Sakamoto and Kubota *ROBOMECH Journal* (2022) 9:4
https://doi.org/10.1186/s40648-022-00219-7

ROBOMECH Journal

RESEARCH ARTICLE

Open Access

Hopping path planning in uncertain environments for planetary explorations

Kosuke Sakamoto[1*] and Takashi Kubota[2]

Abstract

Hopping robots, called hoppers, are expected to move on rough terrains, such as disaster areas or planetary environments. The uncertainties of the hopping locomotion in such environments are high, making path planning algorithms essential to traverse these uncertain environments. Planetary surface exploration requires to generate a path which minimises the risk of failure and maximises the information around the hopper. This paper newly proposes a hopping path planning algorithm for rough terrains locomotion. The proposed algorithm takes into account the motion uncertainties using Markov decision processes (MDPs), and generates paths corresponding to the terrain conditions, or the mission requirements, or both. The simulation results show the effectiveness of the proposed route planning scheme in three cases as the rough terrain, sandy and hard ground environment, and non-smooth borders.

Keywords: Planetary exploration, Path planning, Hopping robot

Introduction

In recent years, hopping robots (hoppers) have received a lot of attention, because they are expected to play an active role in disaster areas [1], celestial bodies [2–5], etc. In September 2019, the MINERVA-II robot, the hopper developed by JAXA/ISAS, succeeded in landing on the surface of the asteroid “Ryugu”, locomoting, and taking the photos as shown in Fig. 1 [6]. This achievement indicates that the planetary surface exploration by hoppers is becoming active increasingly. For example, various environments, such as a Recurring Slope Lineae (RSL) on Mars [7], are expected to be investigated by hoppers. However, there are many challenges to carry out actual planetary surface explorations by a hopper/hoppers. One of the challenges is the path and motion planning problems. The detailed conditions of the planetary surface cannot be known before the robot has arrived on site and explored the celestial body. The environments also have uncertainties of locomotion. In addition, planetary surfaces are almost completely covered with granular media, called regolith. The sandy terrains might cause the robot to get stuck. Therefore, hopping path/motion planning algorithms are essential in order to investigate such environments using hoppers.

The contribution of this paper is to propose a hopping path planning algorithm to traverse uncertain environments. This paper presents the three main results to generate the paths:

- on rough terrain depending on the mission requirements.
- on heterogeneous terrains.
- in non-smooth environment.

It is difficult to follow a path accurately in uncertain environments. The proposed algorithm can calculate the optimized action in each state using markov decision processes (MDPs). One of the advantages of MDPs is that you do not have to re-calculate paths when a hopper fails to follow a path. It is necessary to generate paths in a 3D environment, because hopping motion is spatial. However, the calculation cost is very high in the fully 3D environment. The path planning algorithm needs

*Correspondence: ksakamoto605@g.chuo-u.ac.jp
[1] Department of Electrical Electronic and Communication Engineering, Faculty of Science and Engineering, Chuo University, Bunkyo-ku, 112-8551 Tokyo, Japan
Full list of author information is available at the end of the article

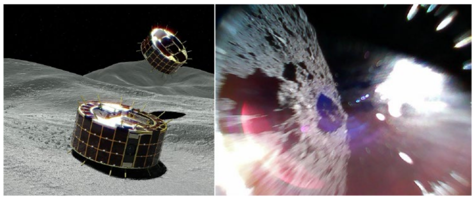


Fig. 1 MINERVA II robot and rendering the contribution. Left: the image of MINERVA II on asteroids. Right: the real photo that the MINERVA II robot took while hopping on asteroid *Ryugu* [6]

the constraint of the hopping trajectory for the collision check to reduce the calculation costs compared with the calculation in 3D. The proposed algorithm is classified as a 2.5D path planning. It is also needed to design the payoff function which depends on the environments. The proposed algorithm can generate the paths corresponding to the requirements of mission sequences by changing the parameters. This paper considers three cases: considering the safety, the information gain, and both. Although the safety is important for successful missions, a hopper needs to perceive the environment around itself. This is because the hopper, or a rover, doesn't know the details of the environment. The virtual environments are created and the path generations are tested in simulations. First, the paths and actions are calculated on the rough terrain, and then the differences between the above mentioned three cases are evaluated. One of the virtual environments includes the sandy places as a heterogeneous terrain. Sand decrease the hopping performance because of slipage, and may cause the hopper to get stuck. The differences in the results between the locomotion on the homogeneous terrain and the heterogeneous terrain are also observed. Finally, the proposed algorithm generates the path in the environment which includes the non-smooth parts, such as rocks, or steps. The hopper can get on such paths by hopping almost vertically.

The next section reviews related works on the topic. Then, the proposed algorithm for the hopping path planning on rough terrains is introduced, and the simulation results to evaluate the proposed algorithm are shown. The results section introduces the three environments in the simulations: the homogeneous rough terrain, the heterogeneous rough terrain, and the non-smooth environment. Finally, the last section summarises this paper and presents future work.

Related works

Path planning, or motion planning algorithms have been studied and improved by many researchers. Since our study focuses on 3D environments and field robotics, this section enumerates some conventional planning algorithms and their applications to 3D environments and natural terrains.

Sampling based algorithms

The probabilistic road map (PRM) [8] and the rapidly exploring random trees (RRTs) [9] are some of the most commonly used planning methods. Both algorithms sample many nodes in the configuration space (C-space) and then generate a path connecting nodes. In the C-space, the collision areas are expanded into the size of a robot around the obstacles. When sampling nodes, collision check is required. The difference between the PRM and RRTs is the way of generating a path. PRM uses a graph search algorithm to generate the path. RRTs connect nodes depending on the constraints.

One of the applications of PRM for 3D environments is presented in [10]. However, the computation time increases due to collision check. In addition, the calculation cost also depends on a graph search algorithm.

RRTs and its series are used more widely than PRM because of the applicability and simplicity. Although the result of only RRT is not an optimal path, the constraints can generate a semi-optimal path. One of the famous algorithm is RRT* [11], and the application in 3D environments is presented in [12]. RRT series generate an only single path. If a robot fail to follow the path, then you need to re-planning a path. This is because we think RRT-based path planning algorithms are not suited for the uncertain environments.

Geometric analytic approaches

As the name suggests, geometric analytic approaches use the geometric information of the environments. Interpolating curve algorithms are the often used in the field of path planning for wheeled vehicles in 2D environments (e.g., [13]). Voronoi diagram is also one of the most famous methods and has been applied for 3D environments [14]. Voronoi diagram divides the environments around specific points (most of the time, the obstacles in robotics). The divided regions generate topological connection; the distance from the edge to the obstacles is the same. To generate the shortest (or optimal) path, it is essential to use a graph search algorithm (which will be described in the next subsection) with voronoi diagram, because voronoi generates only a graph.

Graph search algorithms

Graph theory is an important basic theory in robotics. Robots use a grid map constantly and a path on a grid map is often generated using graph search algorithms. Dijkstra [15], A* [16], or D* [17] are the most used graph search algorithms even in 3D environments. Dijkstra's algorithm can generate the shortest path and is easy to implement for various environments, however the calculation time increases diverge dramatically the number of the nodes. A* and D* reduce the computation cost by using heuristics. This is why these algorithms are often implemented to real time applications. Graph searches are more effective to cooperate with other graph generators. However, one of the shortcomings of these algorithms is that it is difficult to apply them for uncertain environments. If the robots stray off the calculated path, it is required to re-calculate the path.

Model based algorithms

Unlike urban environments, locomotion on natural terrains includes a lot of uncertainties because of the interactions with the terrain surface. Model based algorithms reduce the uncertainty arising from terrains using interaction models. State lattice algorithm [18] is applied on rough terrains. The algorithm uses the grid of state, called state lattice. Based on the mobility model or the terrain interaction, the input parameters are optimized. The method often uses lookup tables or neural networks for estimating the parameters. Therefore, the validity of the results of the optimization depends on the lookup table or the used model. Terramechanics is the field of the terrain-mobility interactions and the application for path planning of wheeled robots has been studied in [19]. Although terramechanics can estimate the motion on sandy terrains, there are few conditions to which terramechanics can be applied because of the uncertainty of the prediction models. Recently, machine learning algorithms based model has been widely studied instead of terramechanics [20]. One of the advantages of machine learning is that we can create the models of various terrain conditions. One of the challenges of this field is to make computationally light on-line learning algorithms.

