IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 16 June 2023, accepted 28 June 2023, date of publication 3 July 2023, date of current version 7 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3291401

RESEARCH ARTICLE

A MDPs-Based Dynamic Path Planning in Unknown Environments for Hopping Locomotion

KOSUKE SAKAMOTO, (Member, IEEE), AND YASUHARU KUNII, (Member, IEEE)
Department of Electrical, Electronic, and Communication Engineering, Chuo University, Bunkyo, Tokyo 112-8551, Japan

Corresponding author: Kosuke Sakamoto (ksakamoto605@g.chuo-u.ac.jp)

This work was supported in part by the Chuo University Joint Research Grant.

ABSTRACT Hopping robots, or ''hoppers'', are promising explorers capable of navigating rough terrain such as disaster areas and celestial environments. For example, rovers exploring planetary surfaces need to minimise the risk of failure and maximise the acquisition of information about their environment. Hopping locomotion in such environments is inherently uncertain because the details of the environment are largely unknown. As a result, path planning algorithms must account for these uncertainties in order to effectively traverse these environments. This study presents a novel hopping path planning algorithm for uncertain environments. The proposed algorithm uses Markov Decision Processes (MDPs) to compute motion uncertainties and subsequently generates optimal actions for all states according to the terrain conditions and mission requirements. In addition, the proposed algorithm incorporates a perception method using hopping locomotion features, which enables dynamic path generation. The proposed algorithm is evaluated through simulations in three different environments, which demonstrate that the hopper can achieve its goals with a 98% success rate on hard ground and heterogeneous terrain, and over 80% success rate on sandy terrain, using the proposed algorithm. Furthermore, the robustness of the proposed algorithm is validated through comparison with a greedy algorithm using the same payoff function, which shows that the proposed algorithm achieves 20 times higher success rate and 38.7% lower average number of steps than the greedy algorithm in the best case scenario.

INDEX TERMS Motion and path planning, optimization and optimal control, space robotics.

I. INTRODUCTION

A variety of locomotion mechanisms for field robots have been developed for activities in various challenging terrains. In particular, hopping locomotion is one of the most promising mechanisms. In fact, numerous hopping robots, called hoppers, have been designed for playing an active role in disaster areas [1], celestial bodies [2], [3], [4], [5], and so on. For example, the MINERVA-II robot, which was developed by JAXA/ISAS, is one of the successful hoppers. The hopper could traverse and take the surface of the asteroid ''Ryugu'' in September 2019 (shown in Fig.1) [6]. JAXA is also planning the SLIM mission, a lunar exploration program that will deploy a hopper rover [7]. Furthermore, the Moonshot project is conducting research on lunar exploration using hoppers and the development of a lunar base [8]. However, there are many challenges to the success of in a planetary surface exploration mission by a hopper/hoppers.

One of the challenges is to cope with control uncertainty of the path or motion planning problems. The details of the planetary environments are unknown until the robot arrives and explores on site. In addition, the Moon or Mars are covered with dry sand, called regolith. Sand decreases hopping performance due to slippage and may cause the hopper to get stuck [10]. In other words, control uncertainty makes it difficult for robots to follow a path accurately on unstructured terrain. Therefore, motion planning must take such uncertainty into account for safe and efficient navigation. In addition, a spatial motion of hopping must be considered in a 3D environment, which indicates that the hopping motion is more complex than a motion in a 2D environment. As a result, control uncertainty may

The associate editor coordinating the review of this manuscript and approving it for publication was Hongli Dong.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

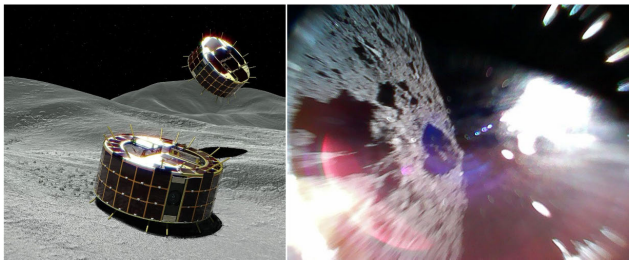


FIGURE 1. MINERVA II robot and taken the photo. Left: the image of MINERVA II on asteroids. Right: the photo of "Ryugu" [9].

be larger than that of wheeled vehicles. Furthermore, the computational cost is higher in a 3D environment than that in a 2D environment. In unknown environments, there is no guarantee that the robot will reach the goal, because the robot cannot generate paths to the goal. What's more, we don't know whether the direction in which the robot is moving is safe or traversable. In the worst case, the robot may get stuck or deadlocked. Therefore, the robot needs to generate actions or paths that are traversable, reduce uncertainty, have sufficient safety, and avoid deadlocks. Conventional path planning algorithms have been improved to generate paths in unknown environments or uncertainty for wheeled robots. However, these algorithms combine two or three methods, operate in partially known environments, or assume that the wheels are well controlled. These assumptions make it difficult to apply the conventional methods to hopping robots for locomotion in completely unknown environments. Therefore, a path planning method is required that enables a hopping robot to account for the uncertainty of its motion in completely unknown environments.

To address the above challenges, this paper proposes and evaluates a dynamic hopping path planning algorithm for traversing unknown environments. The contributions of this paper are listed below:

- We propose an MDP-based hopping path planning algorithm that maximises the payoff and minimises the control uncertainty to reach the goal.
- We design the payoff function, which can change depending on the mission requirements or environmental conditions.

Our proposed method is based on Markov Decision Processes (MDPs), which generate the best actions for all observable states. The motion uncertainty is formulated as a probability and the payoffs at each state are defined in the MDPs, then the MDPs use the probabilities and the payoffs to compute the expected values. The generated best actions maximise the expectation values through MDPs. In this paper, we achieve dynamic path planning by modifying the actions generated by MDPs by expanding the observation area while moving in unknown environments. We design a novel payoff function for a hopping robot operating in unknown environments. The proposed payoff function can quantitatively define uncertainty, thus enabling adaptation to different uncertain environments regardless of the terrain.

TABLE 1. The symbols and notation used in this paper.

Symbol	Description
x	The state of the robot
u	An action
$p(x'\|x,u)$	Probability of transition
$r(x,y)$	Payoff function
$V_T(x)$	Value function at step T
$\pi(x)$	The control policy at the state x
w_1, w_2, w_3	Weight coefficients
$S(x,u)$	Safety cost
$I(x,u)$	Information gain
$D(x,u)$	Penalty term

The proposed algorithm is tested in three environments and its performance is evaluated. This paper extends our previous work, which only considered known environments and restricted the payoff function [11]. This work showed that the proposed algorithm is able to generate paths on unstructured terrain. On the other hand, some deadlocks were observed on sandy surfaces. Accordingly, in this paper, the payoff function is modified to avoid deadlocks and reach the goal on sandy terrain. In addition, to evaluate the performance of the proposed algorithm in unknown environments, we simulate whether the robot can reach the goal or not by alternating between observation and locomotion. Furthermore, we validate the robustness of the proposed algorithm by comparing it with the greedy algorithm using the same payoff function. The performance of these algorithms is evaluated in terms of the goal-reaching rate. This paper also presents a ''treasure hunting'' on hard ground, which evaluates how many interesting objects (called ''treasures'') the robot collects while reaching the goal.

The contents of this paper are described as follows: Section II reviews related works on the topic. Section III introduces the proposed algorithm for hopping path planning on rough terrain, and the methods used are shown. Section IV presents the simulation results to evaluate the proposed algorithm. Finally, conclusions of this paper and future works are shown in Sec.V. Table 1 denotes the symbols and notation used in this paper.

II. RELATED WORKS

Path planning, or motion planning in unknown environments, has been an important technology in robotics. The ''unknown'' or ''uncertain'' environment typically means an environment that is not observed or has dynamic obstacles. One of the most widely used algorithms for partially known or dynamic environments is D [12], which is a heuristic graph search method based on A replanner. D* has been widely applied and extended to robot motion planning, such as D* Lite [13] or Field D [14]. These methods re-plan an initial path based on dynamic environmental information. For example, Mars exploration rovers used Field D to act in natural environments [15]. Jihee Han extended A* to generate a path in unknown environments [16]. The proposed algorithm sets grid points around obstacles in only

the observed area. Although the computational cost of the proposed method is lower than A*, the quality of the paths is almost the same.

Sampling-based methods have also been extended to path planning in unknown environments. Chance-constraint RRT [17] combines the RRT algorithm with the uncertainty of collision as a constraint. The objective function expresses the risk of collision as a probability and the cost of the path length, which allows a trade-off between the shortest path and the least risky path. Lukas et al. applied RRT* to online informative path planning in unknown environments [18]. The proposed method expands a single tree continuously to maximize an objective function, which avoids local minima. Cai et al. proposed a sampling-based method that generates a collision-free path in a dynamic environment considering localization uncertainty [19]. The method biases the sample towards the information-rich region to reduce the uncertainty and collision risk. GMR-RRT* applied human navigation behaviour through Gaussian Mixture Regression (GMR) to the sampling process of RRT [20].

Another mainstream trend is the combination of global and local planning methods. A global path planning algorithm generates an optimal path (e.g., the shortest path) by minimizing an objective function. A local path planning algorithm calculates a control input to avoid dynamic obstacles while following a global path. The algorithms mentioned above, such as A*, D*, or RRT*, are global path planners. DWA is one of the most famous and widely used local planners [21]. The combination of a global path planner and DWA is a practical solution. On the other hand, the performance of DWA depends on the parameter setting of its cost function. Therefore, recent studies have proposed optimization method-based local planners instead of DWA: Reference [22] uses the Voronoi planner to find a collision-free path and a potential field to calculate the control input. Reference [23] generates a global path using PSO and a local path using A*. Reference [24] applies PSO to re-calculate a local path generated by D*. Reference [25] creates a map by random walk at first, then generates optimized waypoints from the start to the goal using PSO. Reference [26] employs the interval type-2 fuzzy logic system (IT2FLS) as a local planner, with parameters optimized by an artificial bee colony (ABC) algorithm.

Recently, reinforcement learning-based algorithms have been widely proposed to improve planning algorithms in unknown environments. Using reinforcement learning (RL), an agent (a robot) learns the best control policy, that maximizes a value function depending on the environment. Hockman and Pavone propose a hopping trajectory planning method on asteroids [27]. They optimized a single hopping trajectory in irregular gravity fields and derived optimal control policies by RL. Reference [28] optimized the parameters of DWA using Q-learning to improve navigation in unknown environments. They modified the evaluation function of DWA and decided the weight coefficients of the evaluation function by Q-learning. The parameter sets performed better than the empirical parameter set. Sombolestan et al. generate the control policy using RL in an unknown environment [29] to find the shortest path without a map in an unknown environment, such as a disaster area. Some studies apply deep reinforcement learning using images from a robot [30], [31], [32]. Reference [33] applied DRL to collision avoidance in unknown environments for multi-robot exploration. They used the 24 laser rangefinder states, two previous velocity states, and two robot position states as inputs instead of images. A drawback of RL-based methods is that the performance of the algorithms depends on the learning environment. In other words, there is no guarantee that the algorithm performs well in environments different from the learning environment.

The Markov Decision Processes (MDPs) based algorithms calculate the best set of actions, called the control policy $\pi(x)$, under motion uncertainty. The uncertainty is expressed as a stochastic transition process. The control policy generated by MDPs maximizes an expected value $V = \Sigma_x \pi(x) \times r$, and hence it is important to design the payoff function r. Nardi et al. [34] formulated the navigation problem in an urban environment as Augmented MDPs. In addition to the motion uncertainty, they incorporated the robot's position uncertainty into the problem. Therefore, their approach generates paths that can trade off between travel distance and safety. Peynot et al. used a Gaussian process regression model to learn the statistical transitions of actions from experience [35], [36]. They then applied MDPs to the outcome and demonstrated safe path planning under control uncertainty. Feyzabadi and Carpin proposed a risk-aware planning method using constrained Markov decision processes (CMDPs) [37]. The method treats multiple objectives as constraints to generate the path through CMDPs. To cope with the complexity of CMDPs, they proposed a hierarchical method that produces suboptimal paths but reduces calculation costs.

The artificial potential field (APF) was applied to path planning by [38]. The method produces a low potential field around the goal or targets, which generates an attractive force, as well as a high potential field around obstacles, which establishes a repulsive force. The method has been widely used due to its robustness and simple mathematical model that is easy to implement. Recent studies have used APF-based models, such as a combination of membrane computing and APF [39], [40], or Q-learning and APF [41]. These studies produced the best parameters of APF using optimization methods to generate a safe and feasible path in obstacle-rich environments. Reference [42] proposed a new attraction field and repulsive field function to avoid local minima.

Several path planning methods use terrain information to improve driving performance on rough terrain. For instance, [43] and [44] employ slope information to predict slip on soft ground and incorporate it into the cost function of path planning; [45] proposes a path planning method that uses

terrain slope to account for kino-dynamic constraints; [46] replicates the terrain on which the robot operates in a physics engine and generates a path. The robot follows the path generated by D*Lite in the simulation. The path is re-generated by setting the terrain that the robot could not traverse as an impassable point, and the process is repeated until the robot reaches the goal.

This paper uses MDPs to solve the hopping path generation problem in unknown environments. The reasons are as follows:

- MDPs can generate robust paths against motion uncertainty.
- MDPs work well despite differences between environments.
- The payoff function of MDPs is easy to extend.

