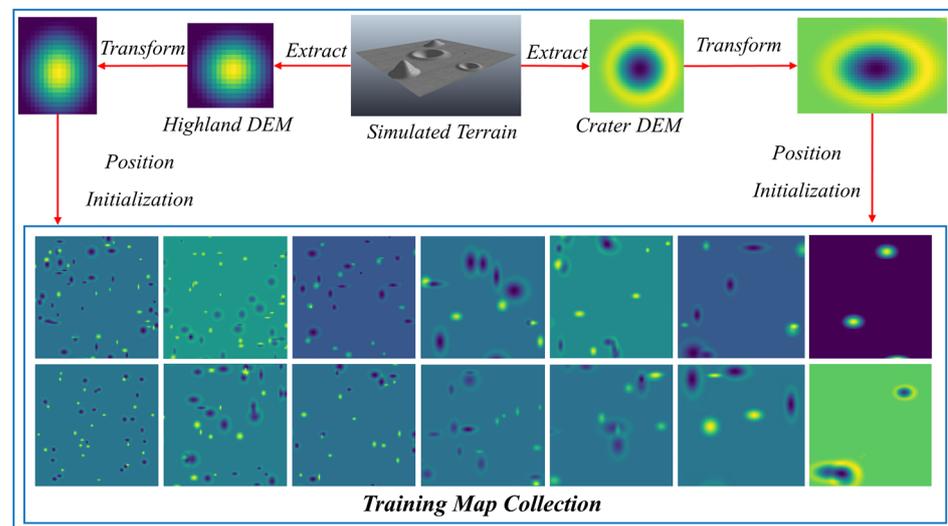


Figure 5. The training map collection.

At the beginning of each training episode, the start position and the target position are randomly initialized. Experiences in the form of  $[s_k, a_k, r_k, s_{k+1}]$  are collected according to Equation (9) for each step, and the loss function in Equation (11) is optimized by an Adam optimizer. Finally, we managed to train the planner with a final planning success rate of 85% for the map collection (over 20,000 episodes for about 48 h), as shown by the training curves in Figure 6.

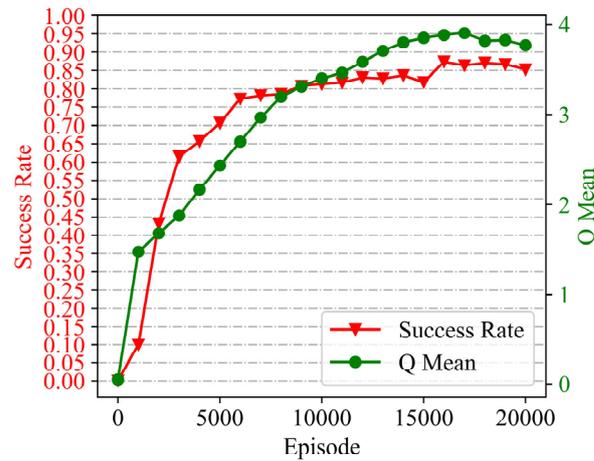


Figure 6. Training curves for the block planner.

6.1.2. Visualized Evaluation of the Block Planner

(1) Value map

A value map displaying the state value of the pixels all over the map in color mapping is an effective way to visually evaluate the learning-based planning methods [19,21]. With the trained SP-ResNet, the value maps in three cases are plotted, as shown in Figure 7a, including a map with flat terrain, a local perspective map with large-scale obstacles, and a global perspective map with small-scale obstacles. It should be noted that the SP-ResNet planner can clearly distinguish between the obstacles, safety areas, and target positions in all three cases.