Uncertainty-aware planning

It is hard to detect the causes of uncertainties, and hence uncertainties are formulated probabilistically. Chance constraint path planning is one of the approaches to deal with the motion uncertainty. CC-RRT* [21] is the combined RRT* algorithm with the uncertainty of the collision as constraints. Particle filter RRT [22] formulates the increase of the uncertainty of the locomotion as a particle filter. These algorithms generate the path which increases the uncertainty as little as possible. However the easiness of a path following or the re-planning of the path are not considered.

The Markov Decision Processes (MDPs) based algorithms (e.g., [23, 24]) calculate the actions in all states. Therefore, it is not essential to re-plan the paths. In particular, MDPs is effective for large uncertain environments. This study uses MDPs as a basic theory. Hockman and Pavone propose the hopping motion planning on asteroids using reinforcement learning [25]. The difference with this study is that they make the uncertainty model on asteroids using reinforcement learning, and minimise the noise of the hopping trajectories. This study focuses on not only asteroids, but also planets, or satellites which have solid surfaces, such as Moon or Mars. In addition, our approach can generate paths corresponding to the requirements in each mission sequence, and calculate the optimal action in all states (details in section). The hopper resumes moving immediately after a failure of the landing on the expected point without re-calculating.

Qualitative comparison

Table 1 shows the comparison table of the path planning methods. The five items, validity, adaptivity, optimality, calculation cost, and uncertainty are evaluated according to the four-grade system. The advantage of the PRM or RRTs is a light calculation cost. Although the RRT* can generate the suboptimal paths and CC-RRT* can generate the uncertainty aware paths, the performances are slightly less than the others. Voronoi can generate the optimal paths and mitigate the calculation cost. However, the paths are not adaptive and do not include uncertainty.

Methods

This section presents the assumptions of environments and the conditions of the hopper. Then, the proposed algorithm is described. This study uses the MDP method. The reasons are that the hopping is a discrete motion, and hence the uncertainty is relatively large. In addition, it is difficult to control the attitude and trajectory while hopping. Therefore, the best action should be pre-calculated in all states.

Assumptions

The environment which is used in this paper causes the motion uncertainties because of the roughness of the

Table 1 The comparison of path planning algorithms. The five items, validity, adaptivity, optimality, calculation cost, and uncertainty are evaluated according to the four-grade system

	PRM, RRTs	Voronoi	Dijkstra, A*, D*	State Lattice	MDPs
Validity	○	○	○	◎	○
Adaptivity	○	×	△	◎	◎
Optimality	△	◎	◎	△	△
Calculation cost	◎	◎	×	△	×
Uncertainty	△	×	×	△	◎

terrain. Moreover, a part of the surface that is covered with granular media, such as regolith, causes slipage. It decreases the performance of the hopper low and increases the motion uncertainties. This study assumes that the uncertainty of the landing point is probabilistically expressed. Figure 2 shows the image of the uncertainty of the hopping.

Hopper Conditions

The shape of the hopper is a sphere of 15 [cm] radius. The hopper can hop any horizontal directions (360°) and some vertical directions (45–90°). The condition is the same as our previous work [26]. The hopper is equipped with springs which generate the hopping force, and stereo cameras to perceive the environments. This study assumes that the hopper can know its position and attitude using an arbitrary localization method. Therefore, this paper does not discuss localization and perception methods. This paper also does not focus on the mobility mechanisms of the hopper in details. The initial velocity v_0 is known when the hopping starts. This is because we can measure the initial elastic energy of springs and a mechanical loss, and then calculate the initial kinetic energy $E_{\text{kin}} = 1/2mv_0^2$.

Markov decision processes

This section describes the basic concept of the MDPs and the application. MDPs assume that robots can observe the state fully, i.e., the perceptual model $p(z|x)$, Where z and x denote the measurement and state, respectively, is deterministic. If we would like to treat fully probabilistic cases, which include the uncertainty of observations, the partially observable markov decision processes (POMDPs) is used. However, this paper does not focus on the POMDPs. The MDPs define the probabilistic action model $p(x_t|u_{t-1}, x_{t-1})$, where u denotes the action.

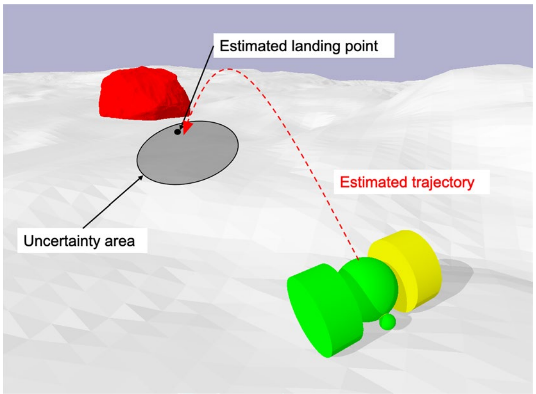


Fig. 2 The image of the hopping with uncertainty

The key technique of the MDPs is the way of designing the payoff function $r(x, u)$. The details of the $r(x, u)$ are described later. This study employs the value iteration method in order to calculate the best control policy. At step T, the control policy $\pi_T(x)$ is expressed as follows:

$$\pi_T(x) = \arg\max_u \left[r(x,u) + \int V_{T-1}(x')p(x'|u,x)dx' \right], \quad (1)$$

where $V_T(x)$ denotes the value function which is defined by:

$$V_T(x) = \gamma \max_u \left[r(x,u) + \int V_{T-1}(x')p(x'|u,x)dx' \right], \quad (2)$$

where γ denotes the discount factor. In the case of $T \to \infty$, Eq. (2) is known as the Bellman equation. Bellman equation induces the control policy to be optimal. Algorithm 1 shows the calculation of the value iteration. Once the $V(x)$ is obtained, the control policy is calculated by Eq. (1). This paper calculates the value iteration in all states.

Algorithm 1 Value iteration.

```
for $i = 1 \to N$ do
    $V(x_i) \leftarrow r_{\min}$
end for
while until $V(x_i)$ converge do
    for $i = 1 \to N$ do
        $V(x_i) \leftarrow \gamma \max_u \left[ r(x_i, u) + \sum_{j=1}^{N} V(x_j)p(x_j|u, x_i) \right]$
    end for
end while
return $V$
```

Payoff function

In general, payoff functions are defined numerically in MDPs, and designed by considering the above assumptions and hopping features. The proposed function is expressed as:

$$r(x,u) = w_1 S(x,u) + w_2 I(x,u) + P(x,u), \quad (3)$$

where $S(x, u)$, $I(x, u)$ and $P(x, u)$ denote the safety cost, the information gain, and the penalty, respectively. The w_1 and w_2 are the weight coefficients. The payoff function applies the min-max normalization to each term $S(x, u)$ and $I(x, u)$.

The safety of the rover operations depends on the interaction between the mobility and the environments, such as slope, obstacles, terrains and so on. The safety cost is proportional to the roughness. The reasons are that hoppers can ride on rocks, and get over steps. These indicate that hoppers can access more dangerous areas than wheeled rovers can access. In addition, the algorithm calculates the hopping trajectory in advance for collision check. If a hopper collide with an obstacle or a hill before reaching the highest point in the trajectory, the hopper is bounced off. Hence, the safety cost becomes very low in the case of a collision occurring. The hopping trajectory is calculated as follows:

$$z = -\frac{1}{4\alpha}(d - 2\alpha)^2 + \alpha, \tag{4}$$

where z and d denote the vertical direction and moving direction of the hopper, respectively. Note that $\alpha = v_0^2/2g\sin^2\eta$ is used to simplify the Eq. (4), where g and η denote the gravitational acceleration and the hopping angle, respectively. Moreover, the origin of the Eq.(4) is the initial position of the hopper. Therefore, the function of the safety cost is described as follows:

$$S(\boldsymbol{x}_p, u) := -f(u)\left|\nabla h(x_p, y_p)\right| \tag{5}$$

where $h(x_p, y_p)$ and $f(u)$ denote the height of the terrain at a point (x_p, y_p) and the switching function, respectively. The roughness on a point (x_p, y_p) of terrain is formulated as the magnitude of the gradient at that point. If the hopper chooses the action u_1, and a collision is detected, $f(u_1)$ returns an infinite value. Otherwise, $f(u_1)$ returns 1.