MDPs calculate the best actions in all observed states, so that a robot does not need to re-generate a path if the robot deviates from the estimated landing point. The maximum value is expressed as the Bellman equation, which ensures the optimality of the policy. Therefore, MDPs can generate the best actions in various environments. One of the crucial aspects of the proposed algorithm is the parallel exploration towards a goal in an unknown environment, which allows reaching the goal while updating the map. Since we define the unknown environment as the unobserved area, our method allows generating the best action in the observed area and reaching the goal while reducing the unobserved area, even in environments where conventional methods, such as the D* algorithm, cannot generate a route to the goal using a high-level planner. In addition, the payoff function enables the inclusion of some constraints based on the mobility platform as penalty terms. These features are the unique advantages of MDPs that the sampling-based method, APF, RL method, or any optimization methods don't have.

III. PROBLEM DESCRIPTION

This section explains the details of the proposed algorithm, the assumptions, and the conditions of the simulation. As described above, this study employs MDPs to calculate uncertainties and generate paths. The path is generated by connecting the actions. MDPs produce the best action in all observed states, which allows a hopper to continue locomotion without replanning new actions when it fails to follow the action. MDPs require that motion uncertainty must be expressed as a stochastic transition. Although motion uncertainty is caused by several factors, such as hardware error, software error, or terrain interaction, this paper doesn't specify the causes and the details of the uncertainty. Since this paper focuses on the performance of the proposed algorithm, the probability of transition is given as a constant value. MDPs pre-calculate the best action in all possible states; however, this study limits the number of possible states due to calculation costs. This study does not focus on attitude or trajectory control while hopping.

In addition, this paper evaluates the generated paths and the success rate of reaching the goal, and doesn't focus on a single hopping trajectory. Regarding the optimization of a single trajectory, [47] proposed an optimization method considering contact uncertainty due to the terrain.

In this paper, we design the payoff function to generate the policy by MDPs. The proposed payoff function consists of safety, information gain, and the direction of the goal. We can design the payoff function by changing the priorities of these terms depending on the mission, the condition of the environment, and so on. The proposed algorithm is tested in simulations. The virtual environment has three cases: hard ground, partially sandy environment, and sandy terrain. The differences in the generated paths are compared between these environments or the priority of the payoff function, and the proposed algorithm is evaluated by the reaching rate to the goal.

A. ASSUMPTIONS

Although this method performs path planning on a 2D grid map, it is characterized as 2.5D path planning because it uses DEM (Digital Elevation Map) elevation data to accommodate for the collision check of hopping trajectories. The hopping trajectory is assumed to be a parabolic motion, which eliminates the need for full 3D calculations using the elevation data. In other words, the elevation $h(x, y)$ and the parabolic trajectory $z = -C_1(\boldsymbol{x} - C_2) + C_3$ at the point $\boldsymbol{x} = (x, y)$ are used for collision check, where $C_1 = \left(\frac{(\boldsymbol{x}_\mathrm{L} - \boldsymbol{x}_\mathrm{H})v_z}{2g}\right)$, $C_2 = \frac{\boldsymbol{x}_\mathrm{L} - \boldsymbol{x}_\mathrm{H}}{2}$, and $C_3 = \frac{v_z}{2g}$. Here, we define the hopping point $\boldsymbol{x}_\mathrm{H}$, the landing point $\boldsymbol{x}_\mathrm{L}$, the initial hopping velocity v_z, and a gravitational acceleration g. Based on this assumption, this study defines the proposed method as a 2.5D path planning algorithm. The robot only knows the direction of the goal in advance, and the details of the environment are unknown. Paths are generated only in known areas, which are expanded step by step by the robot's locomotion. In this paper, "information gain" is defined as the details of the environment acquired through observation. Therefore, the purpose of this paper is for the robot to move towards the goal by repeatedly observing and moving. The motion uncertainties are caused by the roughness of the terrain, the material of the surface, and so on. These uncertainties generally decrease the performance of the hopper. This study is based on the following assumptions:

- Do not specify what causes the uncertainty P and the degree of the uncertainty quantitatively.
- The uncertainty P is expressed as the dispersion of the landing point (x, y) probabilistically, i.e., $0 \leq P(x, y) \leq 1 \wedge \sum_x \sum_y P(x, y) = 1$.

The details of the probabilistic uncertainty are described in Section IV-B. Figure 2 shows the schematic image of hopping with uncertainty. The hopper attempts to land in front of the obstacle, and the estimated trajectory is expressed as a red dashed line. The uncertainty area indicates the dispersion of the landing point. This study doesn't consider the uncertainty from perceptions. Therefore, the position and attitude of the hopper are already known in all possible states.

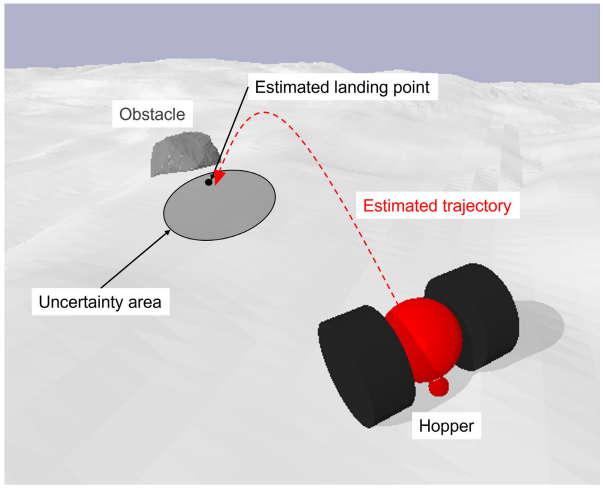


FIGURE 2. The image of the hopping with uncertainty.

B. CONDITIONS OF THE ROBOT

This paper simplified hopper conditions in order to evaluate the proposed algorithm in the simulations as a first step. The hopper is treated as a mass point. The hopper can hop in four horizontal directions (north, south, east, and west) and the hopping trajectory is calculated as a parabolic motion. The trajectory is calculated from the initial velocity v_0. The mechanical loss is zero, and hence the initial kinetic energy $E_k = 1/2mv_0^2$ is equal to the potential energy $E_p = 1/2kx_0^2$, i. e., $E_k = E_p$. The hopper is equipped with power sources that generate the hopping force, and appropriate sensors to perceive the environments, such as stereo cameras. The hopping force is a conservative force, such as a spring.

C. MODELING

Markov decision processes (MDPs) use the probability of the state x' from the selected action u under the state x. The action model $p(x'|x, u)$ is determined previously. Using this model, MDPs determine the control policy $\pi(x)$ that maximizese the expected payoff $r(x, u)$. At step T, the policy $\pi_T(x)$ is given by:

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

where $V_T(x)$ denotes the value function which is expressed as follows:

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

where γ denotes the discount factor. Taking the limit $T \to \infty$, Eq. (2) is called as Bellman equation. Bellman equation means that all value function converge to the maximum value $V(x)$. Therefore, the optimal policy $\pi(x)$ can be obtained by Eq. (1). This paper uses $\gamma = 0.99$.

In order to calculate the optimal value function numerically, this study employs Value iteration method. Algorithm 1 shows the procedure of the value iteration.

Where N denotes the number of the states.

Algorithm 1 Value Iteration

```
for i = 1 → N do
    V(x_i) ← r_min
end for
while until V(x_i) converge do
    for i = 1 → N do
        V(x_i) ← γ max_u [ r(x_i, u) + Σ_{j=1}^{N} V(x_j) p(x_j|u, x_i) ]
    end for
end while
return V
```

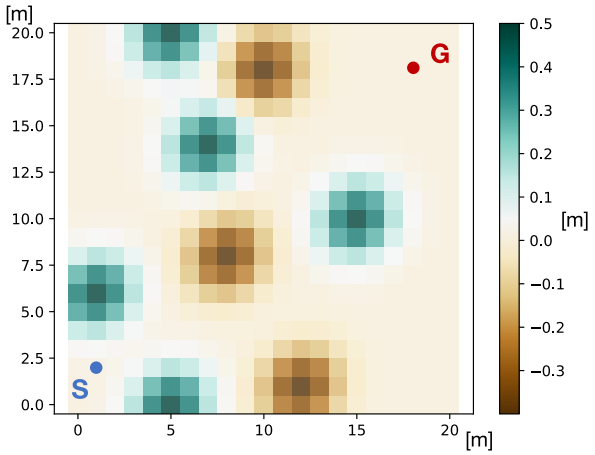


FIGURE 3. The artificial terrain. The higher places are colored green, and the lower places are colored brown. The size of each cell is 1 m × 1 m. The start (blue point) is (1, 2) and the goal (red point) is (18, 18). The environment is same as our previous work [11].

D. EVALUATION FUNCTION

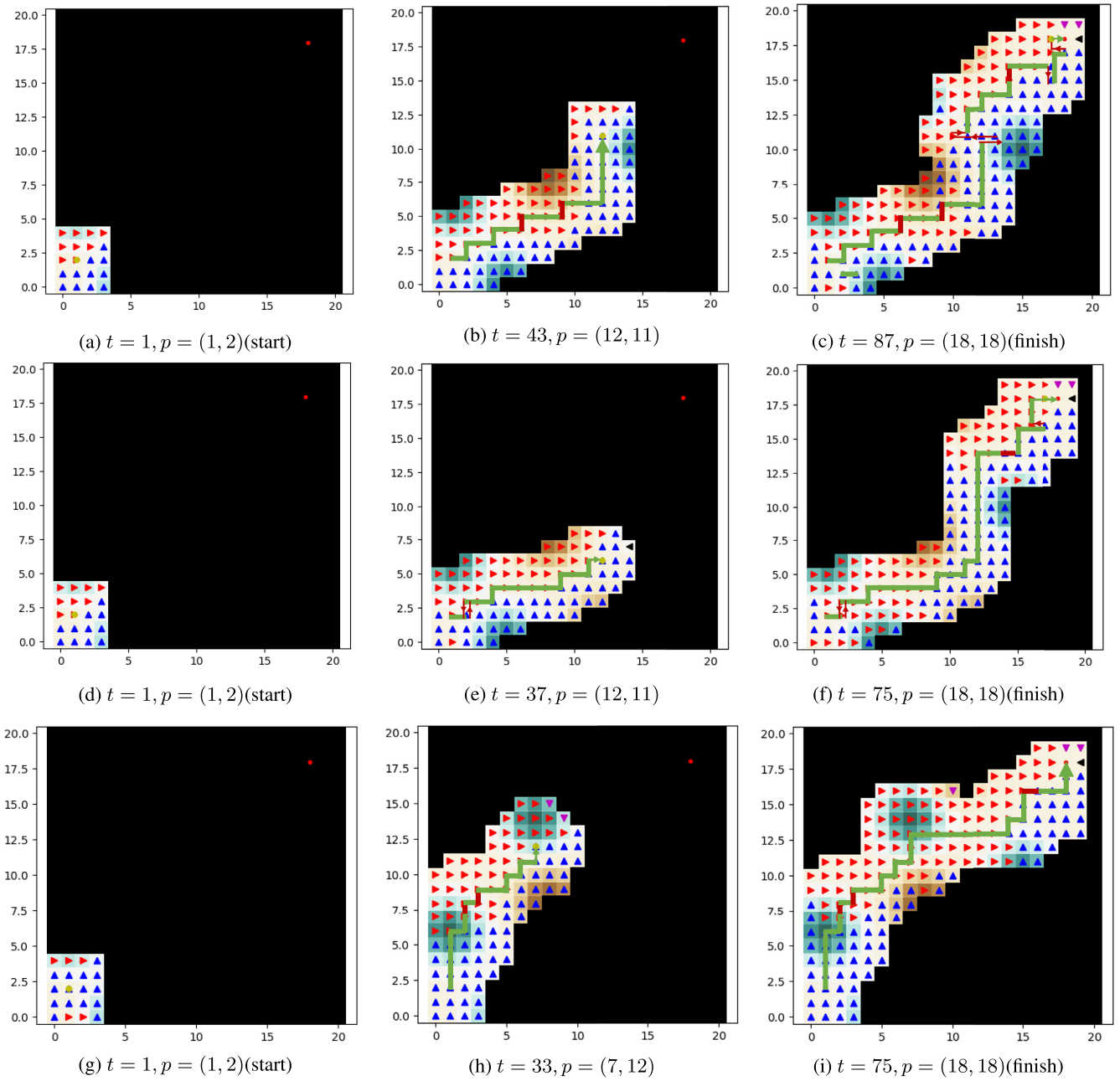
The way of designing the payoff function $r(x, u)$ is one of the important technique of MDPs. The payoff functions are designed by considering the above assumptions and motion features. The design concept of the payoff function of this study is to maximaze the infomation gain and safety, and minimize the length of the path. The proposed payoff function is expressed as:

$$r(x,u) = w_1 S(x,u) + w_2 I(x,u) + w_3 D(x,u) \quad (3)$$

where $S(x, u)$, $I(x, u)$ and $D(x, u)$ denote the safety cost, the information gain, and the penalty term in regard to action, respectively. The w_1, w_2, and w_3 are the weight coefficients.