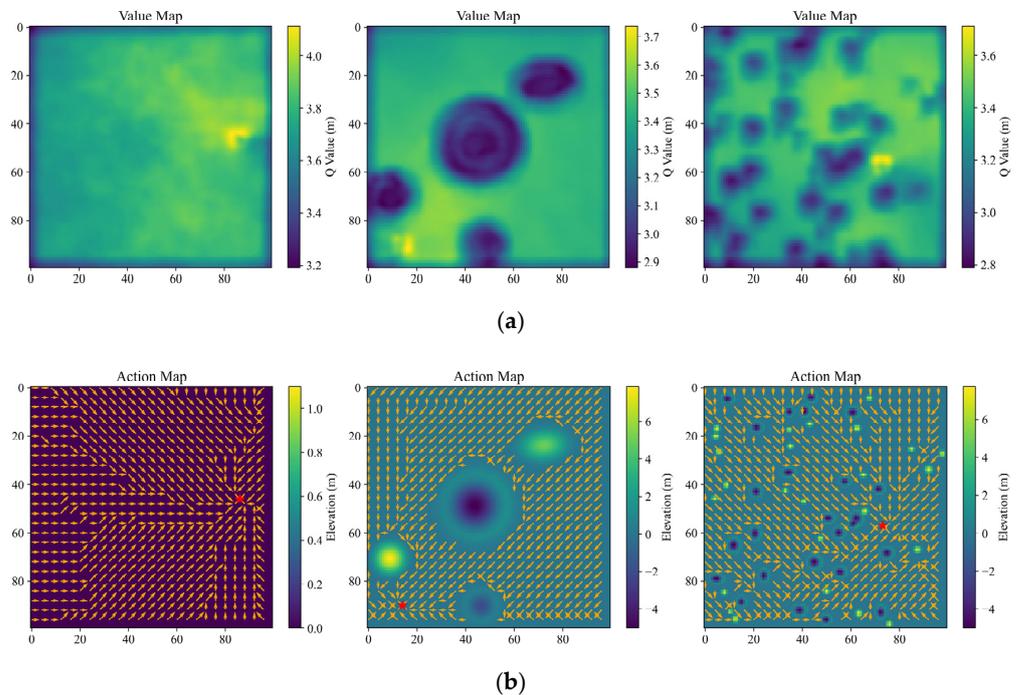


Figure 7. Visualized evaluation of the block planner: (a) value map, (b) action map.