The information gain is also key to explore in the uncertain environments. When a robot traverses an unknown environment, it needs to perceive the environment around and map it in order to make a path or decide a next motion. The entropy $H_p(x)$ is often used as the expected information gain $E[-\log p(x)]$, where $p(x)$ denotes a probability distribution. The belief b is usually used as a $p(x)$. Exploring in planetary environments, a hopper can perceive the environments around the hopper by riding on a high place, such as a rock or a step. In this paper, the information gain is expressed simply as:

$$I(\boldsymbol{x}_p, u) := h(x_p, y_p). \tag{6}$$

This equation means that the higher the position of the hopper, the greater the information it can gather.

Simulation study

The following section describes the simulation study to validate the path planning algorithm expressed in the previous section. First, the assumptions of the simulations are introduced. Next, the results of several cases of simulations are shown and discussed changing the weight coefficient w_1 and w_2, and sandy surface or rigid surface.

Modelling assumptions

This simulation uses the artificial rough terrain shown in Fig. 3. This environment has regions of higher and lower elevation, indicated with a colorbar.

This environment is expressed as digital elevation map (DEM), of size 20[m] × 20[m]. Each state is a 1[m] × 1[m] cell. The terrain is expressed as superposition of 2D Gaussian distribution:

$$h(\boldsymbol{x}_p) = \sum_i \frac{1}{2\pi\sqrt{|\Sigma_i|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}_p - \boldsymbol{\mu}_i)^{\mathrm{T}}\Sigma_i^{-1}(\boldsymbol{x}_p - \boldsymbol{\mu}_i)\right), \tag{7}$$

where $\boldsymbol{x}_p = (x_p, y_p)$ denotes a 2D position. The mean vector $\boldsymbol{\mu}_i$ and the covariance matrix Σ_i are described as follows:

$$\boldsymbol{\mu}_i = (E_i[x], E_i[y]), \tag{8}$$

$$\Sigma_i = \begin{pmatrix} \sigma_{i,x}^2 & \sigma_{i,xy} \\ \sigma_{i,xy} & \sigma_{i,y}^2 \end{pmatrix}, \tag{9}$$

where $E_i[\cdot]$, $\sigma_{i,\cdot}$ and $\sigma_{i,xy}$ denote the expected value, standard deviation and covariance of the i-th gaussian distribution, respectively. This simulation used eight gaussian distribution: five with elevated regions (i.e., positive values) and three depressed regions (i.e., negative values). The means, standard deviations, covariances of the gaussian distributions are shown in Table 2.

These expected values indicate the extremal values. The standard deviations and covariances use identical values, which means the profiles of convex and concave features are the same. The start point is $(x_p, y_p) = (0, 0)$.

The paths are generated by MDPs, and hence each optimal actions chosen by the hopper is calculated

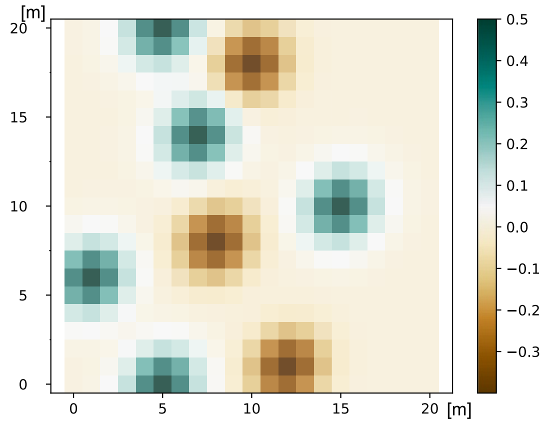


Fig. 3 The artificial terrain. The higher places are colored green, and the lower places are colored brown

Table 2 The parameters of the gaussian distributions

Parameters	Values							
	i = 1	**i = 2**	**i = 3**	**i = 4**	**i = 5**	**i = 6**	**i = 7**	**i = 8**
$E_i[x]$	1	5	5	7	8	10	12	15
$E_i[y]$	6	0	20	14	8	18	1	10
$\sigma_{i,x}$	2							
$\sigma_{i,y}$	2							
$\sigma_{i,xy}$	0							

Table 3 The action list

Action name	Directions			
Moving orientations	North	South	East	West
Turning orientations	Turn right		Turn left	
Vertical hop orientations	North	South	East	West

in all states. The actions used in the simulations are defined in Table 3.

In order to mitigate the calculation cost, the moving orientations of the hopper are restricted to four options, and the turning orientations are restricted to two. Moreover, two hopping angles are used: 45 and 80°. When a collision occurs, the hopper changes the hopping angle from 45 to 80°, and then re-calculates the trajectory. If the collision does not occur, the hopper choose the "Hop up" action to ride on the rock or hill, otherwise the hopper chooses the other action. This simulation employs the Martian gravitational acceleration $g = 3.72[\mathrm{m/s^2}]$. The value of the square of the initial velocity v_0^2 is constant to calculate the hopping distance $d' = 1.0[\mathrm{m}]$ and $h' = 0.25[\mathrm{m}]$ at the hopping angle $\eta = 45$ [deg] on flat terrains. In the case that the hopper moves on slopes and cannot reach to the next state (cell), the hopper continues to do the same action until arriving at the next state.

Results and discussion

The simulation conditions and the results are listed and discussed below in detail. The simulation environments are four cases: hard ground, hetero terrain (hard and sandy terrain), sandy terrain, and flat terrain with two obstacles. The start point is (0, 0), and the goal point is (18, 18), that are shown in "S" and "G" in Figs. 4, 5, 6, 7, 8 and 9.

This simulation used a laptop PC with Intel Core-i9, 8 cores, 2.4 [GHz].

Simulations on hard ground

This work try the three 11 of simulation by changing the weight coefficient in Eq. (3) on hard ground: $(w_1, w_2) = (0, 1), (0.1, 0.9), (0.2, 0.8), \ldots, (1, 0)$. The goal is $(x_p, y_p) = (18, 18)$, and the payoff is enough big (10) at the point. The uncertainties of the hopping maneuver are defined as below: the hopper

- can reach to the expected point with a $70\left(1 - \frac{\theta_{\mathrm{front}}}{(\pi/6)}\right)$% possibility
- may not change the state with a $70\frac{\theta_{\mathrm{front}}}{(\pi/6)}$% possibility
- may be off to the left with a $15\left(1 - \frac{\theta_{\mathrm{right}}}{(\pi/6)}\right)$% possibility or to the right with a $15\left(1 - \frac{\theta_{\mathrm{left}}}{(\pi/6)}\right)$% possibility

where $\theta_{\mathrm{direction}}$ denotes the slope angle to a direction. The reason why $\theta_{\mathrm{direction}}$ divided by $(\pi/6)$ is that the probabilities of the uncertainties of the motions are in inverse proportion to a slope angle. This simulation assumes the maximum angle of slope is 30 degrees, which the hopper may land at an expected point, and be off to the both sides. Note that if the $\theta_{\mathrm{direction}} \geq \pi/6$, the $1 - \frac{\theta_{\mathrm{front}}}{(\pi/6)} := 0$ in order the probability not to be negative values, and $\theta_{\mathrm{left}} = -\theta_{\mathrm{right}}$.

The simulation results are shown in from Figs. 4a to 5e. The arrows indicate the actions the hopper chooses at each state: blue, purple, red, and black arrows denote the moving direction to north, south, east, and west, respectively. The green lines show the path that start from the initial state. The computing time is about 0.79 [s]. In the case of $(w_1, w_2) = (0, 1) - (0.2, 0.8)$ (Fig. 4a–c), the hopper makes the path what traverse on places as high as possible to get the information about the environment. Figure 4d shows that the path considering both the information gain and the safety moves on relatively higher places at first. Then, coming near the goal, the hopper selects the path on the flat terrain.

