Intelligent Service Robotics (2023) 16:19–32
https://doi.org/10.1007/s11370-022-00450-6

ORIGINAL RESEARCH PAPER

Optimal path planning using a continuous anisotropic model for navigation on irregular terrains

J. Ricardo Sánchez-Ibáñez[1] · Carlos J. Pérez-Del-Pulgar[1] · Javier Serón[1] · Alfonso García-Cerezo[1]

Received: 16 May 2022 / Accepted: 15 November 2022 / Published online: 5 December 2022

Abstract
Mobile robots usually need to minimize energy when they are traversing uneven terrains. To reach a location of interest, one strategy consists of making the robot follow the path that demands the least possible amount of energy. Yet, its calculation is not trivial with irregular surfaces. Gravity makes the energy consumption of a robot change according to its heading. Such a variation is subject to the terramechanic characteristics of the surface. This paper introduces a cost function that addresses this variation when traversing slopes. This function presents direction-dependency (anisotropic) and returns the cost for all directions (continuous).. Moreover, it is compatible with the Ordered Upwind Method, which allows finding globally optimal paths in a deterministic way. Besides, the segments of these paths are not restricted to the shape of a grid. Finally, this paper also introduces the description and discussion of a simulation experiment. It served to analyse what kinds of terrain motivate the use of anisotropy. The Ordered Upwind Method was executed on a virtual crater with different terrain parameter configurations, using both isotropic (direction-non-dependent) and anisotropic cost functions. The results evince how in certain situations the use of an anisotropic cost function instead of an isotropic one produces a path that reduces the accumulated cost by up to 20%.

Keywords Mobile Robots · Path planning · Anisotropic · Rough surfaces · Guidance

1 Introduction

Mobile robots demand more and more autonomy. They are necessary for increasingly complex tasks with limited or even nonexistent human supervision. Examples of these tasks in unstructured irregular terrains include searching for victims in emergency response scenarios [1], harvesting and/or fumigating in precision agriculture [2] or even planetary exploration [3] among others. A commonly used strategy to make mobile robots autonomously navigate is the use of path planning algorithms. These algorithms serve to calculate a path that connects the robot location to any other reachable destination [4]. Moreover, minimizing energy consumption as much as possible is desirable to make the robot perform a larger number of tasks. Thus, the path provided to the robot should be the one requiring the least amount of energy. In other words, the path planning algorithm should be optimal. Yet, this is not trivial as the idea of optimality differs between the different existing path planning approaches. We mention here three categories that are used for outdoors navigation in static environments: Sampling-based methods, Graph Search methods and Partial Derivative Equation (PDE) Solving methods [4]. These methods either create a graph that models the environment or use a pre-existing one. This graph is processed to retrieve the path from it.

Sampling-based methods model and process the graph at the same time. They iteratively create samples over a region until the initial and the goal locations are connected.

Carlos J. Pérez-Del-Pulgar, Javier Serón and Alfonso García-Cerezo have contributed equally to this work.

The original the Open Access funding note has been updated.

✉ J. Ricardo Sánchez-Ibáñez
ricardosan@uma.es

Carlos J. Pérez-Del-Pulgar
carlosperez@uma.es

Javier Serón
jseron@uma.es

Alfonso García-Cerezo
ajgarcia@uma.es

[1] Space Robotics Laboratory, Department of Systems Engineering and Automation, Universidad de Málaga, C/Ortiz Ramos s/n, 29071 Malaga, Andalusia, Spain

This is the case for the Rapidly-exploring Random Tree (RRT), which starts from one location and creates samples (branches) resembling the growth of a tree [5]. It is asymptotically optimal: after connecting with the goal location, more samples can be created before retrieving the path. This serves to incrementally find better solutions as time runs. Nevertheless, this entails a high demand for memory resources, since usually large numbers of iterations and samples are needed to come relatively close to the global optimal solution [6].