The safety of the locomotion depends on the interaction between the robot and the environments, i. e., the safety cost is proportional to the roughness of the surface. On the other hand, hopping robots can ride on relatively flat rocks, or get over steps. These features indicate that hoppers can traverse more challenging regions than what wheeled rovers traverse. Therefore, the function of the safety cost is defined as follows:

$$S(\mathbf{x}_p) = -\left|\nabla h(x_p, y_p)\right| - T(\mathbf{x}_p) \quad (4)$$



(a) $t = 1, p = (1, 2)$(start) (b) $t = 43, p = (12, 11)$ (c) $t = 87, p = (18, 18)$(finish)

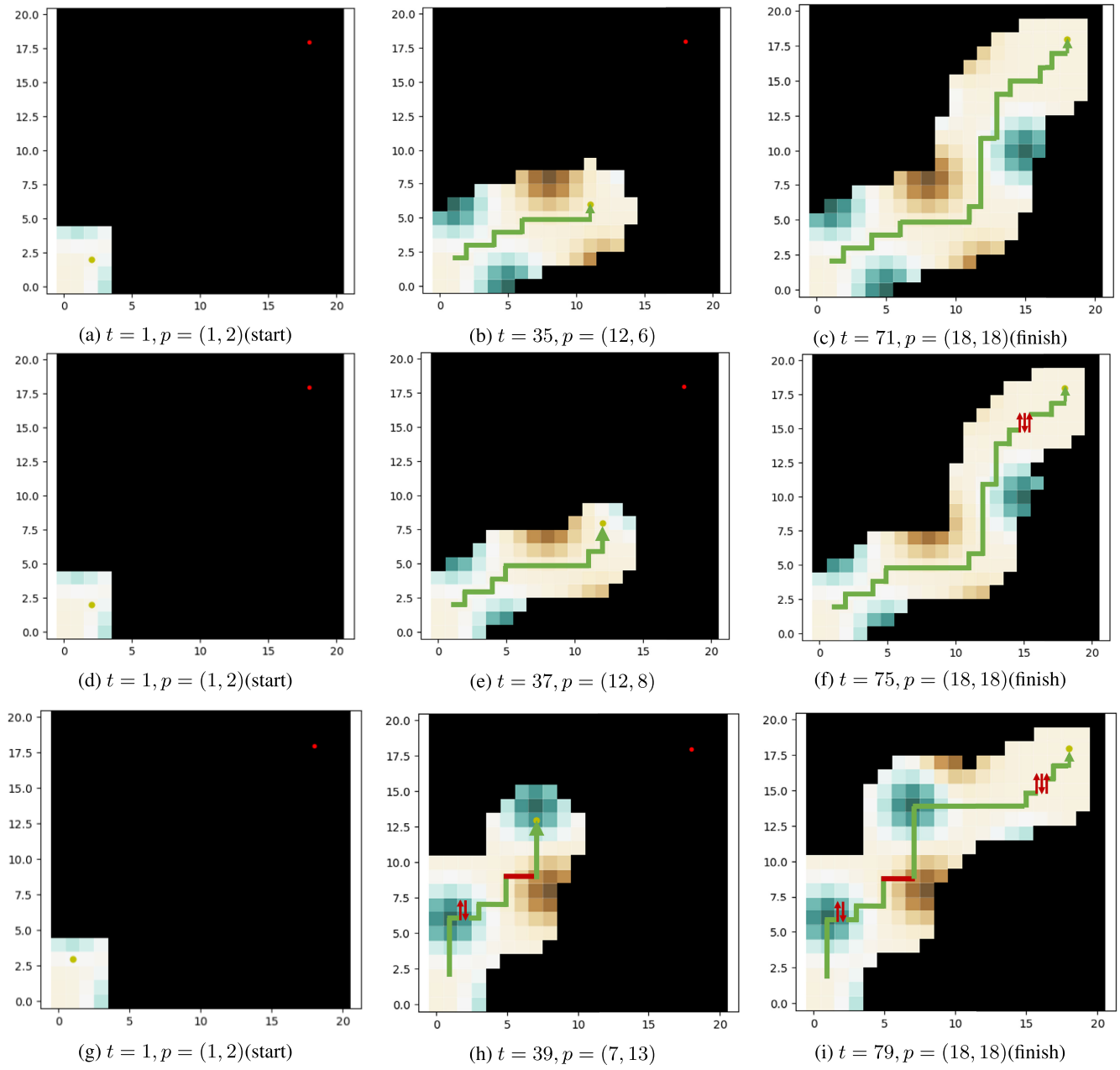
(d) $t = 1, p = (1, 2)$(start) (e) $t = 37, p = (12, 11)$ (f) $t = 75, p = (18, 18)$(finish)

(g) $t = 1, p = (1, 2)$(start) (h) $t = 33, p = (7, 12)$ (i) $t = 75, p = (18, 18)$(finish)

FIGURE 4. The simulation results of the proposed algorithm: (a) - (c) the safety-critical case ($(w_1, w_2) = (1, 0)$), (d) - (f) $S(x, u)$ and $I(x, u)$ have the same level of effect ($(w_1, w_2) = (0.5, 0.5)$), and (g) - (h) the information gain-critical case ($(w_1, w_2) = (0, 1)$) on hard ground. The blue, purple, red, and black arrows denote the moving direction to north, south, east, and west, respectively. The green lines indicate the correct paths on which the hopper moved, and red lines indicate the incorrect paths. t and p denote the number of steps and the hopper position, respectively; Left figures: the initial state; Middle figures: the halfway state; Right figures: the finish.

where $h(x_p, y_p)$ and $T(\mathbf{x}_p)$ denote the height of the terrain at a point (x_p, y_p), and the risk of motion based on the soil condition, respectively. The roughness of terrain on a point $\mathbf{x}_p = (x_p, y_p)^{\mathrm{T}}$ is formulated as the tilt at that point. Given the findings from our prior hopping experiments [10], the hopping performance on sandy soil was found to be inferior to that on hard ground. Furthermore, a phenomenon of becoming stuck may transpire on sandy terrain. Thus, a penalty term $T(\mathbf{x}_p)$ with a positive value is imposed when the robot is on sand, otherwise it is set to zero.

The information gain means how large area the robot can perceive around them. In order to traverses an unknown environment, robots needs to perceive the environment around them and map it. This study assumes that a hopper can observe the environments around the hopper by riding on a high place, such as a rock or a step. This paper describes the



(a) $t = 1, p = (1, 2)$(start) (b) $t = 35, p = (12, 6)$ (c) $t = 71, p = (18, 18)$(finish)

(d) $t = 1, p = (1, 2)$(start) (e) $t = 37, p = (12, 8)$ (f) $t = 75, p = (18, 18)$(finish)

(g) $t = 1, p = (1, 2)$(start) (h) $t = 39, p = (7, 13)$ (i) $t = 79, p = (18, 18)$(finish)

FIGURE 5. The simulation results of the greedy algorithm: (a) - (c) the safety-critical case ($(w_1, w_2) = (1, 0)$), (d) - (f) $S(x, u)$ and $I(x, u)$ have the same level of effect ($(w_1, w_2) = (0.5, 0.5)$), and (g) - (h) the information gain-critical case ($(w_1, w_2) = (0, 1)$) on hard ground. The green lines express the actual routes which the hopper traverses, and red lines show the useless actions. t and p denote the number of steps and the hopper position, respectively; Left figures: the initial state; Middle figures: the halfway state; Right figures: the finish.

information gain as:

$$I(\mathbf{x}_p) = h(x_p, y_p) \tag{5}$$

$D(x, u)$ is a penalty term in order to avoid robots failing to reach the goal. The value of the penalty decreases in proportion to the distance from the goal.

$$D(\mathbf{x}_p, \mathbf{u}) = -\frac{\|\mathbf{x}_g - \mathbf{x}_p\|}{\|\mathbf{x}_p\|} \tag{6}$$

where $\mathbf{x}_g$ denotes the vector to the goal.

We use min-max normalization to equalize the effect of $S(\mathbf{x}_p)$ and $I(\mathbf{x}_p)$, but do not apply the normalization to $D(\mathbf{x}_p, \mathbf{u})$, because the range is $-1 \leq D(\mathbf{x}_p, \mathbf{u}) \leq 0$.

IV. SIMULATION STUDY

This section presents the simulation study to evaluate the proposed algorithm described in Section III. We verify whether the robot can reach the goal by alternately repeating observations of the environment and locomotion in the simulations. The robot recognizes the surrounding environment. The sensing area is proportional to the height

TABLE 2. The action list of the hopper.

Action name	directions			
Moving orientations	North	South	East	West
Turning orientations	Turn right		Turn left	

of the position. The simulation environments are three kind surface: hard ground, sandy terrain, and partially sandy terrain. We evaluate the proposed path planning by changing the weight coefficient w_1 and w_2 in Eq. 3. We set $w_3 = 10$, because $D(\mathbf{x}_p, \mathbf{u})$ is treated as a penalty term. The value is determined empirically. The observation area is a circle of $2 \times \left(h(x_p, y_p) + 1\right)$ [m] radius from the robot.

A. THE ENVIRONMENT OF THE SIMULATION

Fig. 3shows the virtual rough terrain used in the simulation. The elevation of the surface is indicated with a colorbar. This terrain is modeled as a digital elevation map (DEM), of size 20 m × 20 m. Each state is a 1 m × 1 m cell. There are five elevated regions (i.e., positive values) and three depressed regions (i.e., negative values). The start point is $(x_p, y_p) = (1, 2)$, and the goal point is $(x_p, y_p) = (18, 18)$, respectively. The payoff at the goal is 100.

The best actions are calculated by MDPs in all possible states, and hence the path is generated by connecting the actions. The selected actions used in the simulations are shown in Table 2. This simulation assumes that the hopping distance $d' = 1.0$ m and $h' = 0.25$ m are constant at the hopping angle $\eta = 45$ [deg].

B. RESULTS AND DISCUSSION

The simulation terrains are three cases: hard ground, heterogeneous terrain (partly sandy soil), and sandy environment calculated by a desktop PC with Intel Core-i9, 8 cores, 3.7 GHz. All simulation scenarios were subjected to 100 repetitions, and the frequency of the robot achieving the objective, as well as the mean and standard deviation of the number of steps taken, were evaluated. The upper limit of steps is 150, and if the robot is unable to reach the objective within 150 steps, it is deemed to be in a ''deadlock'' state and the simulation is terminated. Similarly, if the robot exits the designated map, the simulation is considered a failure. The proposed algorithm works in an unknown environment with motion uncertainty, whereas the related works cannot. Thus, we compare the proposed algorithm with the greedy algorithm under identical conditions and using the same payoff function. The greedy algorithm chooses the action that leads to the highest reward within the adjacent traversable area. Furthermore, we employ the t-test, a parametric statistical test, to demonstrate the statistical significance between the proposed algorithm and the greedy algorithm. The values of the weight coefficients are selected by the results of our previous work [11].

TABLE 3. The number of successes and the average steps ± s.d. of the proposed algorithm on hard ground.

Objective function	Successes	Steps
Safety-critical case	99/100	75.7 ± 7.3
Intermediate case	98/100	75.0 ± 6.8
Information gain-critical case	99/100	76.0 ± 11.1

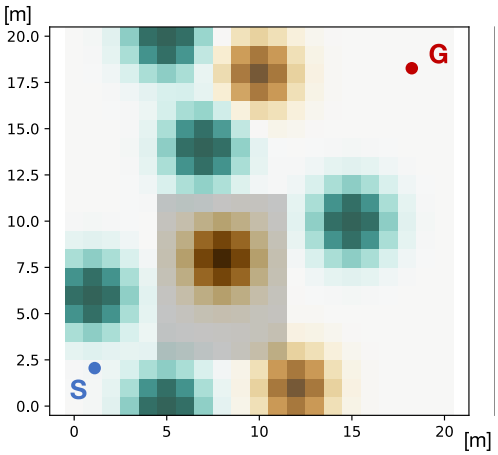


FIGURE 6. The the artificial terrain of the heterogeneous terrain. The sandy region is shown as translucent grey. The other conditions are the same as Sec. IV-B1.