Figures 4e–5e indicate that the path from the initial state is generated on the flat terrains. The hopper also selects the flattest route possible at the start. However, the actions in some states near the goal show that hopper should go through the slopes. It is assumed that the hopper gets more payoff to traverse on slopes than to make a detour in order to reach the goal. Simulation results

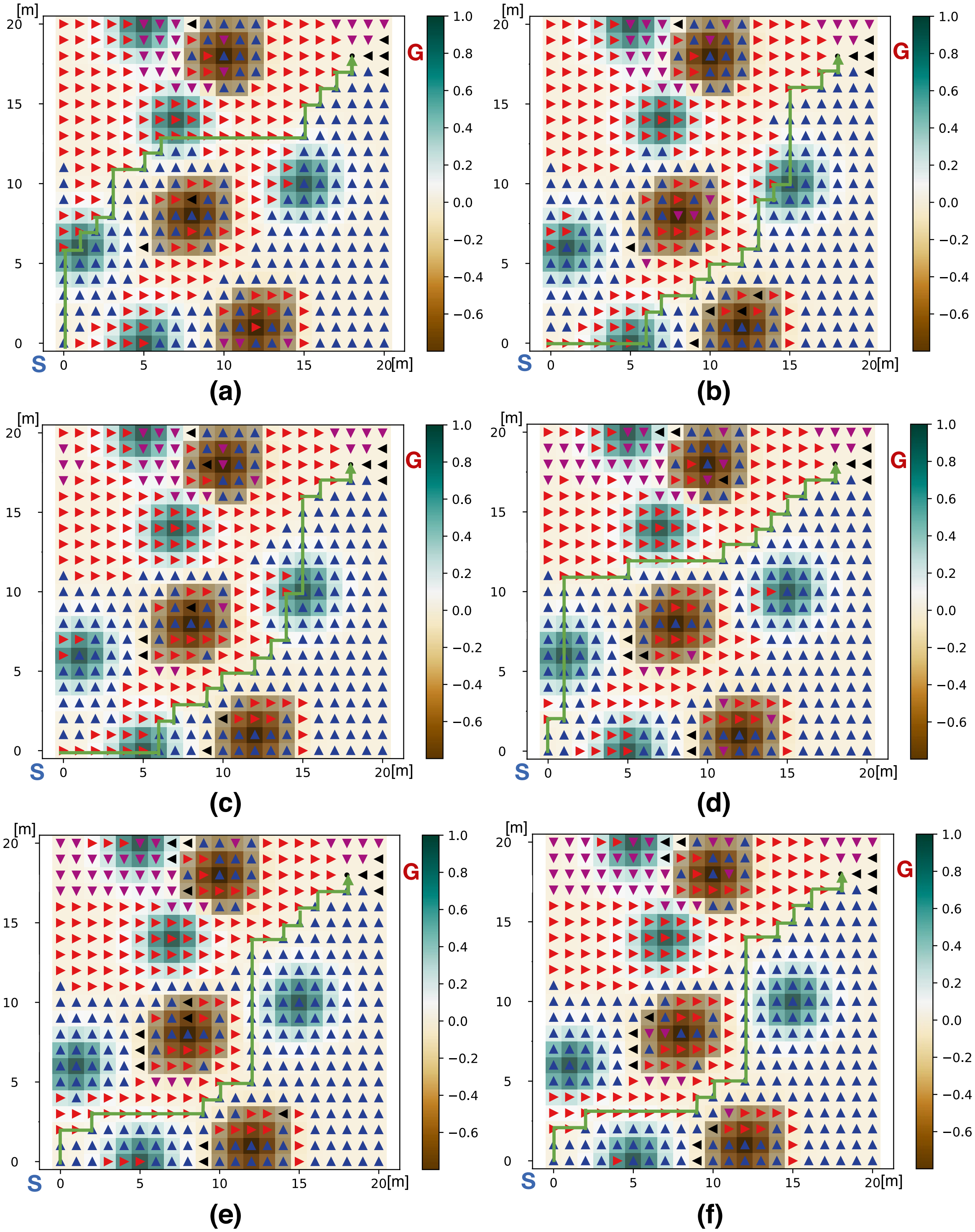


Fig. 4 The path planning results on hard ground (1); **a**: $(w_1, w_2) = (0, 1)$, **b**: $(w_1, w_2) = (0.1, 0.9)$, **c**: $(w_1, w_2) = (0.2, 0.8)$, **d**: $(w_1, w_2) = (0.3, 0.7)$, **e**: $(w_1, w_2) = (0.4, 0.6)$, **f**: $(w_1, w_2) = (0.5, 0.5)$

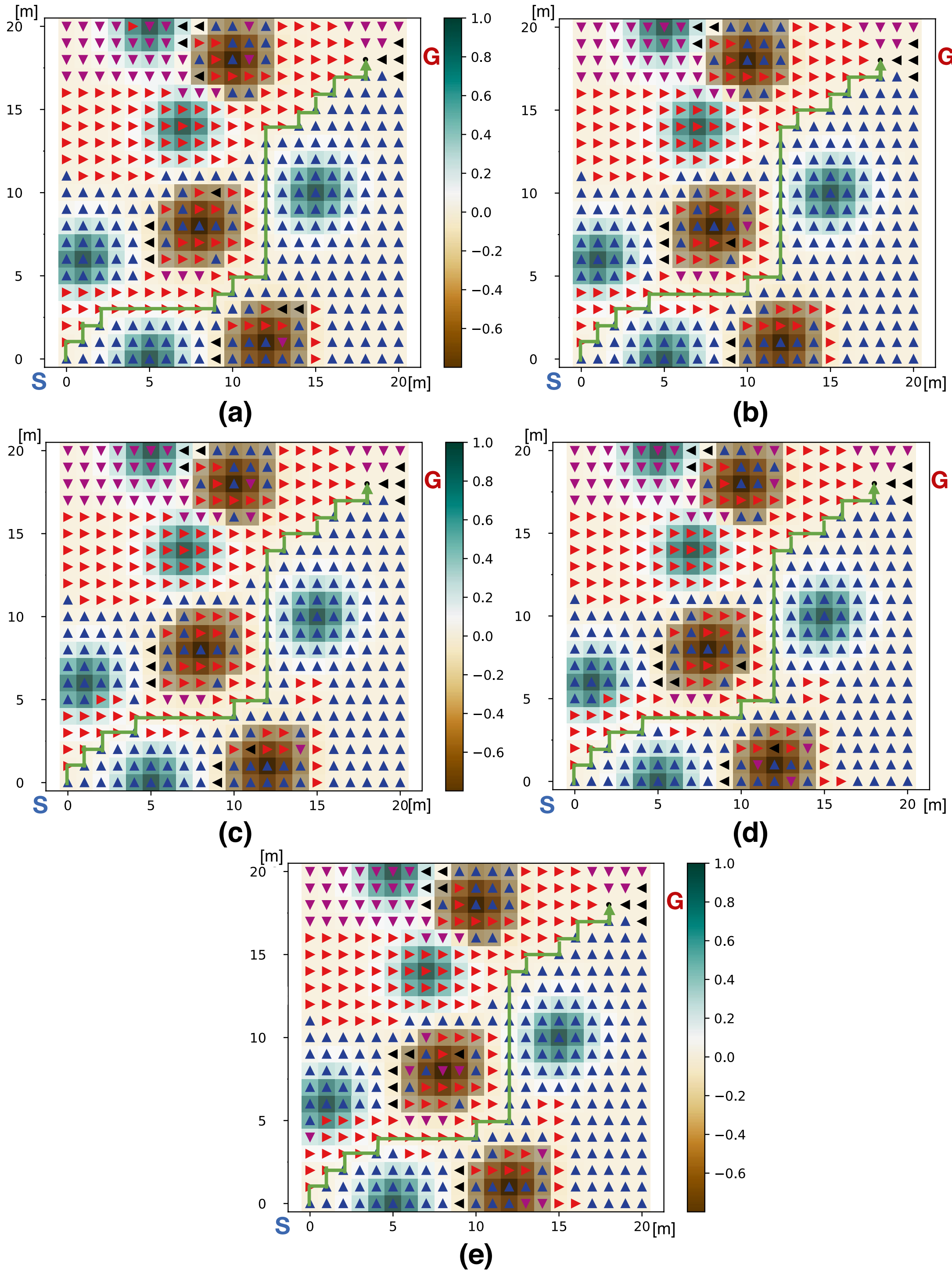
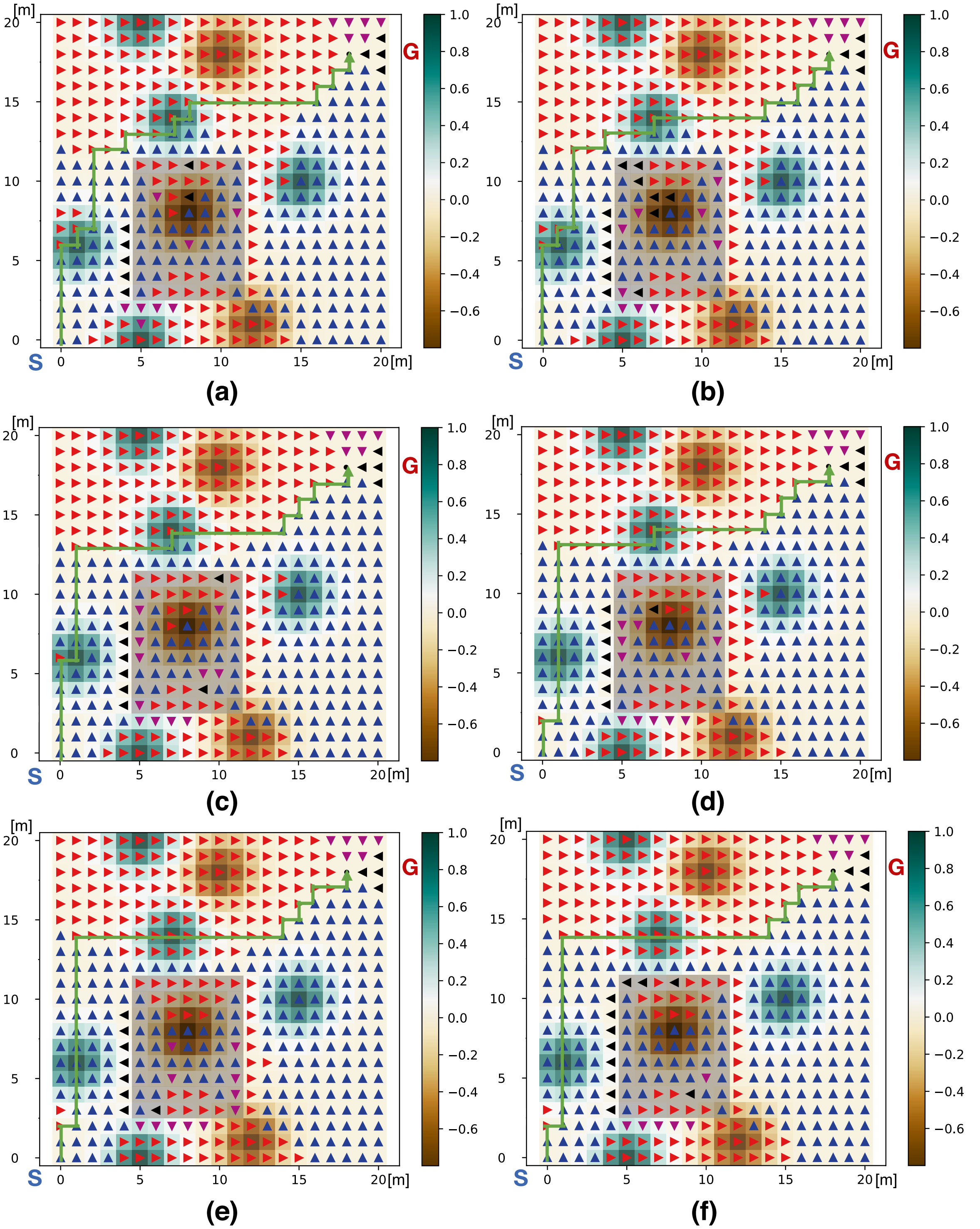