Other approaches process a graph with pre-existing connected samples, named nodes. Graph Search methods calculate a path that is made up of these nodes. The most basic of them is Dijkstra [7]. Heuristic methods like A* evolved from Dijkstra to speed up computation [8]. Their main drawback is that the shape of the path is restricted to connecting neighbouring graph nodes. Any-angle algorithms reduce this restriction by introducing the possibility of making the path be made up of non-neighbouring graph nodes. One of them is Field-D*, an algorithm implemented on the NASA rovers for the Mars Exploration Rovers (MER) and Mars Science Laboratory (MSL) missions [9,10]. However, the final solution is not globally optimal [11]. Algorithms like Theta* are still not globally optimal but provide better results in distance-minimizing path planning [12].

There is another group of path planning algorithms able to generate smooth paths in a globally optimal way: the Partial Derivative Equation (PDE) solving methods. They use also a pre-existing graph, but, unlike Graph Search methods, they do not compute any parent-child relationships between nodes. This removes the necessity to restrict the path to pass through intermediate nodes or edges. Instead, PDE Solving algorithms assign a characteristic direction to each node. This direction corresponds to the optimal heading of any path that crosses the node in question. This direction comes from solving the PDE in the respective node [13]. In this way, the path comes via interpolation with these characteristic directions. The equation used models an optimization formula, and its form determines which solver methods are suitable. For example, the Fast Marching Method (FMM) models the propagation of a wave over the graph [14]. The rate of propagation varies with cost values defined over each node, which can be non-uniform [15]. Nevertheless, as will be seen later, some forms of cost require the consideration of the vehicle heading. This means the cost may change with direction, i.e. it may be anisotropic. FMM produces sub-optimal results when this cost is anisotropic [13]. Another PDE Solving method, the Ordered Upwind Method (OUM), addresses this anisotropy in exchange for increasing the computational load [16]. It can hence find the optimal path considering the variations of cost according to the heading of the robot.

A path is optimal given the locomotion capabilities and the terrain. Therefore, the planner must address the terrain-vehicle interaction. The traverse of uneven surfaces is inherently anisotropic: the direction of the robot affects its energy consumption. For instance, a robot does not invest the same energy ascending and descending the same slope. Several works in the past proposed the use of Graph Search planners along with anisotropic power consumption models based on friction and gravity [17–21]. It is worth mentioning that while not ensuring finding the globally optimal path, these Graph Search approaches allow the use of heavy discontinuities such as forbidding ascending certain inclinations or substituting them by discontinuous zig-zag manoeuvres [20]. Other work uses nonlinear programming techniques to optimize energy while considering dynamic constraints, connecting grid nodes with feasible non-holonomic trajectories [22]. It still relies on Graph Search based methods for global planning, but this has also proven useful for local motion planning, and trajectory tracking. There is recent research oriented toward finding paths for the optimal ascension of slopes using Sampling-based methods like RRT* [23], where the difference in the slip between going straightly or diagonally through them is considered [24]. Another approach proposes the use of Sampling-based methods like SBMPO to find energy-minimizing paths considering any rise in elevation [25].

This paper presents a novel continuous and differentiable anisotropic cost model compatible with the OUM. This cost model serves to make the planner find energy-minimizing paths by addressing the energy consumption of a robot on inclined terrains. The structure of this paper comes in the following form. Section 2 presents the optimization problem that formulates the anisotropic path planner. It also presents its application on irregular terrains, focusing on the algorithm functioning and the cost function requirements. Later on, Sect. 3 fully describes the proposed anisotropic cost function. It starts with the mathematical background and later explains how this function addresses gravity, friction and slip. Next, Sect. 4 presents the results of a simulation experiment, which serves to validate its potential. Last, Sect. 5 provides our conclusions regarding the usage of an anisotropic cost function along with the OUM for navigating through scenarios containing slopes. Moreover, this section provides as well an overview of possible related future work.