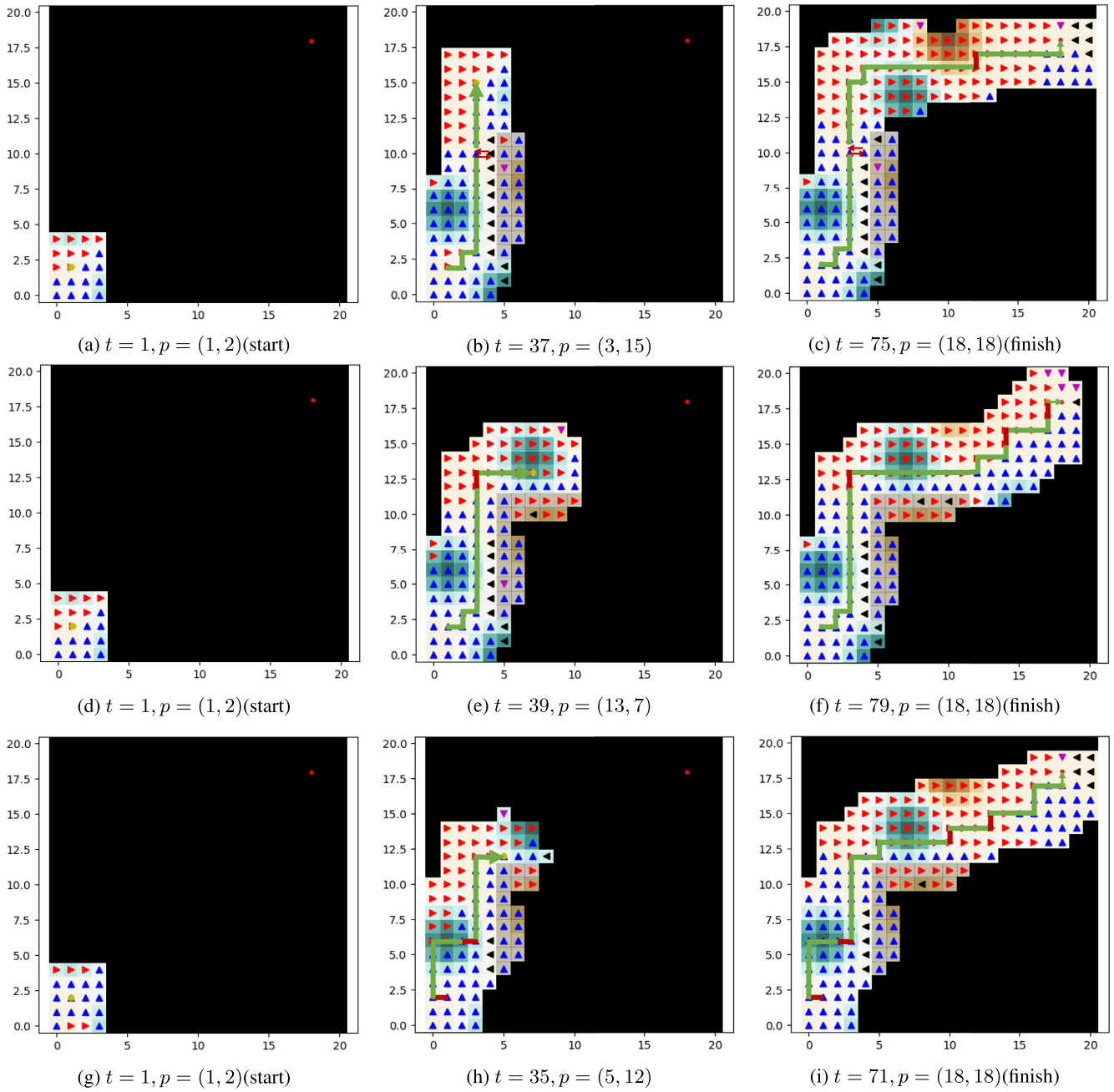
1) SIMULATIONS ON HARD GROUND

This work tries the three simulations by changing the weight coefficient in eq.(3) on hard ground: $(w_1, w_2) = (1, 0), (0.5, 0.5), (0, 1)$. The uncertainties of the hopping locomotion are defined as below:

- Selected action: $90\left(1 - \frac{\theta_{\text{front}}}{(\pi/6)}\right)$ %
- Staying: $90\frac{\theta_{\text{front}}}{(\pi/6)}$ %
- Turn left: $5\left(1 - \frac{\theta_{\text{right}}}{(\pi/6)}\right)$%
- Turn right: $5\left(1 - \frac{\theta_{\text{left}}}{(\pi/6)}\right)$%

where $\theta_{\text{direction}}$ denotes the slope angle to a direction of the selected action (front, left, and right). These probabilities are determined by hand-tuning. This simulation assumes the maximum angle of slope is 30 degrees, which the hopper can traverse, and the uncertainties are in inverse proportion to a slope angle. This is the reason why $\theta_{\text{direction}}$ divided by $(\pi/6)$. In addition, the probability define $1 - \frac{\theta_{\text{front}}}{(\pi/6)} := 0$ in the case of $\theta_{\text{direction}} \geq \pi/6$ in order the probability not to be negative values, and $\theta_{\text{left}} = -\theta_{\text{right}}$.

The simulation results of the proposed algorithm are shown in Fig. 4. The arrows indicate the best action at each state: blue, purple, red, and black arrows denote the moving direction to north, south, east, and west, respectively. The state of the hopper is represented by the yellow dot. The actual route traveled by the hopper is shown in green and red lines: green lines show the same path as the MDPs calculation, and the red lines show the different path. The average computing time until reaching the goal is about 44.3 ± 9.1 sec. per once, i.e., it takes approximately 0.6 seconds per step. Table 3 shows the number of successes and the average steps



(a) $t = 1, p = (1, 2)$(start) (b) $t = 37, p = (3, 15)$ (c) $t = 75, p = (18, 18)$(finish)

(d) $t = 1, p = (1, 2)$(start) (e) $t = 39, p = (13, 7)$ (f) $t = 79, p = (18, 18)$(finish)

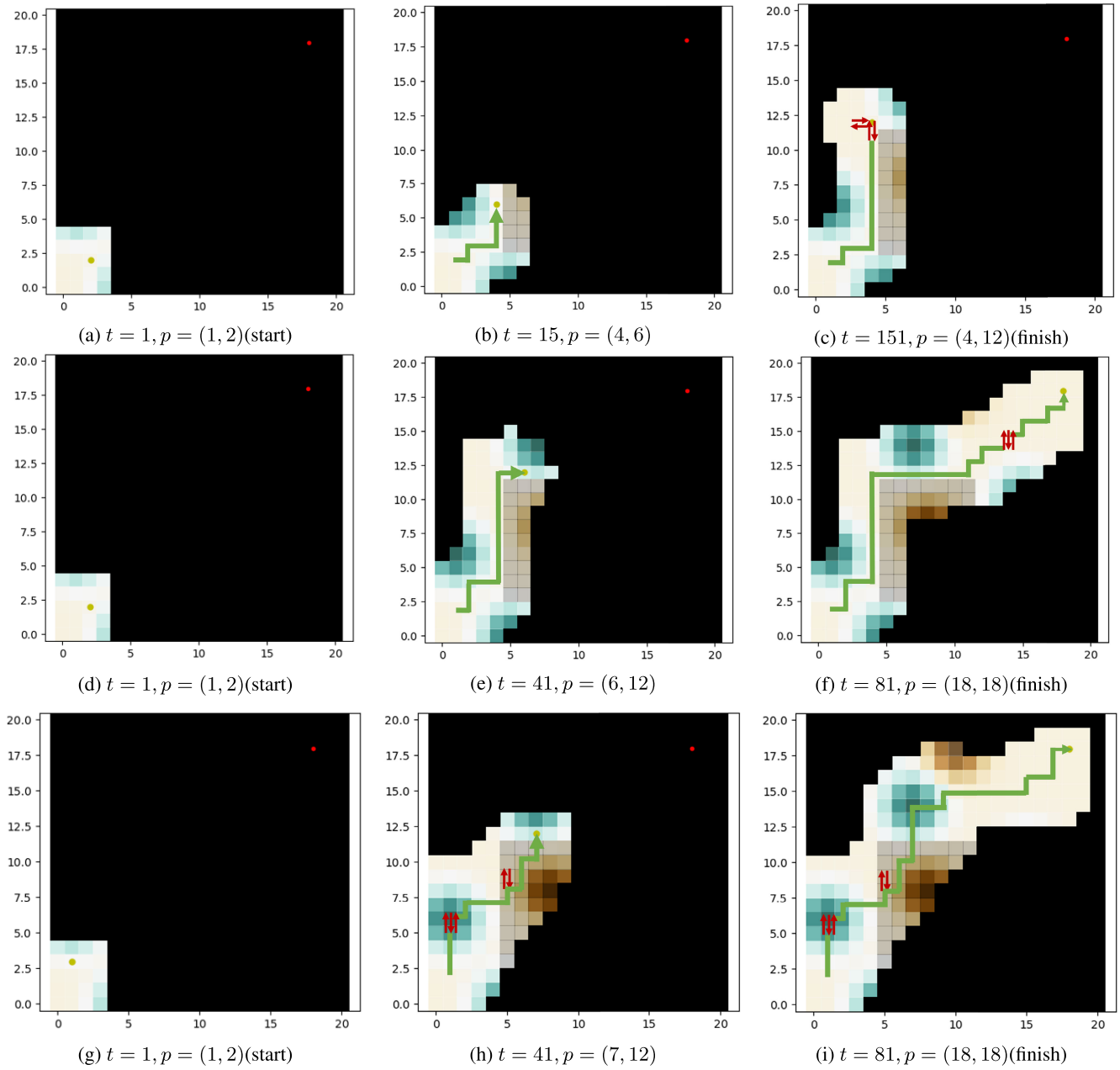
(g) $t = 1, p = (1, 2)$(start) (h) $t = 35, p = (5, 12)$ (i) $t = 71, p = (18, 18)$(finish)

FIGURE 7. The simulation results of the proposed algorithm: (a) - (c) the safety-critical case ($(w_1, w_2) = (1, 0)$), (d) - (f) $S(x, u)$ and $I(x, u)$ have the same level of effect ($(w_1, w_2) = (0.5, 0.5)$), and (g) - (h) the information gain-critical case ($(w_1, w_2) = (0, 1)$) on heterogeneous terrain; Left figures: the initial state; Middle figures: the halfway state; Right figures: the finish.

with standard deviation. The result shows that the proposed algorithm performed high success rates.

As shown in Figs. 4a to 4c, and Figs. 4d to 4f the hopper tends to moves on relatively flat terrain in the safety-critical case ($(w_1, w_2) = (1, 0)$), the intermediate case ($(w_1, w_2) = (0.5, 0.5)$). In the information gain-critical case, unlike in the above two cases, the hopper tends to traverse places as high as possible to get the information. This case has no penalties due to the slope of the terrain, so even if they go off the path, they head straight for the goal. Despite these differences, the number of successes, and the number of steps are almost the same in all cases. This result implies that the proposed algorithm generated the paths which are different routes, but the almost same distance in a homogeneous environment.

The results of the greedy algorithm are shown in Fig 5 and Table. 4. The green lines and red arrows denote the traveled paths and useless reciprocating motions, respectively. The average computing time until reaching the goal is about $(6.5 \pm 0.2) \times 10^{-3}$ sec. per once. The computing time is about 680 times faster than the proposed algorithm. Comparing



(a) $t = 1, p = (1, 2)$(start) (b) $t = 15, p = (4, 6)$ (c) $t = 151, p = (4, 12)$(finish)

(d) $t = 1, p = (1, 2)$(start) (e) $t = 41, p = (6, 12)$ (f) $t = 81, p = (18, 18)$(finish)

(g) $t = 1, p = (1, 2)$(start) (h) $t = 41, p = (7, 12)$ (i) $t = 81, p = (18, 18)$(finish)

FIGURE 8. The simulation results of the greedy algorithm: (a) - (c) the safety-critical case ($(w_1, w_2) = (1, 0)$), (d) - (f) $S(x, u)$ and $I(x, u)$ have the same level of effect ($(w_1, w_2) = (0.5, 0.5)$), and (g) - (h) the information gain-critical case ($(w_1, w_2) = (0, 1)$) on heterogeneous terrain. t and p denote the number of steps and the hopper position, respectively; Left figures: the initial state; Middle figures: the halfway state; Right figures: the finish. In the safety-critical case, the robot is in a deadlock and cannot reach the goal.

TABLE 4. The number of successes and the average steps ± s.d. of the greedy algorithm on hard ground.

Objective function	Successes	Steps
Safety-critical case	99/100	75.5 ± 9.7
Intermediate case	100/100	76.9 ± 6.3
Information gain-critical case	99/100	76.5 ± 7.1

these results with the results of the proposed algorithm, the generated paths and steps are almost the same, however, the computing time is one thousandth smaller than the proposed algorithm despite the value of the weight coefficients. This is because the greedy algorithm only chooses the largest payoff direction, but the proposed algorithm calculates the value iterations. On the other hand, in the information gain-critical case, the robot traversed the edge of the sunken place where the proposed algorithm doesn't generate the path. Table 5 shows the t-test between the greedy algorithm and the proposed algorithm. The null hypothesis for this test is that the proposed algorithm and the greedy algorithm generate the same results, and the null hypothesis is common hypothesis in this paper. The t-values of the safety-critical case and the information gain-critical case are larger than the

TABLE 5. The t-test results between the greedy algorithm and the proposed algorithm on hard ground.

Objective function	t-values	95% CI
Safety-critical case	0.165	1.97
Intermediate case	2.05	1.97
Information gain-critical case	0.762	1.97

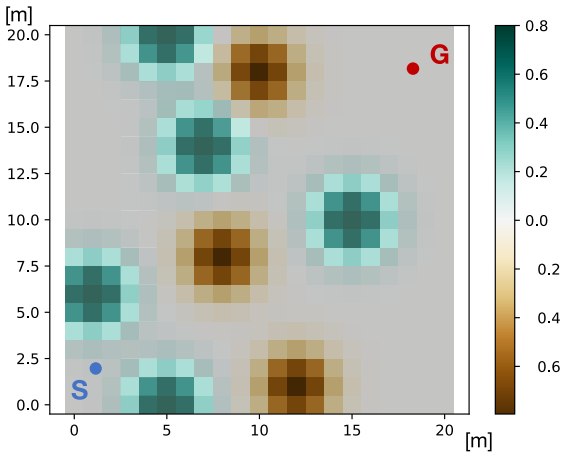


FIGURE 9. The artificial terrain of the sandy terrain. The sandy region is shown as translucent grey. The other conditions are the same as Sec. IV-B1.