Fig. 5 The path planning results on hard gorund (2); **a**: $(w_1, w_2) = (0.6, 0.4)$, **b**: $(w_1, w_2) = (0.7, 0.3)$, **c**: $(w_1, w_2) = (0.8, 0.2)$, **d**: $(w_1, w_2) = (0.9, 0.1)$, **e**: $(w_1, w_2) = (1, 0)$

naturally depend on the conditions of environment or the hopper or both.

These results suggest the following observation:

Observation 1: the proposed algorithm can generate the various paths depending on the requirements: more safe, more challenging, or both safety and challenge.



(a)

(b)

(c)

(d)

(e)

(f)

Fig. 6 The path planning results on heterogeneous terrain (1); **a**: $(w_1, w_2) = (0, 1)$, **b**: $(w_1, w_2) = (0.1, 0.9)$, **c**: $(w_1, w_2) = (0.2, 0.8)$, **d**: $(w_1, w_2) = (0.3, 0.7)$, **e**: $(w_1, w_2) = (0.4, 0.6)$, **f**: $(w_1, w_2) = (0.5, 0.5)$

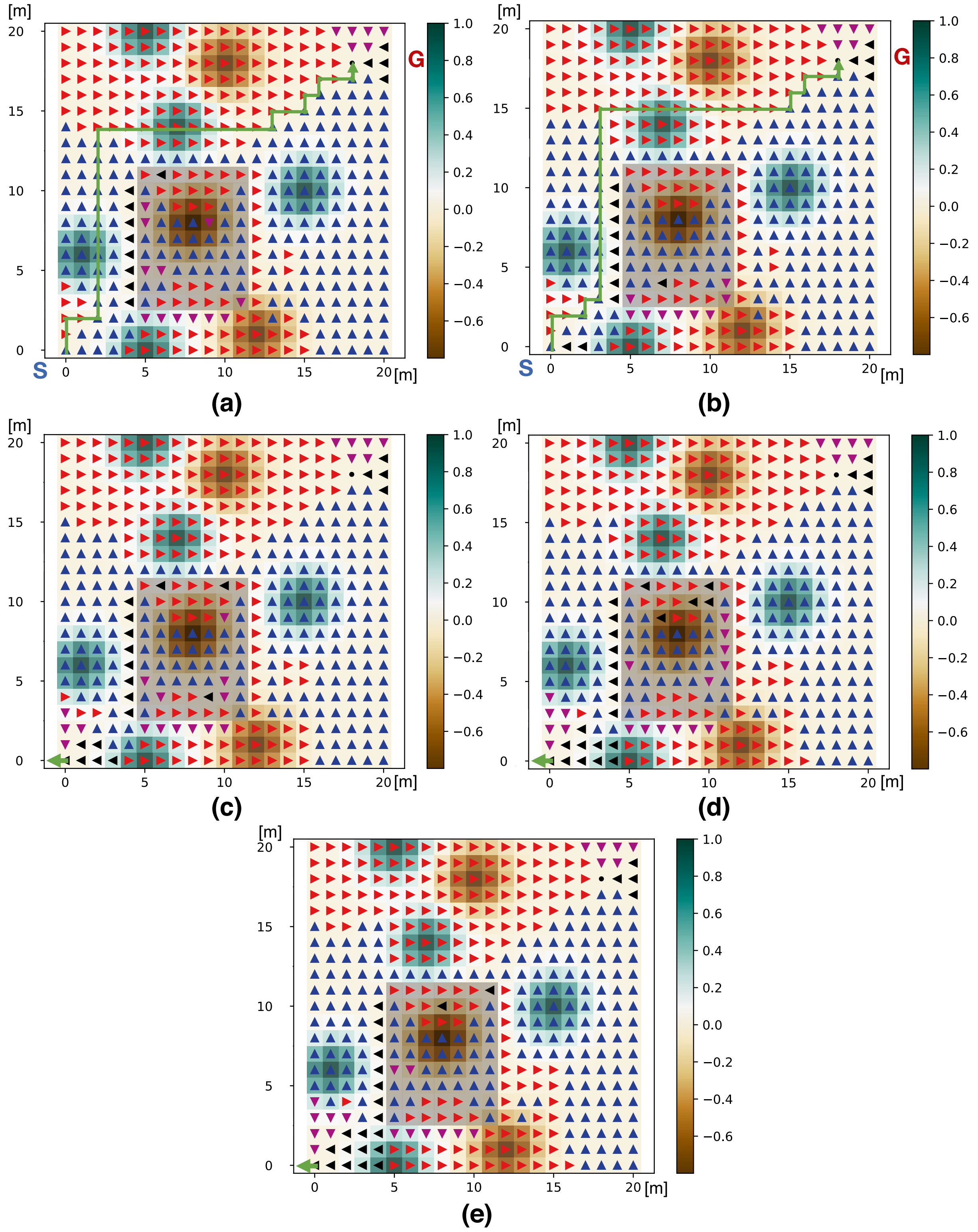


Fig. 7 The path planning results on heterogeneous terrain (2); **a**: $(w_1, w_2) = (0.6, 0.4)$, **b**: $(w_1, w_2) = (0.7, 0.3)$, **c**: $(w_1, w_2) = (0.8, 0.2)$, **d**: $(w_1, w_2) = (0.9, 0.1)$, **e**: $(w_1, w_2) = (1, 0)$