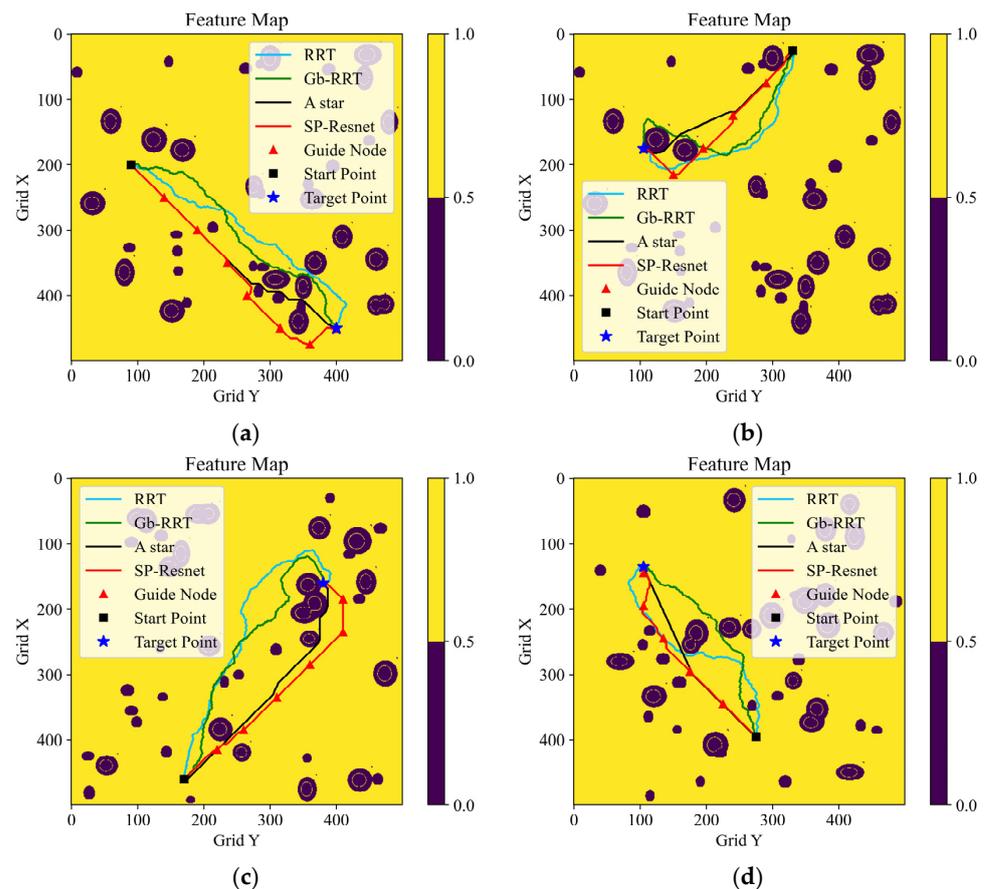
(2) Action map

Furthermore, a type of action map is drawn to directly look over the planned motion all around the original map. On the action map, for a given target position, the best motion direction at each pixel is planned and marked by arrows. Figure 7b depicts the action maps of the previous three cases. The figure shows that the SP-ResNet can identify the

optimal motion direction at almost all traversable positions, though for some minor areas, the planned results are suboptimal, indicating that the training is effective and reliable. It should be noted that the original maps for visualized evaluation in the three cases are beyond the training collection. With the end-to-end mechanism, the value map and the action map can be solved by one instance of network reasoning in a batch.

### 6.2. Comparative Analysis with Baselines

This section compares the learning-based method with the baselines, namely A\*, RRT, and Gb-RRT, for the metrics of planning time consumption and path length. A\* has been provided with complete theoretical proofs to find the optimal path, if it exists [8], which serves as the basic basis for the comparative analysis. Gb-RRT is an efficient method based on the heuristic mechanism of goal-biasing growth, which greatly improves the planning speed of RRT [18]. The planning methods involved are tested on several  $500 \times 500 \text{ m}^2$  maps. The SP-ResNet is applied following the hierarchical planning framework. A\* adopts an action space of eight motions toward eight directions, which is the same as the SP-ResNet planner. The RRT and Gb-RRT methods take a constant step size of one grid (1.0 m), and the Gb-RRT adopts a classical target biasing probability of 0.05. Figure 8 depicts the results of four planning tests, in which the planning methods are distinguished by color. Intuitively, the quality of the path planned with our method is close to the quality of the A\* algorithm, and is apparently better than that of the RRT series algorithms.



**Figure 8.** Comparative tests with the baselines on  $500 \times 500 \text{ m}^2$  maps: (a) task 1, (b) task 2, (c) task 3, and (d) task 4.

The planning time costs and the final path lengths of the four methods are presented in Table 3. It can be seen that our learning-based method performs the best in terms of the planning speed among the baselines, whose planning time consumptions are less than half of the A\*. Moreover, longitudinal comparisons demonstrate that the longer the path is,