2 Optimal anisotropic path planning

An anisotropic PDE serves as the cornerstone to formulate and solve the optimal path planning problem. Its solution leads to finding the optimal path on an irregular surface for a mobile robot. The region $\Omega \subset \mathbb{R}^2$ encloses the region of interest for the problem, depicted as a rectangle in Fig. 1. The target of the path planner is to find the optimal path connecting the goal $\tilde{x}_g \in \Omega$ and the origin $\tilde{x}_o \in \Omega$. This path is a continuous curve that can be parametrized by the path

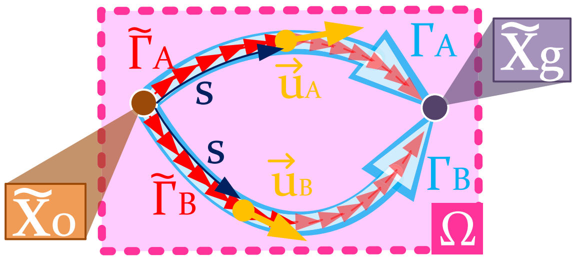


Fig. 1 Different paths $\Gamma = \Gamma(\tilde{x}_o, \tilde{x}_g, \cdot)$ connecting $\tilde{x}_o$ and $\tilde{x}_g$ can be defined upon different tangent directions $\vec{u}(\Gamma(\tilde{x}_o, \tilde{x}_g, s))$, where s is the path length. The role of the anisotropic path planner is to find the optimal path among them

length s. In this way, the function $\Gamma(\tilde{x}_o, \tilde{x}_g, s)$ returns the position at such a curve given a value of s, while $\Gamma(\tilde{x}_o, \tilde{x}_g, \cdot)$ refers to the whole path. This path is the optimal one between $\tilde{x}_o$ and $\tilde{x}_g$ by complying with Eq. (1) and has s_g as its total length. The function $Q(\Gamma(\tilde{x}_o, \tilde{x}_g, s), \vec{u}(\Gamma(\tilde{x}_o, \tilde{x}_g, s)))$ is the cost function. This function returns the value of cost given a location (in this case, $\Gamma(\tilde{x}_o, \tilde{x}_g, s)$) and a direction (in this case, $\vec{u}(\Gamma(\tilde{x}_o, \tilde{x}_g, s))$, which is explained later). $T(\tilde{x}_o, \tilde{x}_g)$ is the amount of cost accumulated along the optimal path. This amount is referred to as the total cost. It is the objective function that the planner has to minimize when searching for the optimal path.

$$T(\tilde{x}_o, \tilde{x}_g) = \min_{\Gamma(\tilde{x}_o, \tilde{x}_g, \cdot) \in \Omega} \left\{ \int_0^{s_g} Q(\Gamma(\tilde{x}_o, \tilde{x}_g, s), \vec{u}(\Gamma(\tilde{x}_o, \tilde{x}_g, s)))\mathrm{d}s \right\} \tag{1}$$

The formulation of the total cost is based on the Dynamic Programming Principle (DPP). Under this principle, the total cost between two points such as the origin $\tilde{x}_o$ and the goal $\tilde{x}_g$ is the minimum possible. This makes path planning methods working with DPP globally optimal. The DPP is expressed in Eq. (2) and explained as follows. For any point $\tilde{x}_{ij} \in \Omega$, the total cost from $\tilde{x}_o$ to that point, $T(\tilde{x}_o, \tilde{x}_{ij})$, and from that point to $\tilde{x}_g$, $T(\tilde{x}_{ij}, \tilde{x}_g)$, is equal to the total cost between $\tilde{x}_o$ and $\tilde{x}_g$, only if this point $\tilde{x}_{ij}$ is placed at the optimal path connecting them, $\Gamma(\tilde{x}_o, \tilde{x}_g, \cdot)$.

$$T(\tilde{x}_o, x_{ij}) + T(x_{ij}, \tilde{x}_g) = T(\tilde{x}_o, \tilde{x}_g), \quad \forall x_{ij} \in \Gamma(\tilde{x}_o, \tilde{x}_g, s) \in \Omega \tag{2}$$

With regards the direction $\vec{u}(\tilde{x}_{ij})$, it is the direction tangent to the optimal path that passes through $\tilde{x}_{ij}$. This is expressed in (3). This direction is also called the characteristic direction of a location $\tilde{x}_{ij} \in \Omega$. In this way, given how the anisotropic cost function is defined, all the optimal paths passing through the location in question will have this same characteristic direction on it.

$$\vec{u}(\tilde{x}_{ij}) = \left.\frac{\mathrm{d}\Gamma(\tilde{x}_o, \tilde{x}_g, s)}{\mathrm{d}s}\right|_{\tilde{x}_{ij}} \tag{3}$$