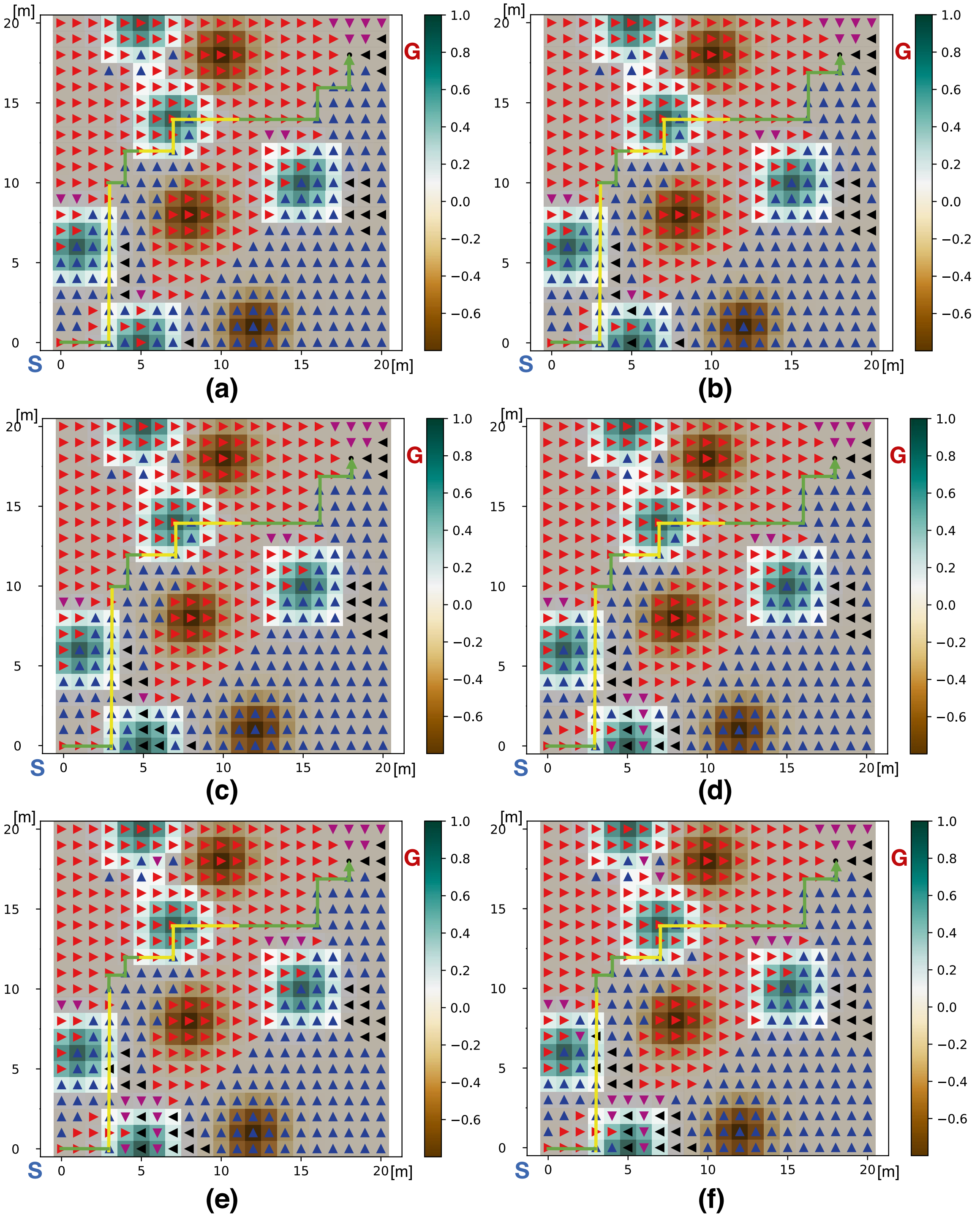


Fig. 8 The path planning results on sandy terrain (1); **a**: $(w_1, w_2) = (0, 1)$, **b**: $(w_1, w_2) = (0.1, 0.9)$, **c**: $(w_1, w_2) = (0.2, 0.8)$, **d**: $(w_1, w_2) = (0.3, 0.7)$, **e**: $(w_1, w_2) = (0.4, 0.6)$, **f**: $(w_1, w_2) = (0.5, 0.5)$

Simulations on heterogeneous terrain

This section shows the simulation on heterogenous terrain. A part of this terrain $(5 \leq x \leq 11, 3 \leq y \leq 11)$ is covered with sand. The hopping performance decreases on sandy terrain: the hopping distance $d' = 0.5$ [m] and height $h' = 0.125$ [m] at the hopping angle $\eta = 45$ [deg].

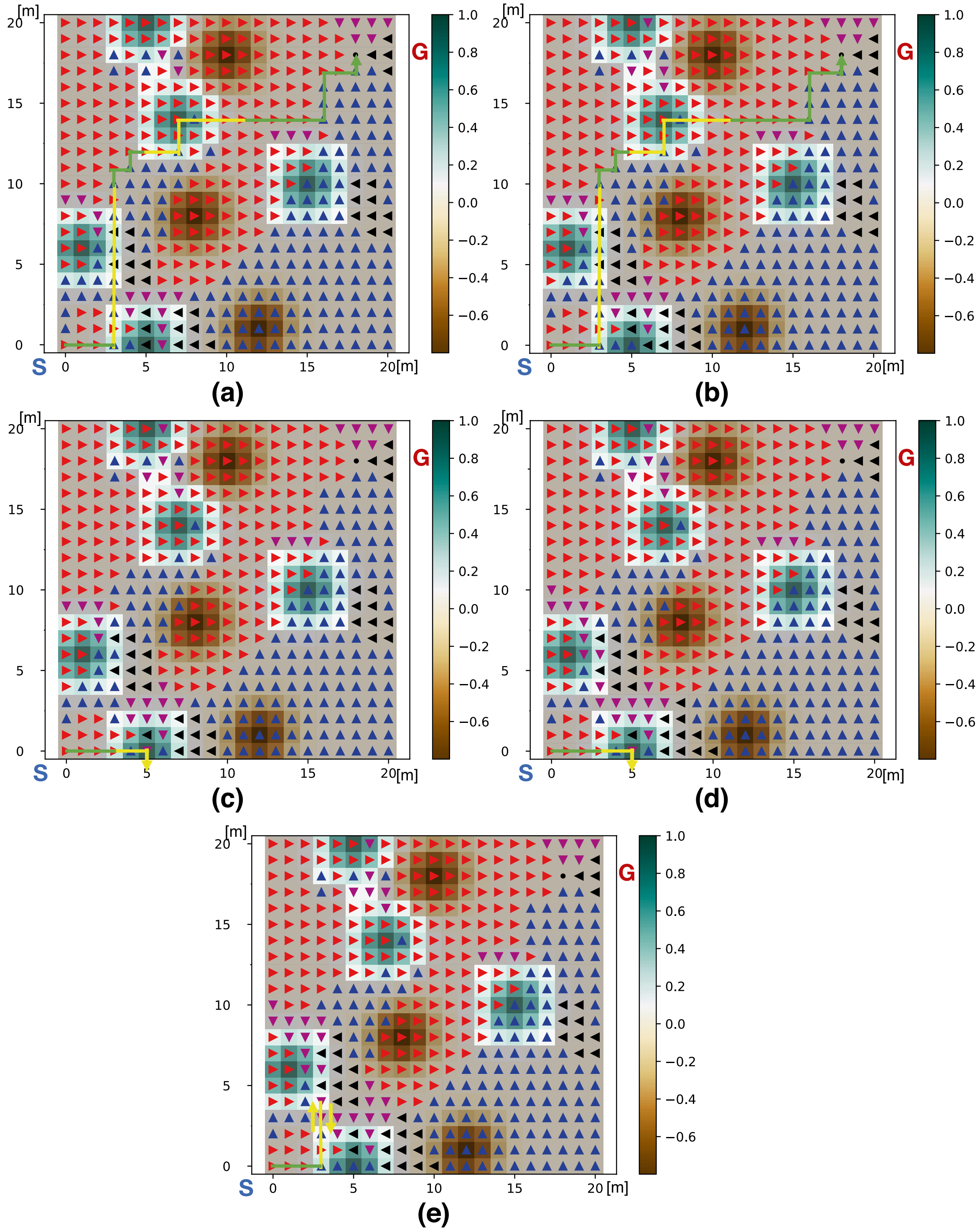


Fig. 9 The path planning results on sandy terrain (2); **a**: $(w_1, w_2) = (0.6, 0.4)$, **b**: $(w_1, w_2) = (0.7, 0.3)$, **c**: $(w_1, w_2) = (0.8, 0.2)$, **d**: $(w_1, w_2) = (0.9, 0.1)$, **e**: $(w_1, w_2) = (1, 0)$