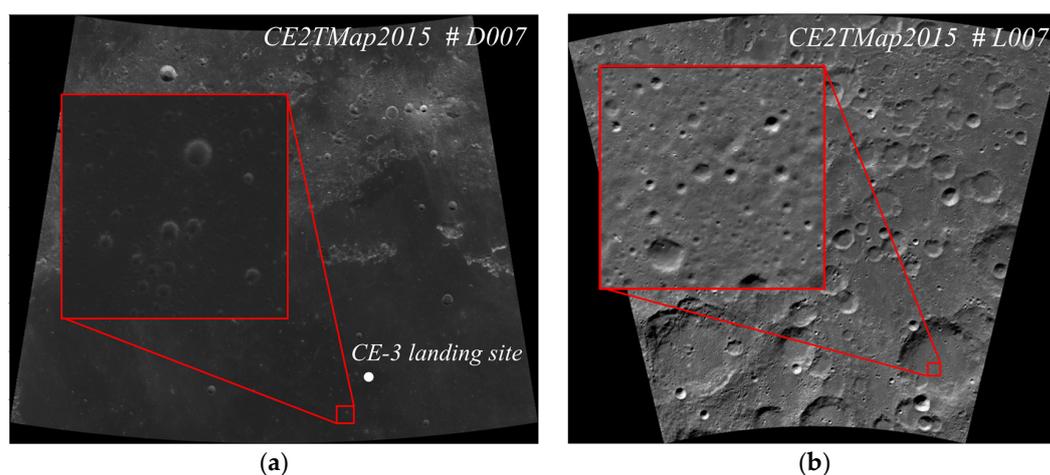
the greater the advantage of computational efficiency our end-to-end method has. As we expected, however, the final paths generated by the learning-based method are always partly greater than those by A\*. Due to the strong randomness of the RRT series algorithm, the time costs and the path lengths of RRT and Gb-RRT are statistically averaged over 20 repeated planning instances. It can be noted that the Gb-RRT's planning speed is very unstable, although on some occasions it can nearly match that of our method. In addition, the paths created by the RRT series algorithm are significantly longer than those of the end-to-end algorithm. It can be concluded that the method based on the hierarchical planning framework and the SP-ResNet planner has the advantages of a rapid planning speed and robust stability, which makes it better suited for scenarios with long-range maneuvers and complex environments. The above planning tests are run on a platform of Intel Core i9 with a CPU @2.40 GHz (32 GB) and NVIDIA GeForce RTX 2080 Super with Max-Q Design (8G).

**Table 3.** Comparative analysis of path planning tasks on  $500 \times 500 \text{ m}^2$  maps.

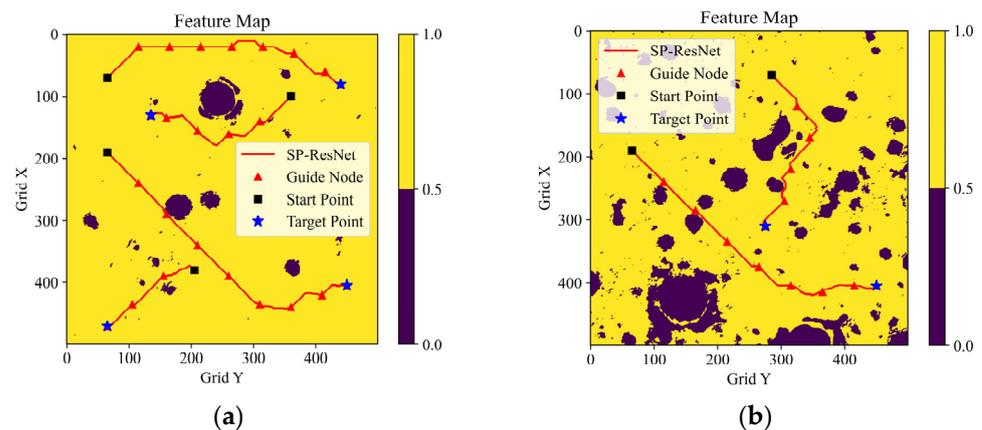
Planning Tasks	Metrics	A*	RRT	Gb-RRT	SP-ResNet
Task 1	Time cost/s	8.93	42.38	5.16	2.87
	Path length/m	413.55	512.50	525.10	473.27
Task 2	Time cost/s	4.71	62.11	3.64	2.18
	Path length/m	291.61	564.00	383.00	340.36
Task 3	Time cost/s	11.87	36.74	11.55	2.79
	Path length/m	395.54	539.10	529.40	430.28
Task 4	Time cost/s	6.62	36.80	1.03	2.30
	Path length/m	333.65	530.90	425.80	348.98