To solve (1), it is reformulated as the Hamilton-Jacobi-Bellman (HJB) equation, following the reasoning of previous works [13,16,26]. This equation, shown in (4), establishes the correspondence between the anisotropic cost $Q(\tilde{x}_{ij}, \vec{\psi})$ and the spatial gradient of the total cost ∇T. Here, the anisotropic cost function is defined not only according to the location of any grid $\tilde{x}_{ij}$ but also to the heading function $\vec{\psi}$. The latter function is the direction of the robot in the XY-plane, i.e. its heading, returned as a 2D unit vector. The direction $\vec{\psi}$ that complies with (4) in a grid node $\tilde{x}_{ij}$ corresponds hence to the characteristic direction $\vec{u}(\tilde{x}_{ij})$ passing through it. Moreover, following the DPP in (2) and the definition of the characteristic direction in (3), the total cost in (1) is expressed from $\tilde{x}_o$ to $\tilde{x}_{ij}$ as well as from $\tilde{x}_{ij}$ to $\tilde{x}_g$.

$$\begin{aligned} &\min_{\vec{\psi}}\{\nabla T \cdot \vec{\psi} + Q(\tilde{x}_{ij}, \vec{\psi})\} = 0, \ \forall \tilde{x}_{ij} \in \tilde{\Omega} \\ &\nabla T = \nabla T(\tilde{x}_o, \tilde{x}_{ij}) = \nabla T(\tilde{x}_{ij}, \tilde{x}_g) \end{aligned} \tag{4}$$

The Ordered Upwind Method (OUM) uses (4) to find the optimal path on a regular grid. According to the work of Sethian and Vladimirsky [13], this algorithm has a computational complexity that depends on the anisotropy Υ of the cost function: $O(\Upsilon n_{\text{nodes}} \log(n_{\text{nodes}}))$. Here, n_{nodes} is the number of nodes of the grid. This anisotropy Υ comes as the ratio between the highest and the lowest values of cost depending on the heading. However, for the case presented in this paper, the anisotropy varies from node to node, i.e. the anisotropy here is $\Upsilon(\tilde{x}_{ij})$. Therefore, the computational complexity is expected to be variable as well, bounded by the maximum existing anisotropy in the grid $\tilde{\Omega}$. According to Equation (5), this anisotropy $\Upsilon(\tilde{x}_{ij})$ of a node $\tilde{x}_{ij}$ comes as the ratio between the highest possible cost at its location and the lowest according to the heading direction $\vec{\psi}$.

$$\Upsilon(\tilde{x}_{ij}) = \frac{\max_{\vec{\psi}}\{Q(\tilde{x}_{ij}, \vec{\psi})\}}{\min_{\vec{\psi}}\{Q(\tilde{x}_{ij}, \vec{\psi})\}} \tag{5}$$

There is an improved version of the OUM, named the bi-directional OUM. It generates the same solution in a faster way than OUM by reducing the number of visited nodes [26]. Bi-OUM calculates both $T(\tilde{x}_o, \tilde{x}_{ij})$ and $T(\tilde{x}_{ij}, \tilde{x}_g)$ using two propagating waves in two parallel loops: *Goal wave*, which starts from $\tilde{x}_g$, and *Origin wave*, which starts from $\tilde{x}_o$. This exploits the DPP presented in Eq. (2). Figure 2 shows this functioning with a regular grid. Each parallel loop is coloured with a different colour, either purple (*Goal wave*) or orange (*Origin wave*). As can be checked in this figure, is not neces-