TABLE 6. The number of successes and the average steps ± s.d. of the proposed algorithm on heterogeneous terrain.

Objective function	Successes	Steps
Safety-critical case	100/100	77.5 ± 6.6
Intermediate case	100/100	76.5 ± 6.8
Information gain-critical case	99/100	76.3 ± 7.6

95% CI value, which rejects the null hypothesis. This result shows that the proposed algorithm generates the better path than the greedy algorithm in this case. The both algorithms perform almost the same efficiency, but the greedy algorithm is better than the proposed algorithm in terms of calculation cost. The computational time of the proposed algorithm is approximately 0.6 seconds per each action by 3.7 GHz CPU. The time can be calculated to be approximately 1.5 seconds using 1.5 GHz CPU, such as a Raspberry Pi 4. The computational time is enough fast to be implemented in a real application.

2) SIMULATIONS ON HETEROGENEOUS TERRAIN

In this case, a part of this terrain ($5 \leq x \leq 11, 3 \leq y \leq 11$) is sandy surface. We assume the hopping distance and height decrease on sand: the hopping distance $d' = 0.5$ m and height $h' = 0.125$ m at the hopping angle $\eta = 45$ deg. Furthermore, slip is easier to occur on sandy surface than on hard ground. This is why the safety cost Eq. (4) includes the penalty $T(\boldsymbol{x}) = -5$. Therefore, the motion uncertainties on sandy terrain are defined as follows:

- Selected action: $70\left(1 - \frac{\theta_{\text{front}}}{(\pi/6)}\right)$ %
- Staying: $70\frac{\theta_{\text{front}}}{(\pi/6)} + 10\%$
- Turn left: $10\left(1 - \frac{\theta_{\text{right}}}{(\pi/6)}\right)\%$
- Turn right: $10\left(1 - \frac{\theta_{\text{left}}}{(\pi/6)}\right)\%$

There is a risk of stuck in sand regardless of the slope angle, which is expressed as the 10% possibility in the staying action. The probabilities on hard ground and the weight coefficients are the same as the section IV-B1. The environment is shown in Fig. 6.

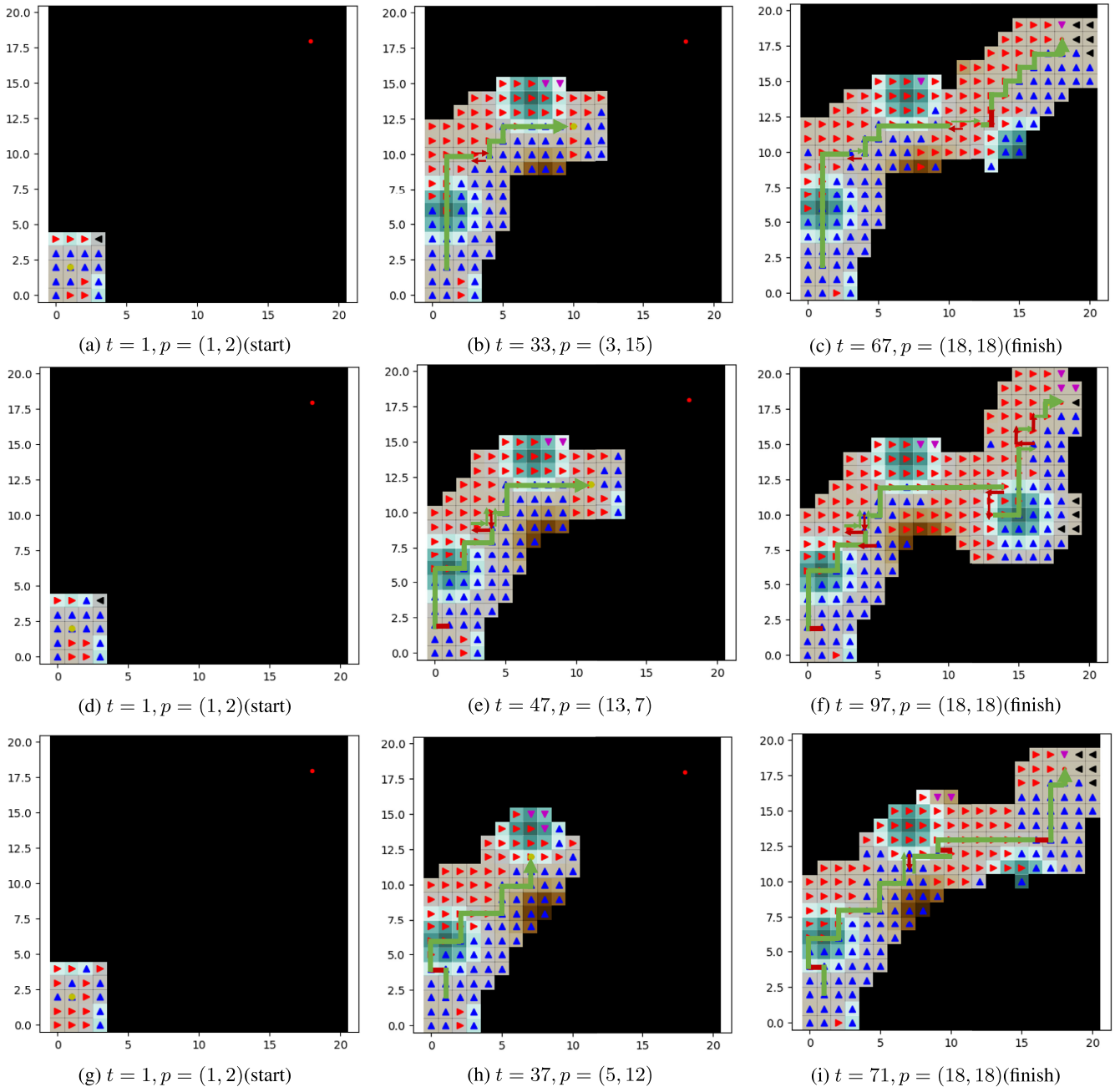
The results of the proposed algorithm are shown in Fig. 7. The sandy places are illustrated as translucent grey. The average computing time until reaching the goal is about 42.1 ± 7.3 sec. per once. Table 6 shows the number of successes and the average steps with standard deviation. The success rates are 100% in two cases, which shows the proposed algorithm works well in this situation.

In all the cases, the number of successes and the number of steps are similar and higher than Section IV-B1. Figure 7 shows that the generated paths are similar in all the cases, which trend differs from the results in Sec. IV-B1. This is because the hopper has moved upward from the initial state in order to avoid the sandy area. The hopper has traversed relatively flat terrain in the safety-critical case ($(w_1, w_2) = (1, 0)$), and places as high as possible in the other cases, especially in the information gain-critical case ($(w_1, w_2) = (0, 1)$). These trends are the similar to the results of Section IV-B1. The number of steps of the safety-critical case is only slightly larger than the results of Section IV-B1. As shown in Fig.7b, the reason is that when the hopper moves toward the sandy area unintentionally, the robot takes extra steps in order to go away from the area.

The results of the greedy algorithm are shown in Fig 8 and Table. 7. The average computing time until reaching the goal is about $(6.8 \pm 0.3) \times 10^{-3}$ [sec] per once. In the safety-critical case ($(w_1, w_2) = (1, 0)$), the success rate is exceedingly low, and even if the robot can reach the goal, it takes more steps than in the other cases. The reason why the success rate is low is that the robot is in a deadlock, as shown in Fig.8c. On the other hand, such a deadlock has not appeared in the results of the proposed algorithm, which indicates the effectiveness of the proposed algorithm. In the other cases, the success rate is almost the same, but the steps are larger than the proposed algorithm. The robot traversed near the sandy places and sometimes moved to the sandy places, which increases steps and uncertainty. The computational time is almost same as the Sec. IV-B1. Table 8 shows the t-test result of this results between the proposed algorithm and the greedy algorithm. The result also rejects the null hypothesis which is defined in the Sec. IV-B1. These results show that the proposed algorithm demonstrates better performance in an environment that has a sandy area than the greedy algorithm.

3) SIMULATIONS ON SANDY TERRAIN

This section evaluates the performance of the two algorithms on sandy ground. This simulation environment is covered with sand except the place where the height is $h(\mathbf{x}) \geq 0.1$. As described above, hopping performance decreases on sandy terrain in general. The simulation conditions are the



(a) $t = 1, p = (1, 2)$(start) (b) $t = 33, p = (3, 15)$ (c) $t = 67, p = (18, 18)$(finish)

(d) $t = 1, p = (1, 2)$(start) (e) $t = 47, p = (13, 7)$ (f) $t = 97, p = (18, 18)$(finish)

(g) $t = 1, p = (1, 2)$(start) (h) $t = 37, p = (5, 12)$ (i) $t = 71, p = (18, 18)$(finish)

FIGURE 10. The simulation results: (a) - (c) the safety-critical case ($(w_1, w_2) = (1, 0)$), (d) - (f) $S(x, u)$ and $I(x, u)$ have the same level of effect ($(w_1, w_2) = (0.5, 0.5)$), and (g) - (h) the information gain-critical case ($(w_1, w_2) = (0, 1)$) on sandy terrain; Left figures: the initial state; Middle figures: the halfway state; Right figures: the finish.

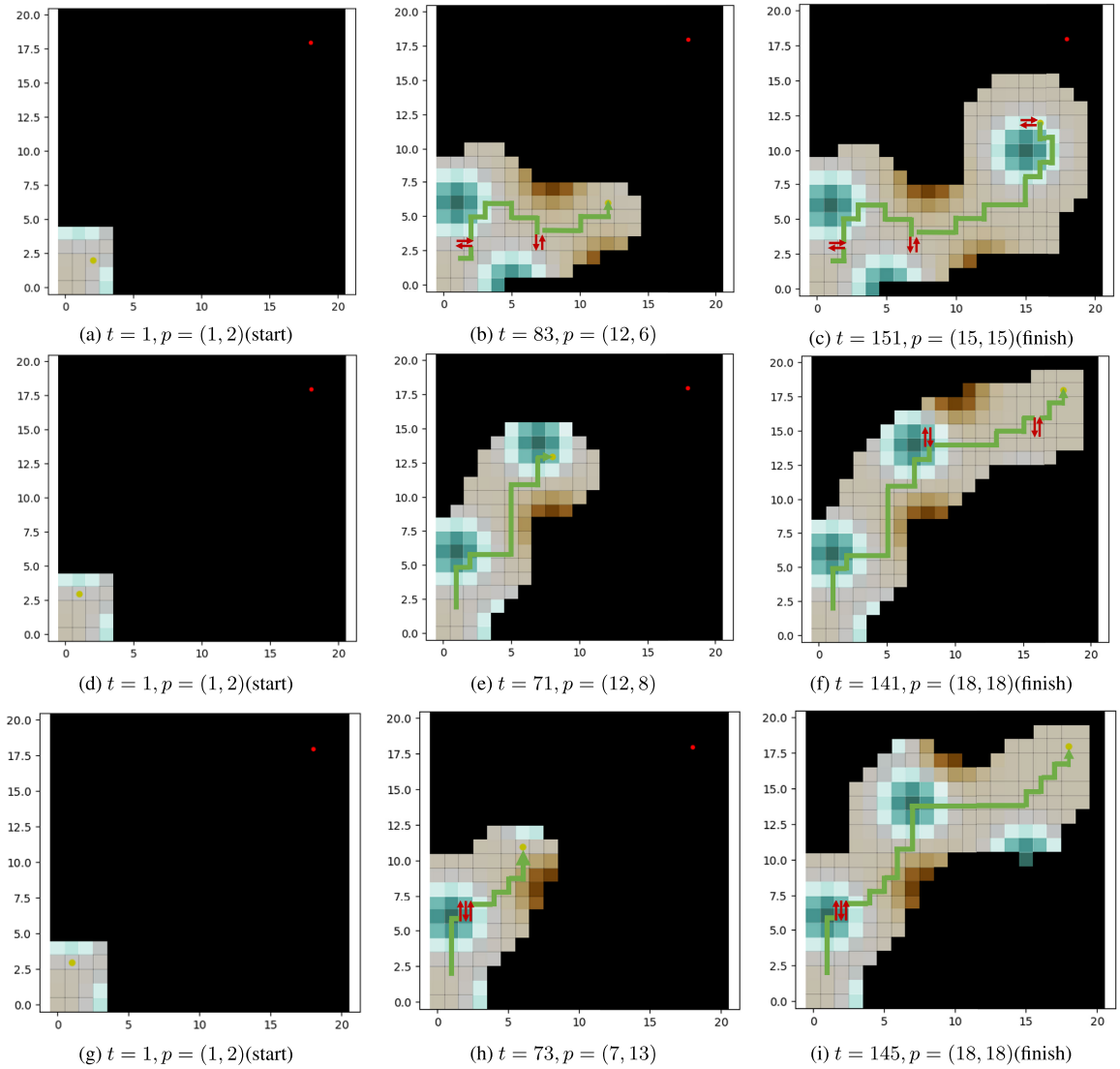
TABLE 7. The number of successes and the average steps ± s.d. of the greedy algorithm on heterogeneous ground.