### 6.3. Tests with CE2TMap2015

The feasibility and the effect of our planning method were evaluated using the real lunar terrain data in CE2TMap2015 [5]. Two  $20 \text{ m/pixel}$  areas were selected as the test maps, as shown in Figure 9. For each of these maps, several path planning tests were conducted to validate our SP-ResNet planner and the hierarchical planning framework, as shown in Figure 10. Map 1 is a  $10 \times 10 \text{ km}^2$  area in the east of the Sinus Iridum area that is relatively flat overall, and there are a few craters. Map 2 is a  $10 \times 10 \text{ km}^2$  area at the bottom of a crater near the south pole that is covered by dense craters and highlands.

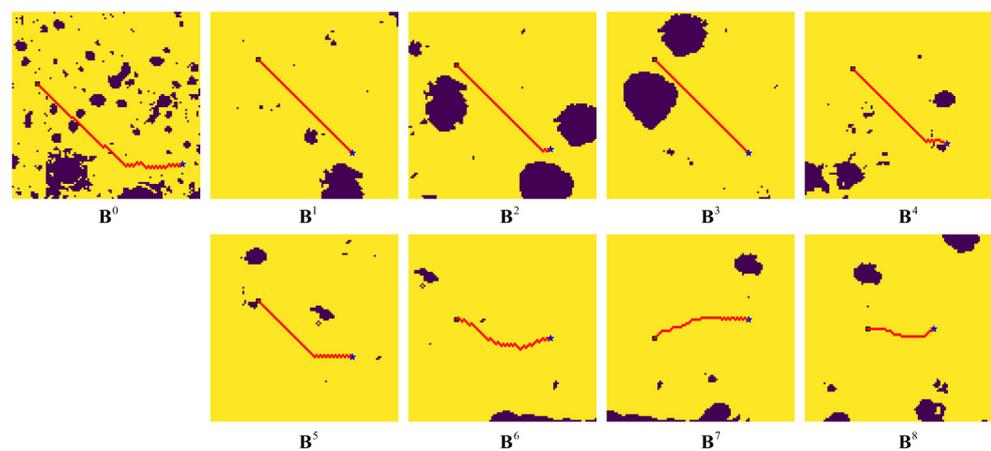


**Figure 9.** Real lunar DEMs from the CE2TMap2015. (a) Map 1: a  $10 \times 10 \text{ km}^2$  area in the east of the Sinus Iridum area on the Moon. (b) Map 2: a  $10 \times 10 \text{ km}^2$  area at the bottom of a crater near the south pole.



**Figure 10.** Path planning tests on real lunar terrains: (a) planning results with Map 1 and (b) planning results with Map 2.

Figure 11 depicts the results planned by blocks during the implementation process of a planning task on Map 2, in which  $B^0$  shows the initial rough path on the size-reduced map, and  $B^1$ – $B^8$  shows the path planning results for the blocks along the initial path. In the visualized results in Figures 10 and 11, the blue stars indicate the start points, the black cube indicates the target points, and the guide points are marked by red triangles.



**Figure 11.** The planned paths by blocks for one of the planning tasks on Map 2.

It can be seen from the above test results that the SP-ResNet planner performs well for the real lunar natural terrain, whether it is deformed from an original feature map with small-scale obstacles or from a precise block with large-scale obstacles, though the planner is trained using a map collection of simulated terrains. It can also be concluded that the hierarchical planning framework is effective for long-range path planning on lunar surfaces.