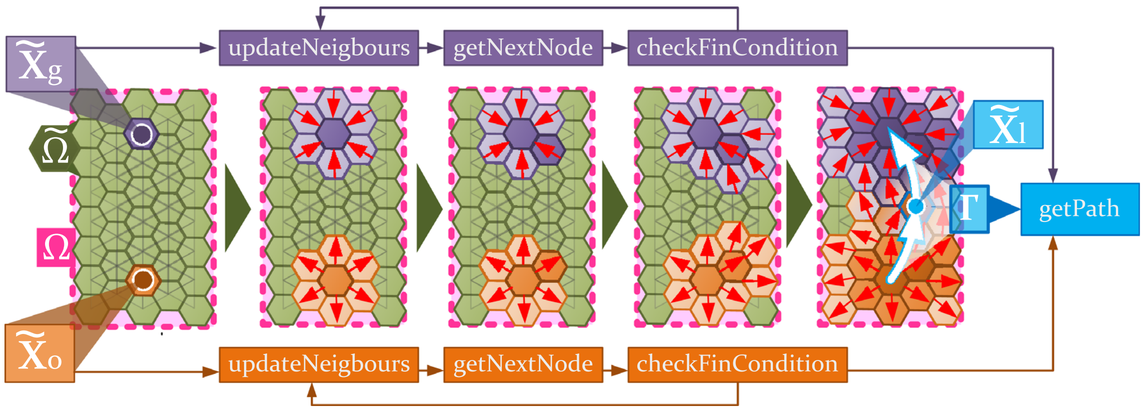


Fig. 2 Schematic of the functioning of bi-OUM. Two expanding waves start from the goal and the origin nodes ($\tilde{x}_g$ and $\tilde{x}_o$) respectively. The loop controlling each wave is coloured in purple and orange. The red arrows indicate the characteristic direction $\vec{u}(\tilde{x}_{ij})$ that passes through each node $\tilde{x}_{ij}$

sary that the two waves propagate until reaching the starting position of the other wave. When both waves reach an intermediate linking node $\tilde{x}_l$ the bi-OUM process stops. This is because, as highlighted in Eq. (2), this node complies with the DPP and there is only a single path passing through it: the optimal path. This path is calculated by following the characteristic direction that is also calculated in both waves, one from $\tilde{x}_l$ to $\tilde{x}_o$ and another from $\tilde{x}_l$ to $\tilde{x}_g$. In this way, bi-OUM achieves the same optimal solution as the OUM but invests less time than the latter. This is because the bi-directional version visits fewer nodes than the original [26].

To use the HJB equation in (4) for calculating the total cost and the characteristic direction of any node, the bi-OUM employs a semi-Lagrangian discretization. It was already used by Sethian and Vladimirsky [13] when they introduced the OUM. This discretization produces Eqs. (6) and (7) as a result. They serve to calculate, in an iterative way, the value of total cost. The *Origin wave* uses Eq. (6) while the *Goal wave* uses Eq. (7). In both expressions the update process depends on two other nodes, $\tilde{x}_{i'j'}$ and $\tilde{x}_{i''j''}$, whose corresponding values of total cost are already calculated. The main difference between (6) and (7) is that the total cost from the goal node has to consider that the anisotropic cost function multiplies the input value of characteristic direction by -1. This is because the propagating wave advances in the contrary direction to the heading of the robot. The iterations end when the value of $\epsilon \in [0, 1]$ is found after converging. Equation (8), also coming from the semi-lagrangian discretization, expresses how the characteristic direction is calculated, based on the value of ϵ found.

$$T(\tilde{x}_o, \tilde{x}_{ij}) = \min_{\epsilon\in[0,1]} \big\{Q(\tilde{x}_{ij}, \vec{u}(\tilde{x}_{ij}))|\epsilon\tilde{x}_{i'j'} + (1-\epsilon)\tilde{x}_{i''j''} - \tilde{x}_{ij}| + \epsilon T(\tilde{x}_o, \tilde{x}_{i'j'}) + (1-\epsilon)T(\tilde{x}_o, \tilde{x}_{i''j''})\big\} \tag{6}$$