Objective function	Successes	Steps
Safety-critical case	5/100	107.2 ± 9.7
Intermediate case	99/100	92.6 ± 10.5
Information gain-critical case	99/100	83.0 ± 10.1

same as the Section IV-B1 and IV-B2. The environment is shown in Fig. 9.

The results are shown in Fig. 10. The sandy terrains are displayed as translucent grey. The computing time is about 62.1 ± 3.7 sec. These results are similar in spite of the difference in the weight coefficients. The hopper tends to approach the elevated places from sandy terrain to get better rewards. In other words, the hopper go to only the hills to get the reward because most of the terrains are covered with sand. This may be deduced that the ratio changes of w_1 and w_2 produce only the value changes of the reward on the hills. Therefore, the hopper had chose the similar routes in all cases.

From Table. 9, the success rate is a bit smaller and the steps are larger than the other results. The reason is



(a) $t = 1, p = (1, 2)$(start) (b) $t = 83, p = (12, 6)$ (c) $t = 151, p = (15, 15)$(finish)

(d) $t = 1, p = (1, 2)$(start) (e) $t = 71, p = (12, 8)$ (f) $t = 141, p = (18, 18)$(finish)

(g) $t = 1, p = (1, 2)$(start) (h) $t = 73, p = (7, 13)$ (i) $t = 145, p = (18, 18)$(finish)

FIGURE 11. The simulation results of the greedy algorithm: (a) - (c) the safety-critical case ($(w_1, w_2) = (1, 0)$), (d) - (f) $S(x, u)$ and $I(x, u)$ have the same level of effect ($(w_1, w_2) = (0.5, 0.5)$), and (g) - (h) the information gain-critical case ($(w_1, w_2) = (0, 1)$) on sandy terrain. t and p denote the number of steps and the hopper position, respectively; Left figures: the initial state; Middle figures: the halfway state; Right figures: the finish. In the safety-critical case, the robot is in a deadlock and cannot reach the goal.

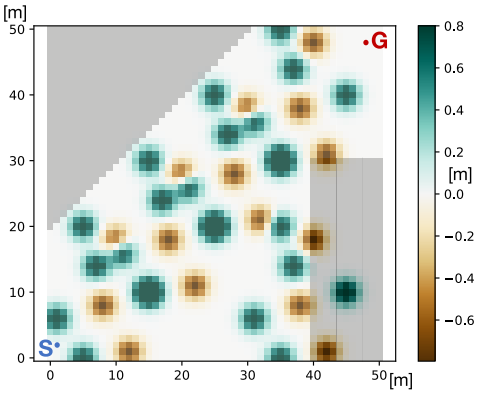


FIGURE 12. The complicated large environment. The translucent grey area is sandy terrain.

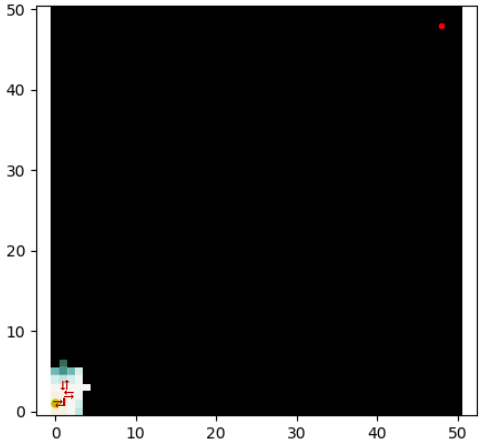
indicated that the low performance of the hopping in sandy environment increases the steps. In addition, the uncertainties

TABLE 8. The t-test results between the greedy algorithm and the proposed algorithm on heterogeneous ground.

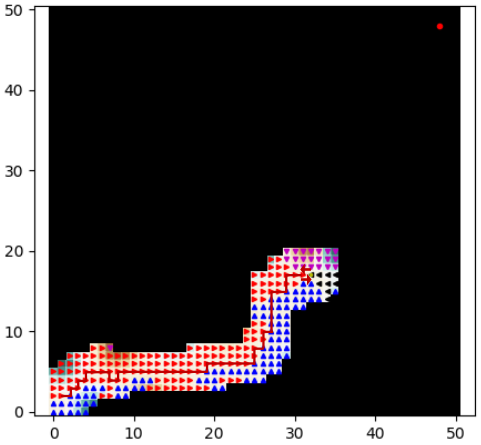
Objective function	t-values	95% CI
Safety-critical case	9.61	1.98
Intermediate case	12.8	1.97
Information gain-critical case	5.27	1.97

of motion are higher on sandy terrain than on hard ground (the probabilities are described in the Sec. IV-B1 and IV-B2), which deteriorates the performance of the path following. Consequently, the steps increase and the robot cannot arrive at the goal within 150 steps in some cases.

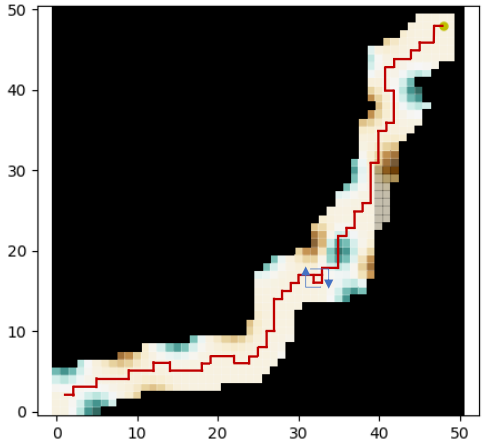
The results of the greedy algorithm are shown in Fig 11 and Table. 10. The average computing time until reaching the goal is about $(7.0 \pm 0.4) \times 10^{-2}$ [sec] per once. The success rates are smaller and the steps are rather larger than our method, especially in the safety-critical case ($(w_1, w_2) = (1, 0)$).



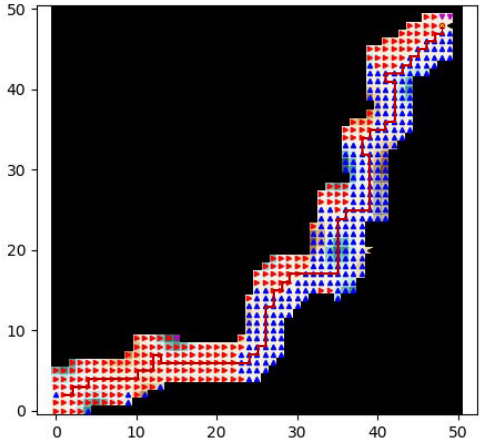
(a) The result of the safety-critical case of the greedy algorithm.



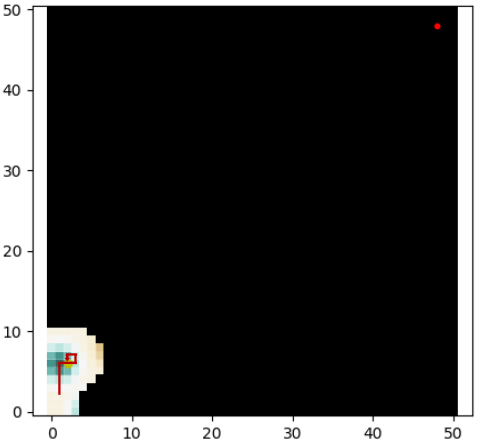
(b) The result of the safety-critical case of the proposed algorithm.



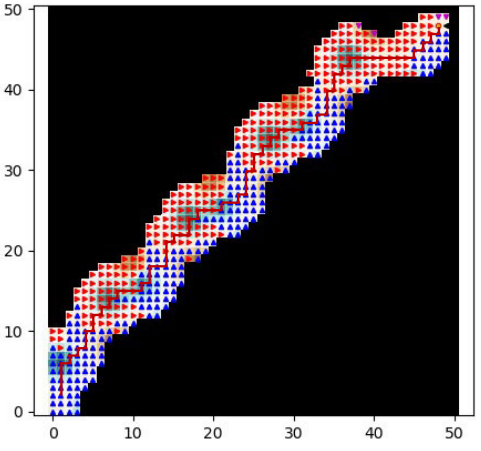
(c) The result of the intermediate case of the greedy algorithm.



(d) The result of the intermediate case of the proposed algorithm.



(e) The result of the information gain-critical case of the greedy algorithm.



(f) The result of the information gain-critical case of the proposed algorithm.

FIGURE 13. The results of the greedy algorithm (left) and the proposed algorithm (right): (a), (b) the safety-critical case ($(w_1, w_2) = (1, 0)$), (c), (d) $S(x, u)$ and $I(x, u)$ have the same level of effect ($(w_1, w_2) = (0.5, 0.5)$), and (e), (f) the information gain-critical case ($(w_1, w_2) = (0, 1)$). The blue, purple, red, and black arrows denote the moving direction to north, south, east, and west, respectively. The red lines indicate the actual paths on which the hopper moved.

In such the case, the robot takes a detour, or reciprocating motion because only safety is considered. As a result, the steps increases and the robot cannot reach the goal within the 150 steps as shown in Fig. 11c. In the other cases, the increase of the steps also reduce the success rates. In terms of the computational time, it is almost same as the other results. The t-test result is shown in the Table 11, which also rejects the null hypothesis. According to the above results, the comparisons present the effectiveness of our method in sandy environment.

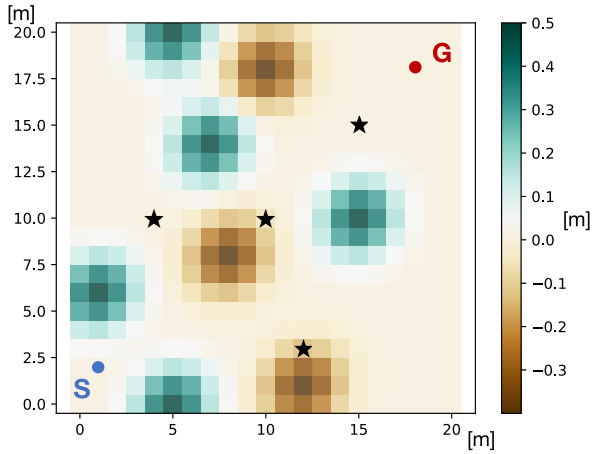


FIGURE 14. The treasures on the artificial terrain. The treasures are plotted as black star. The environment is same as Sec. IV-B1.

TABLE 9. The number of successes and the average steps ± s.d. of the proposed algorithm on sandy terrain.

Objective function	Successes	Steps
Safety-critical case	81/100	111.7 ± 10.7
Intermediate case	92/100	114.9±12.2
Information gain-critical case	93/100	117.7±11.3

TABLE 10. The number of successes and the average steps ± s.d. of the greedy algorithm on sandy terrain.

Objective function	Successes	Steps
Safety-critical case	41/100	127.7 ± 13.6
Intermediate case	86/100	120.1±12.8
Information gain-critical case	89/100	121.2±12.4

TABLE 11. The t-test results between the greedy algorithm and the proposed algorithm on sandy terrain.

Objective function	t-values	95% CI
Safety-critical case	7.03	1.98
Intermediate case	2.78	1.97
Information gain-critical case	1.98	1.97

4) SIMULATIONS ON COMPLICATED LARGE ENVIRONMENT

In this section we present simulation studies in larger and more complex environments. Figure 12 shows the simulation environment. The translucent grey area indicates the sandy terrain, and the color bar represents the height of the terrain. The start point is at (1, 2) and the finish point is at (48, 48). The upper limit of steps is 800. Other conditions are the same as in the previous section. The simulation results are shown in Fig. 13. The results of the greedy algorithm are displayed on the left side, and the results of the proposed algorithm are on the right side. Tables 12 and 13 represent the success numbers and steps for each case, respectively. The average computing time of the proposed algorithm and the greedy algorithm are 309.3 ± 4.3 sec. and $(7.5 \pm 0.5) \times 10^{-2}$ sec., respectively. These results demonstrate that the proposed algorithm also performs better than the greedy algorithm in this environment. In particular, in the information gain-critical case and the intermediate case, the success rate of the proposed method is significantly higher than that of the conventional method, and the number of steps is less than half. Figure 13 also shows that the paths reflect the distribution of the weight coefficients for each evaluation function. In the safety-critical case, the greedy algorithm failed to reach the goal in all cases and the proposed algorithm succeeded only five times, which is a low result. This is because the safety-critical evaluation function generates conservative actions, and the feasibility of getting stuck in a large area is considered to be higher than in the small environment.

TABLE 12. The number of successes and the average steps ± s.d. of the proposed algorithm in the complicated large environment.

Objective function	Successes	Steps
Safety-critical case	5/100	425.5 ± 105.7
Intermediate case	100/100	222.6±13.1
Information gain-critical case	95/100	216.8±12.4

TABLE 13. The number of successes and the average steps ± s.d. of the greedy algorithm in the complicated large environment.

Objective function	Successes	Steps
Safety-critical case	0/100	–
Intermediate case	47/100	527.3±155.4
Information gain-critical case	3/100	583.3±108.5

TABLE 14. The t-test results between the greedy algorithm and the proposed algorithm in the complicated large environment.