## 7. Conclusions

It is of great significance to improve the computational efficiency of planetary roving path planning, especially for the cases of long-range maneuvers or obstacle constraints that are time-varied, such as illumination. This research is devoted to investigating a novel learning-based global path planning method that can plan a safe path directly using a binary feature map in a fast and stable way. A binary feature map is constructed based on topography and illumination traversability analyses. A hierarchical planning framework containing step iteration and block iteration is designed that not only breaks through the defect of the poor adaptability of ANN-based methods, but also improves the planning speed. A step planner is constituted based on DRL, which determines features and plans

paths through a double-branch ResNet. The step planner is trained with a map collection based on simulated terrains, achieving a final planning success rate of 85%. Then, both a value map and an action map are generated to validate the step planner's ability of feature abstracting and path planning. Comparative analyses with A\*, RRT, and Gb-RRT for long-range planning tasks demonstrate the prominent planning speed and stability of our method. Finally, the proposed learning-based planning system is verified to be effective on real lunar terrains using CE2TMap2015.

The study is an important attempt to improve the path planning efficiency in long-range extraterrestrial roving based on a learning method. The performance of the involved planning system may be further improved in several ways. In fact, the reward and sampling mechanisms need further investigation to improve the optimality of the SP-ResNet planner, which seems to still have a certain gap from A\*. In addition, DRL algorithms for continuous action space can be introduced into the planner constitution to produce smoother paths.

**Author Contributions:** Conceptualization, Y.Z. and R.H.; methodology, Y.Z. and R.H.; numerical analysis, R.H.; investigation, Y.Z. and R.H.; writing—original draft preparation, R.H.; writing—review and editing, Y.Z. and R.H. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work is supported by Huzhou Institute of Zhejiang University under the Huzhou Distinguished Scholar Program (ZJHI—KY0016).

**Data Availability Statement:** Not applicable.

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

## References

1. Qunzhi, L.I.; Jia, Y.; Peng, S.; Han, L. Top Design and Implementation of the Lunar Rover Mission Planning. *J. Deep. Space Explor.* **2017**, *4*, 58–65.
2. Wang, Y.; Wan, W.; Gou, S.; Peng, M.; Liu, Z.; Di, K.; Li, L.; Yu, T.; Wang, J.; Cheng, X. Vision-based decision support for rover path planning in the chang'e-4 mission. *Remote Sens.* **2020**, *12*, 624. [\[CrossRef\]](#)
3. Smith, M.; Craig, D.; Herrmann, N.; Mahoney, E.; Krezel, J.; McIntyre, N.; Goodliff, K. The Artemis Program: An Overview of NASA's Activities to Return Humans to the Moon. In Proceedings of the IEEE Aerospace Conference Proceedings, Big Sky, MT, USA, 7–14 March 2020. [\[CrossRef\]](#)
4. Elliott, J.O.; Sherwood, B.; Austin, A.; Smith, M.; Polit-Casillas, R.; Howe, A.S.; Voecks, G.; Colaprete, A.; Metzger, P.; Zacny, K.; et al. ISRU in support of an architecture for a self-sustained lunar base. In Proceedings of the International Astronautical Congress, IAC, Washington, DC, USA, 21–25 October 2019.
5. Li, C.; Liu, J.; Ren, X.; Yan, W.; Zuo, W.; Mu, L.; Zhang, H.; Su, Y.; Wen, W.; Tan, X.; et al. Lunar Global High-precision Terrain Reconstruction Based on Chang'e-2 Stereo Images. *Wuhan Daxue Xuebao/Geomat. Inf. Sci. Wuhan Univ.* **2018**, *43*, 485–495. [\[CrossRef\]](#)
6. Xin, X.; Liu, B.; Di, K.; Yue, Z.; Gou, S. Geometric quality assessment of chang'E-2 global DEM product. *Remote Sens.* **2020**, *12*, 526. [\[CrossRef\]](#)
7. McEwen, A.S.; Eliason, E.M.; Bergstrom, J.W.; Bridges, N.T.; Hansen, C.J.; Delamere, W.A.; Grant, J.A.; Gulick, V.C.; Herkenhoff, K.E.; Keszthelyi, L.; et al. Mars reconnaissance orbiter's high resolution imaging science experiment (HiRISE). *J. Geophys. Res. E Planets* **2007**, *112*. [\[CrossRef\]](#)
8. Hart, P.E.; Nilsson, N.J.; Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [\[CrossRef\]](#)
9. Weiren, W.U.; Zhou, J.L.; Wang, B.F.; Liu, C. Key technologies in the teleoperation of Chang'E-3 "Jade Rabbit" rover. *Sci. Sin. Inf.* **2014**, *44*, 425–440.
10. Aoude, G.S.; How, J.P.; Garcia, I.M. Two-stage path planning approach for solving multiple spacecraft reconfiguration maneuvers. *J. Astronaut Sci.* **2008**, *56*, 515–544. [\[CrossRef\]](#)
11. Karaman, S.; Frazzoli, E. Sampling-based Algorithms for Optimal Motion Planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [\[CrossRef\]](#)
12. Zhu, S.; Zhu, W.; Zhang, X.; Cao, T. Path planning of lunar robot based on dynamic adaptive ant colony algorithm and obstacle avoidance. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 1729881419898979. [\[CrossRef\]](#)
13. Lanfeng, Z.; Lina, Y.; Hua, F. Lunar Rover Path Planning Based on Comprehensive Genetic Algorithm Based on Slip Prediction. In Proceedings of the 3rd International Conference on Artificial Intelligence, Automation and Control Technologies (AIAC), Xi'an, China, 25–27 April 2019.
14. Wang, J.; Chi, W.; Li, C.; Wang, C.; Meng, M.Q.H. Neural RRT\*: Learning-Based Optimal Path Planning. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 1748–1758. [\[CrossRef\]](#)