In addition, the hopper might stuck in sand. This is why the penalty (negative value: -0.5) are added to the payoff function expressed as Eq. (3). Therefore, the uncertainty of locomotion on sand is re-defined as follows: the hopper

- can reach to the expected point with a $50\left(1-\frac{\theta_{\text{front}}}{(\pi/6)}\right)\%$ possibility
- may not change the state with a $50\frac{\theta_{\text{front}}}{(\pi/6)}$ possibility
- might get a stuck with a 20% possibility
- may be off to the left with a $15\left(1-\frac{\theta_{\text{right}}}{(\pi/6)}\right)\%$ possibility or to the right with a $15\left(1-\frac{\theta_{\text{left}}}{(\pi/6)}\right)\%$ possibility

This simulation employs the same values of the weight coefficients as the section .

The results are shown in from Figs. 6a to 7e. The part of sandy terrains are figured as translucent grey. The computing time is about 0.79 [s]. The paths are generated from the initial state, which avoid sandy terrains as shown in from Figs. 6(1) to 7b. The main difference between the generated paths is the actions on the elevated regions. If the hopper prioritizes to get information, the paths pass on the two tops of the hills in order to maximize the information gain. Figure 7a, b indicate the hopper selects a safer path than the others.

On the other hand, Fig. 7c–e show that there are no path from the initial state. These results indicate the hopper doesn't reach the goal from the start. In the case of prioritizing safety, the environment is too dangerous to go to the goal. In other words, the goal is not attractive for the hopper. There are the three choices in this case: (1) increasing the payoff of the goal, (2) decreasing the penalty of the sandy terrains, or (3) giving up to reach the goal in this environment. (1) Should be chosen if a robot want very much to arrive at the goal. Figure 10 shows the case that the payoff of the goal increases from 10 to 50. This case also chooses only the safety cost as the payoff functions. Although traversing on slopes, the hopper can arrive at the goal. The choice of (2) indicates that a robot might overestimate the risk of sandy terrain (e.g., the probability of stuck). The path may be generated from the start to the goal by decreasing the penalty. The hopper may move on sand in this case. If a robot have more maps, the robot should select the (3). Detours might be found. These results indicate the following observation:

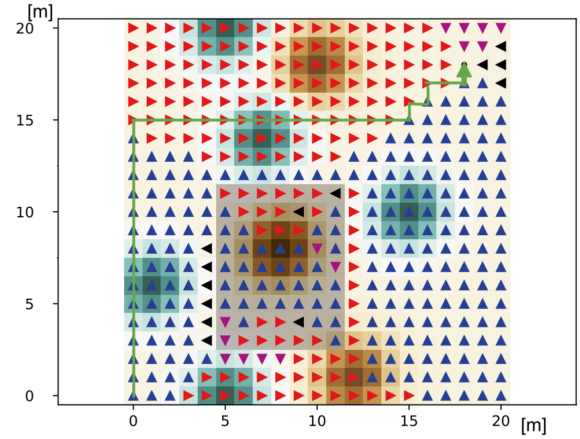


Fig. 10 The case of the more attractive goal $(w_1, w_2) = (1, 0)$. The hopper find the path from the start to the goal, which passes on the slopes

Observation 2: the proposed algorithm can generate the various path depending on the requirements on heterogeneous terrains. If the hopper doesn't get the path which reaches the goal from the start, examine the choices (1), (2), and (3) in order to find other routes.

Simulations on sandy terrain

This section shows the simulation on heterogenous terrain. Most of this terrain is covered with sand where the hopping performance decreases. This simulation uses the different performance: hopping distance $d' = 2.0$ [m] and height $h' = 0.5$ [m] on hard ground and $d' = 1.0$ [m] and height $h' = 0.25$ [m] on sandy terrains. The hopping angle is the same as the above sections $\eta = 45$ [deg]. The other assumptions are also same as the previous section.

The results are shown in from Fig. 8a–e. The part of sandy terrains are figured as translucent grey. The paths are illustrated as two colors: the green lines show the hopping routes on sandy terrain, and the yellow lines show the hopping on hard ground. The computing time is about 0.79 [s]. The paths are generated from the initial state as shown in from Figs. 8a to 9b. Even though the weight coefficients are different, these paths are almost same. The reason may be deduced that to reach the goal is most important with minimizing traversing on sandy terrain for hopper regardless of reward functions.

On the other hand, Figs. 9c, 8d, and 9e show that there are no path which starts from the initial state to the goal. In these cases, the hopper consideres that going to the goal is dangerous, i.e., the goal is not attractive for the hopper. The hopper might generate paths by selecting the option discussed in the previous section. Note that it is important to evaluate the risk of locomotion on sandy terrain, because the hopper might get stuck. These results indicate the following observation:

Observation 3: the proposed algorithm can generate a path on sandy terrains. If the hopper doesn't get the path which reaches the goal from the start, examine the choices in *Obserbation 2* in order to find other routes with taking a risk into account carefully.

Vertical hopping simulation

Finally, the vertical hopping actions are evaluated. The simulation uses the other environment. Figure 11 illustrates the new environment and the simulation result. The size of the environment is 14 [m] × 14 [m]. The goal is set on the $(x_p, y_p) = (14, 14)$. The environment is a flat plane, and includes the tall rock (the height is 1.0 [m]), and the low rock (the height is 0.25 [m]). The borders between the rocks and the flat terrain are not smooth. The gradients on the borders are very large (almost infinity). The other safety cost is defined for the vertical hopping as below:

$$S(\boldsymbol{x}_p, u_i) := -\Delta h, \tag{10}$$

where u_i and Δh denote the vertical hopping action and the difference $\left|h(x_{p+i}, y_{p+i}) - h(x_p, y_p)\right|$, respectively. The vertical hop is shown as a green arrow regardless of the hopping orientations. The payoff function of this section includes the information gain and the safety cost $((w_1, w_2) = (0.5, 0.5))$ in order to confirm the vertical hop action. The uncertainties of the vertical hopping action in this simulation are the following: the hopper

- can reach to the expected point with a $70\left(1 - \frac{\Delta h}{h_{\max}}\right)$% possibility
- may be off to the left with a $15\left(1 - \frac{\Delta h}{h_{\max}}\right)$% possibility or to the right with a $15\left(1 - \frac{\Delta h}{h_{\max}}\right)$% possibility
- fails to get on the rock with a $100\frac{\Delta h}{h_{\max}}$% possibility

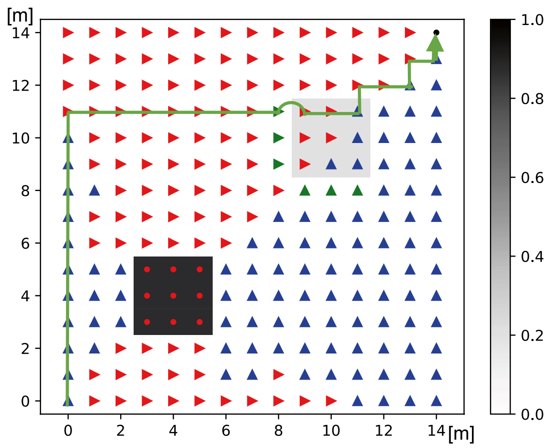


Fig. 11 The environment for the vertical hopping simulation. The tall rock is shown as dark grey, and the low rock is shown as light grey. The vertical hopping actions are plotted as green arrows. The path includes the vertical hopping shown as a green curve

where $h_{\max}$ is the hopping height at the hopping angle $\eta = 80$ [deg]. The uncertainties of the 45 [deg] hopping are the same as referred to the previous section. Here, $\theta_{\text{direction}} = 0$, because the terrain is flat. This simulation assumes that the probability of a failure is proportional to Δh.