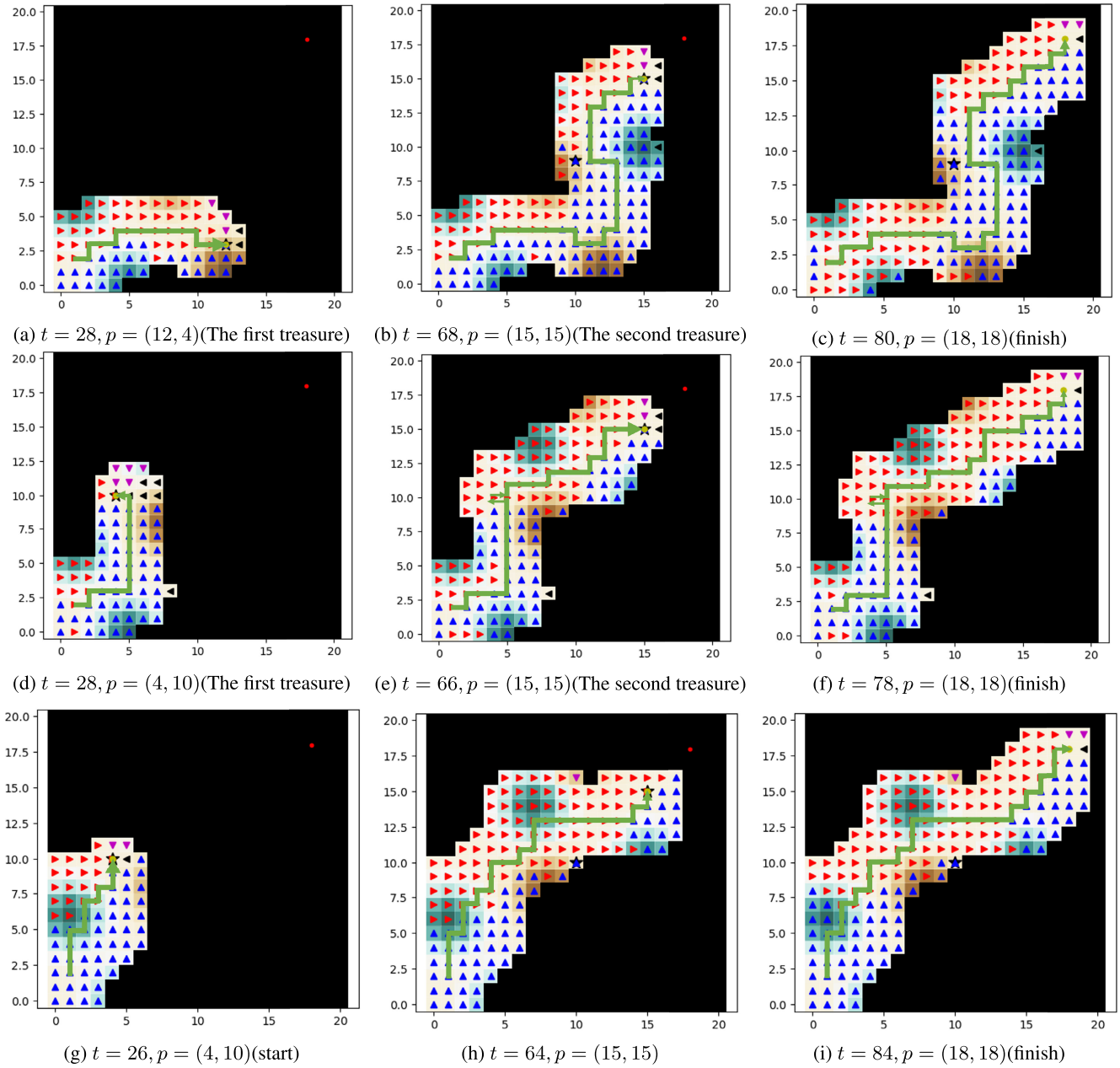
Objective function	t-values	95% CI
Safety-critical case	–	1.98
Intermediate case	19.54	1.97
Information gain-critical case	31.42	1.97

Table 14 presents the t-test results. The t-test rejects the null hypothesis, which is defined in Sec. IV-B1.

5) TREASURE HUNTING ON HARD GROUND

This section presents the performance of treasure hunting. In an actual planetary exploration mission, when a rover finds an attractive object during locomotion to the goal, the rover might be required to retrieve the object even if the object is not on the path. In this case, the rover must decide whether to re-generate a path to the object or to give up getting the object because of the safety criteria. The tested environment and the conditions of the hopper are the same as in Sec. IV-B1, because the proposed algorithm and the greedy method perform well in the environment. The treasure positions are (12, 4), (10, 10), (4, 10), (15, 15), respectively. The environment is shown in Fig. 14. The black stars denote the treasures.

The results of the proposed algorithm are shown in Fig. 15 and Table 15. The average computing time is 50.7 ± 5.5 sec. per one episode. Compared to the results presented in Section IV-B1, it was observed that the average number of steps and computational time increased, as the robot took more steps to find the treasures. However, the goal-reaching rate remained consistent with the results obtained using the proposed algorithm in Section IV-B1. In all cases, the most common outcome was the acquisition of two treasures.



(a) $t = 28, p = (12, 4)$(The first treasure) (b) $t = 68, p = (15, 15)$(The second treasure) (c) $t = 80, p = (18, 18)$(finish)

(d) $t = 28, p = (4, 10)$(The first treasure) (e) $t = 66, p = (15, 15)$(The second treasure) (f) $t = 78, p = (18, 18)$(finish)

(g) $t = 26, p = (4, 10)$(start) (h) $t = 64, p = (15, 15)$ (i) $t = 84, p = (18, 18)$(finish)

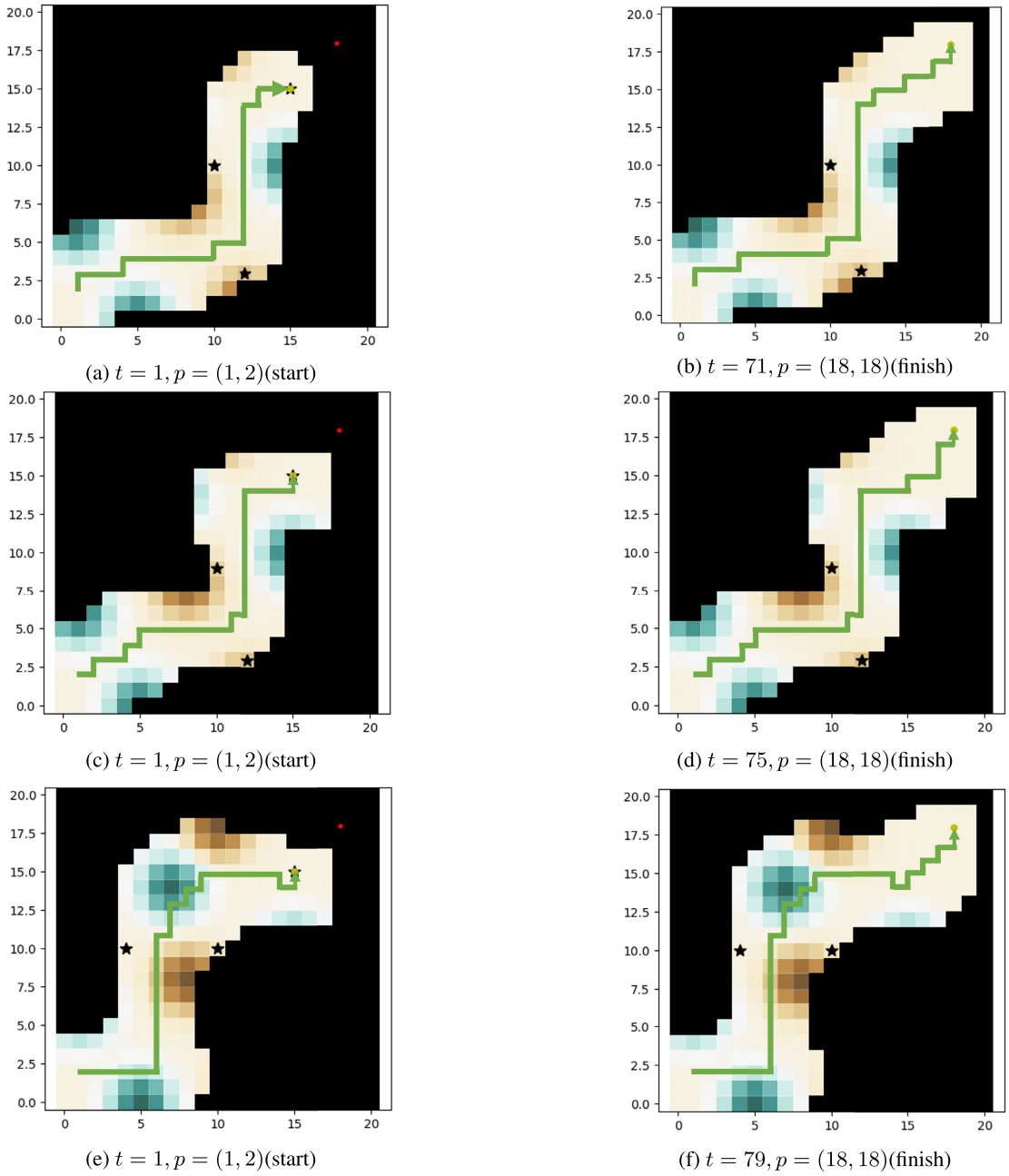
FIGURE 15. The results of the treasure hunting by the proposed algorithm: (a) - (c) the safety-critical case ($(w_1, w_2) = (1, 0)$), (d) - (f) $S(x, u)$ and $I(x, u)$ have the same level of effect ($(w_1, w_2) = (0.5, 0.5)$), and (g) - (h) the information gain-critical case ($(w_1, w_2) = (0, 1)$). The blue, purple, red, and black arrows denote the moving direction to north, south, east, and west, respectively. The green lines indicate the correct paths on which the hopper moved. t and p denote the number of steps and the hopper position, respectively; Left figures: the first treasure; Middle figures: the second treasure; Right figures: the finish.

TABLE 15. The number of successes, getting treasures (and the number of episodes), and the average steps ± s.d. of the proposed algorithm on hard ground.

Objective function	Successes	Getting treasures (episodes)	Steps
Safety-critical case	99/100	2(82), 1(17)	88.4 ± 8.1
Intermediate case	100/100	2(93), 1(7)	91.5±9.0
Information gain-critical case	98/100	3(2), 2(96)	78.0±7.9

TABLE 16. The number of successes, getting treasures (the number of episodes), and the average steps ± s.d. of the greedy algorithm on hard ground.

Objective function	Successes	Getting treasures (episodes)	Steps
Safety-critical case	91/100	2(2), 1(89)	78.5 ± 7.1
Intermediate case	100/100	2(19), 1(81)	79.1±6.9
Information gain-critical case	100/100	2(10), 1(90)	77.8±7.2



(a) $t = 1, p = (1, 2)$(start)
(b) $t = 71, p = (18, 18)$(finish)
(c) $t = 1, p = (1, 2)$(start)
(d) $t = 75, p = (18, 18)$(finish)
(e) $t = 1, p = (1, 2)$(start)
(f) $t = 79, p = (18, 18)$(finish)

FIGURE 16. The results of the treasure hunting by the greedy algorithm: (a), (b) the safety-critical case ($(w_1, w_2) = (1, 0)$), (c), (d) $S(x, u)$ and $I(x, u)$ have the same level of effect ($(w_1, w_2) = (0.5, 0.5)$), and (e), (f) the information gain-critical case ($(w_1, w_2) = (0, 1)$). The green lines express the actual routes which the hopper traverses. t and p denote the number of steps and the hopper position, respectively; Left figures: the first treasure; Right figures: the finish.

TABLE 17. The t-test results of the number of getting treasures between the greedy algorithm and the proposed algorithm.

Objective function	t-values	95% CI
Safety-critical case	9.86	1.97
Intermediate case	5.76	1.97
Information gain-critical case	8.93	1.97

In the information-gain critical scenario, there were two cases where the robot successfully located three treasures. Although Fig. 15 shows the route that the robot traversed is similar to Fig. 4 in Sec.IV-B1, the robot took a winding route in order to get the treasures. These results indicate that the proposed algorithm can collect the treasures without decreasing the goal-reaching rate.

Figure 16 and Table 16 show the results of the greedy algorithm. The average computing time is $(7.0 \pm 0.2) \times 10^{-3}$ sec. The steps, computing time, and the goal-reaching rate are almost the same as in Fig. 5 in Sec.IV-B1, except for the goal-reaching rate in the safety-critical case. The most frequent events were those in which one treasure was obtained. Therefore, two results are shown in each instance

in Fig. 16. If the robot traverses adjacent to the treasure, the robot obtains the treasure, owing to the characteristic of the greedy algorithm. The routes are also similar to the results of the greedy algorithm in Sec.IV-B1. The t-test results are shown in Table 17. The result rejects the null hypothesis. The comparisons of the results imply that the proposed algorithm performs better than the greedy algorithm. Furthermore, the proposed algorithm can deal with sudden requests such as ''treasure hunts'' without performance degradation.

V. CONCLUSION

This paper presents a hopping path planning algorithm for exploration in unknown environments. The main contributions of this paper are the follows:

- The proposed method is a 2.5-dimensional path planning method that takes into account hopping trajectories.
- This paper demonstrated the robustness of the proposed method to generate paths in various unknown environments.