15. Tamar, A.; Wu, Y.; Thomas, G.; Levine, S.; Abbeel, P. Value iteration networks. In Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS), Barcelona, Spain, 5–10 December 2016.
16. Pflueger, M.; Agha, A.; Sukhatme, G.S. Rover-IRL: Inverse reinforcement learning with soft value iteration networks for planetary rover path planning. *IEEE Robot. Autom. Lett.* **2019**, *4*, 1387–1394. [[CrossRef](#)]
17. Zhang, J.; Xia, Y.; Shen, G. A novel learning-based global path planning algorithm for planetary rovers. *Neurocomputing* **2019**, *361*, 69–76. [[CrossRef](#)]
18. Urmson, C.; Simmons, R. Approaches for Heuristically Biasing RRT Growth. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Las Vegas, NV, USA, 27–31 October 2003; Volume 2.
19. Bai, J.H.; Oh, Y.J. Global Path Planning of Lunar Rover Under Static and Dynamic Constraints. *Int. J. Aeronaut. Space Sci.* **2020**, *21*, 1105–1113. [[CrossRef](#)]
20. Yu, X.; Huang, Q.; Wang, P.; Guo, J. Comprehensive global path planning for lunar rovers. In Proceedings of the 2020 3rd International Conference on Unmanned Systems, ICUS, Harbin, China, 27–28 November 2020.
21. Yu, X.; Guo, J.; Zhao, Y.; Yan, P. Fast and safe path planning for lunar rovers. *Hangkong Xuebao/Acta Aeronaut. Astronaut. Sin.* **2021**, *42*, 524153. [[CrossRef](#)]
22. Sutoh, M.; Otsuki, M.; Wakabayashi, S.; Hoshino, T.; Hashimoto, T. The right path: Comprehensive path planning for lunar exploration rovers. *IEEE Robot. Autom. Mag.* **2015**, *22*, 22–23. [[CrossRef](#)]
23. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. In Proceedings of the 27th Conference on Neural Information Processing Systems (NIPS 2013), Lake Tahoe, NV, USA, 5–10 December 2013.
24. 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](#)] [[PubMed](#)]
25. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016), Las Vegas, NV, USA, 26 June–1 July 2016.