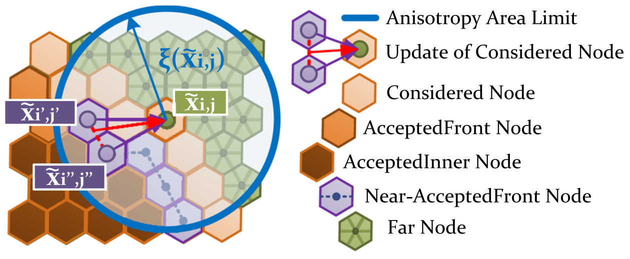


Fig. 3 Update process of a Considered node $\tilde{x}_{ij}$. Its values of total cost and characteristic direction are computed taking into account Accepted-Front nodes within the distance $\xi(\tilde{x}_{ij})$ expressed in Eq. (9), also called Near-AcceptedFront nodes

$$T(\tilde{x}_{ij}, \tilde{x}_g) = \min_{\epsilon\in[0,1]} \big\{Q(\tilde{x}_{ij}, -\vec{u}(\tilde{x}_{ij}))|\epsilon\tilde{x}_{i'j'} + (1-\epsilon)\tilde{x}_{i''j''} - \tilde{x}_{ij}| + \epsilon T(\tilde{x}_{i'j'}, \tilde{x}_g) + (1-\epsilon)T(\tilde{x}_{i''j''}, \tilde{x}_g)\big\} \tag{7}$$

$$\vec{u}(\tilde{x}_{ij}) = \frac{\epsilon\tilde{x}_{i'j'} + (1-\epsilon)\tilde{x}_{i''j''} - \tilde{x}_{ij}}{||\,\epsilon\tilde{x}_{i'j'} + (1-\epsilon)\tilde{x}_{i''j''} - \tilde{x}_{ij}\,||} \tag{8}$$

Each node has two state functions: S^o_{ij} and S^g_{ij}. Each of these functions indicates the state of the node $\tilde{x}_{ij}$ according to the *Origin wave* or the *Goal wave* respectively. S^o_{ij} and S^g_{ij} present each one state out of the four presented in Table 1, which will be further explained later. The nodes $\tilde{x}_{i'j'}$ and $\tilde{x}_{i''j''}$ that affect the explained update processes of the total cost and the characteristic direction must present an AcceptedFront state. Besides, they must be located close enough to $\tilde{x}_{ij}$, with a proximity distance lower than $\xi(\tilde{x}_{ij})$. This distance threshold is defined in Eq. (9). It comes in function of the anisotropy $\Upsilon(\tilde{x}_{ij})$ that is present at such node, already expressed in (5). $\xi(\tilde{x}_{ij})$ is also proportional to the grid resolution Λ. Figure 3 shows how this distance works.

$$\xi(\tilde{x}_{ij}) = \Lambda\Upsilon(\tilde{x}_{ij}) \tag{9}$$

The process followed by bi-OUM to visit the grid nodes, represented in Fig. 2, is also presented as pseudo-code in Algorithm 1. First of all, both S^{o}_{ij} and S^{g}_{ij} are initialized to Far for every node. Thereafter, each loop sets the total cost of the respective starting node to zero, a null value to the characteristic direction (since initial and final desired orientations are not defined) and the Accepted state to the respective state function. Next, each process updates the neighbours of the corresponding starting node thanks to the updateNeighbours() function. Two functions are iteratively executed by the *Goal wave* and the *Origin wave*. On the one hand, updateNeighbours() performs three actions. First, it checks whether an AcceptedFront node no longer has Far or Considered neighbours and changes its state to AcceptedInner. Second, it converts those Far nodes that are neighbours to a target node $\tilde{x}_t$ into Considered. This target node is initially the starting node from which the wave expands. Third, the total cost of the Considered neighbours to the target node is updated using either Eqs. (6) or (7), as well as the characteristic direction using Eq. (8). Thereafter, the following target node is chosen using the getNextNode() function. This function takes the existing Considered node with the lowest value of total cost, $\tilde{x}_t$, and changes its state into AcceptedFront.