The proposed algorithm uses Markov Decision Processes (MDPs) to determine the optimal action in all possible observed states. One of the key features of the proposed algorithm is its adaptive payoff function, which can be adjusted to meet different mission requirements. The path is generated by sequentially connecting actions in known areas, using the robot's locomotion capabilities. The algorithm is defined as a 2.5D path planning method, as actions are calculated in 2D and terrain heights are used for collision detection. The effectiveness of the proposed algorithm is evaluated through comparison of generated paths in virtual environments, as well as through metrics such as goal attainment rate, number of steps taken, and comparison with greedy algorithms. The simulation results showed the following contributions of the proposed algorithm: it achieved a high success rate and was able to avoid deadlocks in all cases. In contrast, the greedy algorithm was unable to avoid deadlocks, especially in heterogeneous environments. The proposed algorithm required fewer steps to reach the goal than the greedy algorithm, especially on sandy terrain. The proposed algorithm also outperformed the greedy algorithm in more complicated large environments. Furthermore, the proposed algorithm demonstrated superior handling of sudden events, such as treasure hunts, compared to the greedy algorithm. These contributions illustrate the robustness of the proposed algorithm for path generation and adaptive actions in unknown environments.

As future work, we plan to conduct physical experiments on rough terrains, including sandy terrain, to validate the effectiveness of our approach and identify areas for improvement in real-world applications. One limitation of Markov Decision Processes is the computational cost, which is proportional to the size of the known area in which actions are generated. For example, in the environment shown in Fig. 3, when all the regions are known, the time taken to generate the path from the start to the destination using the proposed method, A* and D* Lite is 0.065 second, 0.005 seconds, and 0.016 seconds respectively. This environment is not assumed in this study and the advantage of the proposed method - its ability to generate actions to the goal in an unknown environment - is not utilized. Nevertheless, this example illustrates the high computational load of MDP. To address these issues, we aim to reduce the computational cost or develop a new approach by improving MDPs or by combining the approaches discussed in Section II.

REFERENCES

[1] H. Tsukagoshi, M. Sasaki, A. Kitagawa, and T. Tanaka, ''Design of a higher jumping rescue robot with the optimized pneumatic drive,'' in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 1276–1283.

[2] R. G. Reid, L. Roveda, I. A. Nesnas, and M. Pavone, ''Contact dynamics of internally-actuated platforms for the exploration of small solar system bodies,'' in *Proc. Int. Symp. Artif. Intell., Robot. Automat. Space (i-SAIRAS)*, 2014, p. 9.

[3] D. Mège, J. Gurgurewicz, J. Grygorczuk, Ł. Wiśniewski, and G. Thornell, ''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*, vol. 121, pp. 200–220, Apr. 2016.

[4] S. Montminy, E. Dupuis, and H. Champliaud, ''Mechanical design of a hopper robot for planetary exploration using SMA as a unique source of power,'' *Acta Astronautica*, vol. 62, nos. 6–7, pp. 438–452, Mar. 2008.

[5] T. Yoshimitsu, ''Development of autonomous rover for asteroid surface exploration,'' in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Apr. 2004, pp. 2529–2534.

[6] T. Yoshimitsu and T. Kubota, ''Engineering challenges and results by MINERVA-II asteroid surface rovers,'' *J. Robot. Soc. Jpn.*, vol. 38, no. 8, pp. 754–761, 2020.

[7] Y. Nakauchi, T. Maeda, K. Saiki, T. Yoshimitsu, M. Ohtake, M. Otsuki, H. Nagaoka, H. Sato, H. Shiraishi, C. Honda, K. Yoshikawa, C. Yamanaka, Y. Ishihara, and Y. Kunii, ''Development status of slim onboard near-infrared spectrometer and small lunar probe,'' Inst. Space Astron. Sci., Japan Aerosp. Explor. Agency(JAXA)(ISAS), Tokyo, Japan, Tech. Rep. sA6000163137, g24-2, 2021.

[8] Cabinet Office. (2023). *Moonshot Goal 3 Realization of AI Robots That Autonomously Learn, Adapt to Their Environment, Evolve in Intelligence and Act Alongside Human Beings, by 2050*. Accessed: May 18, 2023. [Online]. Available: https://www8.cao.go.jp/cstp/moonshot/sub3.html

[9] *Minerva-II1: Images From the Surface of Ryugu*. Accessed: May 20, 2023. [Online]. Available: http://www.hayabusa2.jaxa.jp/en/topics/20180927e_MNRV/

[10] K. Sakamoto, M. Otsuki, T. Maeda, K. Yoshikawa, and T. Kubota, ''Evaluation of hopping robot performance with novel foot pad design on natural terrain for hopper development,'' *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3294–3301, Oct. 2019.

[11] K. Sakamoto and T. Kubota, ''Hopping path planning in uncertain environments for planetary explorations,'' *ROBOMECH J.*, vol. 9, no. 1, pp. 1–15, Dec. 2022.

[12] A. Stentz, ''Optimal and efficient path planning for partially known environments,'' in *Intelligent Unmanned Ground Vehicles*. Cham, Switzerland: Springer, 1997, pp. 203–220.

[13] S. Koenig and M. Likhachev, ''D* lite,'' in *Proc. AAAI*, vol. 15, 2002, pp. 476–483.

[14] D. Ferguson and A. Stentz, ''Using interpolation to improve path planning: The field D* algorithm,'' *J. Field Robot.*, vol. 23, no. 2, pp. 79–101, 2006.

[15] J. Carsten, A. Rankin, D. Ferguson, and A. Stentz, ''Global path planning on board the Mars exploration rovers,'' in *Proc. IEEE Aerosp. Conf.*, Mar. 2007, pp. 1–11.

[16] J. Han, ''A surrounding point set approach for path planning in unknown environments,'' *Comput. Ind. Eng.*, vol. 133, pp. 121–130, Jul. 2019.

[17] B. D. Luders, S. Karaman, and J. P. How, ''Robust sampling-based motion planning with asymptotic optimality guarantees,'' in *Proc. AIAA Guid., Navigat., Control (GNC) Conf.*, Aug. 2013, p. 5097.

[18] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, ''An efficient sampling-based method for online informative path planning in unknown environments,'' *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1500–1507, Apr. 2020.

[19] K. Cai, C. Wang, S. Song, H. Chen, and M. Q.-H. Meng, "Risk-aware path planning under uncertainty in dynamic environments," *J. Intell. Robotic Syst.*, vol. 101, no. 3, pp. 1–15, Mar. 2021.
[20] J. Wang, T. Li, B. Li, and M. Q.-H. Meng, "GMR-RRT*: Sampling-based path planning using Gaussian mixture regression," *IEEE Trans. Intell. Vehicles*, vol. 7, no. 3, pp. 690–700, Sep. 2022.
[21] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.
[22] K. Ok, S. Ansari, B. Gallagher, W. Sica, F. Dellaert, and M. Stilman, "Path planning with uncertainty: Voronoi uncertainty fields," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 4596–4601.
[23] B. Nakisa, M. N. Rastgoo, and M. Z. A. Nazri, "Target searching in unknown environment of multi-robot system using a hybrid particle swarm optimization," *J. Theor. Appl. Inf. Technol.*, vol. 96, no. 13, pp. 4055–4065, 2018.
[24] F. A. Raheem, "Path planning algorithm using D* heuristic method based on PSO in dynamic environment," *Amer. Academic Sci. Res. J. Eng., Technol., Sci.*, vol. 49, no. 1, pp. 257–271, 2018.
[25] E. Krell, A. Sheta, A. P. R. Balasubramanian, and S. A. King, "Collision-free autonomous robot navigation in unknown environments utilizing PSO for path planning," *J. Artif. Intell. Soft Comput. Res.*, vol. 9, no. 4, pp. 267–282, Oct. 2019.
[26] S. Hu, W. Chen, M. Wu, T. Liao, and H. Chou, "Graph-based path planning and ABC-optimized IT2FLS for autonomous mobile robot exploration within unknown environments," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2021, pp. 1345–1350.
[27] B. Hockman and M. Pavone, "Stochastic motion planning for hopping rovers on small solar system bodies," in *Robotics Research*. Cham, Switzerland: Springer, 2020, pp. 877–893.
[28] L. Chang, L. Shan, C. Jiang, and Y. Dai, "Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment," *Auto. Robots*, vol. 45, no. 1, pp. 51–76, Jan. 2021.
[29] S. M. Sombolestan, A. Rasooli, and S. Khodaygan, "Optimal path-planning for mobile robots to find a hidden target in an unknown environment based on machine learning," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 5, pp. 1841–1850, May 2019.
[30] X. Lei, Z. Zhang, and P. Dong, "Dynamic path planning of unknown environment based on deep reinforcement learning," *J. Robot.*, vol. 2018, pp. 1–10, Sep. 2018.
[31] S. Wen, Y. Zhao, X. Yuan, Z. Wang, D. Zhang, and L. Manfredi, "Path planning for active SLAM based on deep reinforcement learning under unknown environments," *Intell. Service Robot.*, vol. 13, no. 2, pp. 263–272, Apr. 2020.
[32] N. Ab Azar, A. Shahmansoorian, and M. Davoudi, "Uncertainty-aware path planning using reinforcement learning and deep learning methods," *Comput. Knowl. Eng.*, vol. 3, no. 1, pp. 25–37, 2020.
[33] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14413–14423, Dec. 2020.
[34] L. Nardi and C. Stachniss, "Uncertainty-aware path planning for navigation on road networks using augmented MDPs," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 5780–5786.
[35] T. Peynot, S.-T. Lui, R. McAllister, R. Fitch, and S. Sukkarieh, "Learned stochastic mobility prediction for planning with control uncertainty on unstructured terrain," *J. Field Robot.*, vol. 31, no. 6, pp. 969–995, Nov. 2014.
[36] R. McAllister, T. Peynot, R. Fitch, and S. Sukkarieh, "Motion planning and stochastic control with experimental validation on a planetary rover," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 4716–4723.
[37] S. Feyzabadi and S. Carpin, "Risk-aware path planning using hirerachical constrained Markov decision processes," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2014, pp. 297–303.
[38] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Mar. 1985, pp. 500–505.
[39] U. Orozco-Rosas, K. Picos, and O. Montiel, "Hybrid path planning algorithm based on membrane pseudo-bacterial potential field for autonomous mobile robots," *IEEE Access*, vol. 7, pp. 156787–156803, 2019.
[40] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, "Mobile robot path planning using membrane evolutionary artificial potential field," *Appl. Soft Comput.*, vol. 77, pp. 236–251, Apr. 2019.
[41] U. Orozco-Rosas, K. Picos, J. J. Pantrigo, A. S. Montemayor, and A. Cuesta-Infante, "Mobile robot path planning using a QAPF learning algorithm for known and unknown environments," *IEEE Access*, vol. 10, pp. 84648–84663, 2022.
[42] S. M. H. Rostami, A. K. Sangaiah, J. Wang, and X. Liu, "Obstacle avoidance of mobile robots using modified artificial potential field algorithm," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, pp. 1–19, Dec. 2019.
[43] L. Zhou, L. Yang, and H. Tang, "Research on path planning algorithm and its application based on terrain slope for slipping prediction in complex terrain environment," in *Proc. Int. Conf. Secur., Pattern Anal., Cybern. (SPAC)*, Dec. 2017, pp. 224–227.
[44] C. Saranya, M. Unnikrishnan, S. A. Ali, D. Sheela, and V. Lalithambika, "Terrain based d* algorithm for path planning," *IFAC-PapersOnLine*, vol. 49, no. 1, pp. 178–182, 2016.
[45] X. Wang, B. Hu, and M. Zhou, "OGBPS: Orientation and gradient based path smoothing algorithm for various robot path planners," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2019, pp. 1453–1458.
[46] B. Sebastian and P. Ben-Tzvi, "Physics based path planning for autonomous tracked vehicle in challenging terrain," *J. Intell. Robotic Syst.*, vol. 95, no. 2, pp. 511–526, Aug. 2019.
[47] L. Drnach and Y. Zhao, "Robust trajectory optimization over uncertain terrain with stochastic complementarity," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1168–1175, Apr. 2021.

KOSUKE SAKAMOTO (Member, IEEE) received the B.S. and M.S. degrees in mechanical engineering from Waseda University, Tokyo, Japan, in 2016 and 2018, respectively, and the Ph.D. degree in electrical engineering from The University of Tokyo, Tokyo, in 2021. Since 2021, he has been an Assistant Professor with the Department Electrical, Electronic and Communication Engineering, Faculty of Science and Engineering, Chuo University, Tokyo. His research interests include motion control, path planning, terrain interaction, and mechanical design optimization in robotics.

YASUHARU KUNII (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Chuo University, Tokyo, Japan, in 1992 and 1994, respectively, and the Dr. (Eng.) degree in electrical engineering from The University of Tokyo, Japan, in 1997. He was a Research Fellow DC1 of JSPS with the Institute of Industrial Science, The University of Tokyo, from 1994 to 1997, he has been joining the Science and Engineering Department, Chuo University, as a Lecturer, since 1997, was an Associate Professor, from 1999 to 2014, and a Professor, in 2014. From 2010 to 2011, he was a Visiting Professor with the Technical University of Munich, Bayern, Germany. His research interests include field and space robotics, especially in tele-control and tele-intelligence.