The result of this simulation shows that the path can include the vertical hop at the $(x, y) = (9, 11)$. The vertical hopping actions are plotted as green arrows. The path includes the vertical hopping shown in a green curve. The computing time is about 0.55 [s]. It is also confirmed the other vertical hopping actions around the low rock. The hopper can get on the low rock to gain the information about the environment around the hopper. The actions around the tall rock avoid collisions by calculating the hopping trajectories. The result indicates the following observation.

Observation 4: the proposed algorithm can generate paths that pass on the non-smooth borders (e.g, rocks), and avoid the collision with too high rocks.

Conclusion

This paper presents the hopping path planning method for traversing on rough terrains, such as disaster areas, planetary environments, or both. One of the features of the proposed algorithm is calculating the hopping trajectory for the collision check. It enables generating paths on undulating terrain, such as slopes. This means the proposed algorithm is the path planning method in 2.5D. The advantage of this method is that the calculation cost is reduced, which is better than the fully 3D path planning methods. The contributions of this study is the followings: the proposed algorithm can generate the paths on rough terrain depending on the mission requirements by *Observation 1*, on heterogeneous terrain by *Observation 2*, on sandy terrain by *Observation 3*, and can traverse on non-smooth fields, such as rocks, or steps by *Observation 4*.

This summary remarks the important limitations of this work. (L1) this work does not consider the locomotion mechanisms of the hopper. The actions used in the proposed algorithm should be selected based on the hopping mechanisms. In particular, the way of recovering the attitude of the hopper after a landing is essential in order to continue the locomotion. (L2) The uncertainties of motions depend on the real environments. It is hardly that the expected uncertainties which described in section *Simulation Study* correspond with the actual uncertainties.

In order to tackle the limitations, future works are listed as below. (S1) developing hopping robots and correct hopping data. The works assist us to understand the restrictions of hopping actions, and improve the proposed algorithm. Moreover, we will include the energy constraints in the payoff function, which can reduce the number of turns. (S2) The reinforcement learning can renew uncertainties of motion, or a hopping model, or the both. The MDPs are the basic theory of the reinforcement learning. Therefore, the proposed algorithm can use the reinforcement learning method relatively easily. The transfer learning is also used to modify a hopping model and uncertainties, which the hopper learned in advance.

Acknowledgements
Not applicable.

Authors' contributions
KS and TK conceived the idea of the study; KS designed and performed research, and wrote the paper. All authors discussed the results and revised the manuscript. Both authors read and approved the final manuscript.

Funding
This work was supported by JSPS KAKENHI Grant JP19J10278.

Availability of data and materials
Data sharing is not applicable to this article as no datasets were generated or analysed during the current study.

Declaration

Ethics approval and consent to participate
The authors agree this section.

Consent for publication
The authors agree the publication.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Department of Electrical Electronic and Communication Engineering, Faculty of Science and Engineering, Chuo University, Bunkyo-ku, 112-8551 Tokyo, Japan. [2]Institute of Space and Aeronautical Science, Japan Aerospace Exploration Agency, Sagamihara, 252-5210 Kanagawa, Japan.

Received: 14 April 2021 Accepted: 31 January 2022
Published online: 14 February 2022

References

1. Tsukagoshi H, Sasaki M, Kitagawa A, Tanaka T (2005) Design of a higher jumping rescue robot with the optimized pneumatic drive. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 1276–1283. IEEE
2. Reid RG, Roveda L, Nesnas IA, Pavone M (2014) Contact dynamics of internally-actuated platforms for the exploration of small solar system bodies. In: International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), Saint-Hubert, Canada, p. 9
3. Mège D, Gurgurewicz J, Grygorczuk J, Wiśniewski Ł, Thornell G (2016) The highland terrain hopper (hopter): concept and use cases of a new locomotion system for the exploration of low gravity solar system bodies. Acta Astronautica 121:200–220
4. Montminy S, Dupuis E, Champliaud H (2008) Mechanical design of a hopper robot for planetary exploration using sma as a unique source of power. Acta Astronautica 62(6–7):438–452
5. Yoshimitsu T (2004) Development of autonomous rover for asteroid surface exploration. In: IEEE International Conference on Robotics and Automation (ICRA), vol. 3, pp. 2529–2534
6. JAXA: MINERVA-II1: Images from the surface of Ryugu. http://www.hayabusa2.jaxa.jp/en/topics/20180927e_MNRV/
7. Levy J (2012) Hydrological characteristics of recurrent slope lineae on mars: evidence for liquid flow through regolith and comparisons with antarctic terrestrial analogs. Icarus 219(1):1–4
8. Kavraki LE, Svestka P, Latombe J, Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Robot Automat 12(4):566–580
9. LaValle SM, Kuffner JJ Jr (2001) Randomized kinodynamic planning. Int J Robot Res 20(5):378–400
10. Kavraki L, Latombe JC (1994) Randomized preprocessing of configuration for fast path planning. In: IEEE International Conference on Robotics and Automation, pp. 2138–2145. IEEE.
11. Karaman S, Frazzoli E (2010) Optimal kinodynamic motion planning using incremental sampling-based methods. In: IEEE Conference on Decision and Control (CDC), pp. 7681–7687. IEEE.
12. Choudhury S, Scherer S, Singh S (2013) Rrt*-ar: Sampling-based alternate routes planning with applications to autonomous emergency landing of a helicopter. In: IEEE International Conference on Robotics and Automation, pp. 3947–3952. IEEE.
13. Reeds J, Shepp L (1990) Optimal paths for a car that goes both forwards and backwards. Pacific J Mathemat 145(2):367–393
14. Kim D-S, Cho Y, Kim D (2005) Euclidean voronoi diagram of 3d balls and its computation via tracing edges. Comput Aided Design 37(13):1412–1424
15. Verscheure L, Peyrodie L, Makni N, Betrouni N, Maouche S, Vermandel M (2010) Dijkstra's algorithm applied to 3d skeletonization of the brain vascular tree: evaluation and application to symbolic. In: IEEE International Engineering in Medicine and Biology Conference, pp. 3081–3084. IEEE.
16. De Filippis L, Guglieri G, Quagliotti F (2012) Path planning strategies for uavs in 3d environments. J Intell Robot Syst 65(1–4):247–264
17. Koenig S, Likhachev M (2005) Fast replanning for navigation in unknown terrain. IEEE Trans Robot 21(3):354–363
18. Howard TM, Kelly A (2007) Optimal rough terrain trajectory generation for wheeled mobile robots. Int J Robot Res 26(2):141–166
19. Ishigami G, Kewlani G, Iagnemma K (2010) Statistical mobility prediction for planetary surface exploration rovers in uncertain terrain. In: 2010 IEEE International Conference on Robotics and Automation, pp. 588–593. IEEE.
20. Cunningham C, Ono M, Nesnas I, Yen J, Whittaker WL (2017) Locally-adaptive slip prediction for planetary rovers using gaussian processes. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 5487–5494. IEEE.
21. Luders BD, Karaman S, How JP (2013) Robust sampling-based motion planning with asymptotic optimality guarantees. In: AIAA Guidance, Navigation, and Control (GNC) Conference, p. 5097.
22. Melchior NA, Simmons R (2007) Particle rrt for path planning with uncertainty. In: Proceedings 2007 IEEE International Conference on Robotics and Automation, pp. 1617–1624. IEEE.
23. Peynot T, Lui S-T, McAllister R, Fitch R, Sukkarieh S (2014) Learned stochastic mobility prediction for planning with control uncertainty on unstructured terrain. J Field Robot 31(6):969–995
24. McAllister R, Peynot T, Fitch R, Sukkarieh S (2012) Motion planning and stochastic control with experimental validation on a planetary rover. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4716–4723. IEEE.
25. Hockman B, Pavone M (2020) Stochastic motion planning for hopping rovers on small solar system bodies. Robotics research. Springer, Berlin, pp 877–893
26. Sakamoto K, Otsuki M, Maeda T, Yoshikawa K, Kubota T (2019) Evaluation of hopping robot performance with novel foot pad design on natural terrain for hopper development. IEEE Robot Automat Lett 4(4):3294–3301