Algorithm 1 The bi-OUM.

```
Input: $\tilde{x}_0, \tilde{x}_g, \tilde{\Omega}$
Output: $\tilde{\Gamma}$
1: $T(\tilde{x}_{ij}, \tilde{x}_g), \vec{u}(\tilde{x}_{ij}) \leftarrow \infty, NaN$          $\forall \tilde{x}_{ij} \in \tilde{\Omega}$
2: $S^{\mathrm{g}}_{ij} \leftarrow Far$          $\forall \tilde{x}_{ij} \in \tilde{\Omega}$
3: $T(\tilde{x}_g, \tilde{x}_g), \vec{u}(\tilde{x}_g) \leftarrow 0, NaN$
4: $S^{\mathrm{g}}_{\mathrm{g}} \leftarrow AcceptedFront$
5: $T(\tilde{x}_o, \tilde{x}_{ij}), \vec{u}(\tilde{x}_{ij}) \leftarrow \infty, NaN$          $\forall \tilde{x}_{ij} \in \tilde{\Omega}$
6: $S^{\mathrm{o}}_{ij} \leftarrow Far$          $\forall \tilde{x}_{ij} \in \tilde{\Omega}$
7: $T(\tilde{x}_o, \tilde{x}_o), \vec{u}(\tilde{x}_o) \leftarrow 0, NaN$
8: $S^{\mathrm{o}}_{\mathrm{o}} \leftarrow AcceptedFront$
9: $\tilde{x}_{tg} \leftarrow \tilde{x}_g$
10: $\tilde{x}_{to} \leftarrow \tilde{x}_o$
11: while $\neg checkFinCondition()$ do          ▷ Both waves
12:     $updateNeighbours(\tilde{x}_{tg})$          ▷ Goal wave
13:     $updateNeighbours(\tilde{x}_{t0})$          ▷ Origin wave
14:     $\tilde{x}_{tg} \leftarrow getNextNode(T(\cdot, \tilde{x}_g))$          ▷ Goal wave
15:     $\tilde{x}_{t0} \leftarrow getNextNode(T(\tilde{x}_o, \cdot))$          ▷ Origin wave
16: end while
17: return $getPath(S_g, S_0)$
```

Finally, the checkFinCondition() function evaluates $\tilde{x}_{\mathrm{t}}$ to check if its state is AcceptedInner or AcceptedFront for both loops. If so, the waves stop and $\tilde{x}_{\mathrm{t}}$ becomes $\tilde{x}_{\mathrm{l}}$, indicated in Fig. 2 as a blue dot. Later on, the getPath() function returns the optimal path from two portions: one from $\tilde{x}_{\mathrm{l}}$ to $\tilde{x}_{\mathrm{o}}$ and another from $\tilde{x}_{\mathrm{l}}$ to $\tilde{x}_{\mathrm{o}}$. Each waypoint is calculated as indicated in (10), where the step distance is d_{step}, and the characteristic direction for each waypoint is interpolated from the values calculated on the nodes. The final path $\tilde{\Gamma} = \left\{\tilde{\Gamma}_{\mathrm{o}}, \tilde{\Gamma}_1, \tilde{\Gamma}_2 ... \tilde{\Gamma}_{\mathrm{g}}\right\}$ is obtained by joining both portions (see Fig. 2).

$$\begin{aligned} \tilde{\Gamma}_{k-1} &= \tilde{\Gamma}_k - d_{\mathrm{step}} \vec{u}(\tilde{\Gamma}_k) \;,\; if\ Origin\ wave \\ \tilde{\Gamma}_{k+1} &= \tilde{\Gamma}_k + d_{\mathrm{step}} \vec{u}(\tilde{\Gamma}_k) \;,\; if\ Goal\ wave \end{aligned} \tag{10}$$

3 Anisotropic cost function for inclined surfaces

This section presents the steps followed to make an anisotropic cost function $Q(\tilde{x}_{ij}, \vec{\psi})$ be based on slope parameters and compatible with the (bi-)OUM.

3.1 Slope-based anisotropy

Figure 4 shows the slope model used to represent the interaction between the robot and inclined terrain. In the absence of any kinematic configuration capable to reconfigure itself [27], the pose of the robot body will change according to the shape of the terrain surface. This change will be determined by the contact points between the robot and the surface. Nevertheless, to avoid making the formulation more complex, a simplification is made: the robot-terrain interaction is modelled after a single contact point. This simplification assumes the robot body is always parallel to an imaginary inclined plane. The normal vector of this plane, named $\vec{\nu}_{ij}$ in Fig. 4, will be hence coincident with the Z-axis of the robot local reference frame. This imaginary plane is inclined at a certain angle from the horizontal XY plane (the plane perpendicular to the gravity vector). The value of this angle corresponds to the slope gradient α_{ij} and is equal to the angle between the normal vector $\vec{\nu}_{ij}$ and the global Z-axis as showcased in Fig. 4. The direction the slope faces, i.e. the direction in which the steepest descent occurs, is the aspect or $\vec{\gamma}_{ij}$. It can be obtained from projecting the normal vector of the slope, $\vec{\nu}_{ij}$, onto the 2D XY plane and normalizing it.

To make the anisotropic cost function $Q(\tilde{x}_{ij}, \vec{\psi})$ compatible with the (bi-)OUM, it is necessary to understand its requirements. First of all, this function must always return a positive and non-zero value. Complying with this condition avoids producing undesirable local minimum points that compromise the optimality of the resulting path, making it sub-optimal [13]. Next, the solution that bi-OUM produces is viscous, meaning it not only exists but is also unique and stable. This is thanks to the fact that the computed solution omits any discontinuity present in the real solution of the total cost [16]. However, the computed solution is guaranteed to be unique if, and only if, it is ensured that the inverse of the cost function, $1/Q(\tilde{x}_{ij}, \vec{\psi})$, is fully differentiable and convex [13,16]. For this reason, as shown in Fig. 5 the inverse of the real cost of the robot, $1/\tilde{Q}(\tilde{x}_{ij}, \vec{\psi})$, is approximated with a

Table 1 Possible states of grid nodes in the calculation of the values of total cost and characteristic direction

State	Visited	Total cost and characteristic direction	All neighbours are accepted
Far	No	Irrelevant	No
Considered	Yes	Tentative	No
AcceptedFront	Yes	Definitive	No
AcceptedInner	Yes	Definitive	Yes

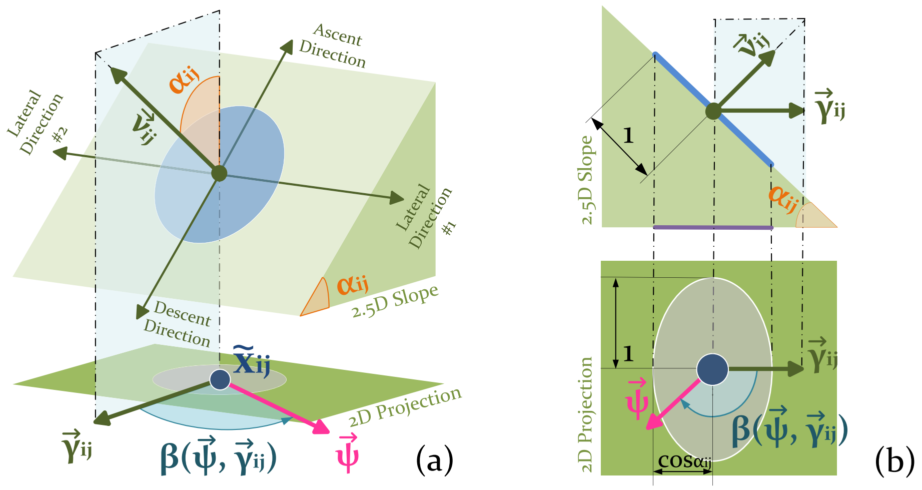


Fig. 4 Graphical representation of the slope model used to build up the anisotropic cost function. This model encompasses multiple variables such as the slope gradient α, the aspect $\vec{\gamma}_{ij}$, the normal vector $\vec{\nu}_{ij}$, the heading direction $\vec{\psi}_{ij}$ and the relative angle $\beta(\vec{\psi}, \vec{\gamma}_{ij})$. All of them are marked in this conceptual depiction. **a** 3D Perspective view of slope model. **b** Lateral and top views of the slope model

closed conic curve. This curve is $1/Q(\tilde{x}_{ij}, \vec{\psi})$, which is formulated as a displaced ellipse due to its simplicity. Equation (11) expresses the polar form of this ellipse, whose radius is $1/Q(\tilde{x}_{ij}, \vec{\psi})$. This form uses $\beta(\vec{\psi}, \vec{\gamma}_{ij})$, which represents the angle between the robot heading $\vec{\psi}$ and the aspect $\vec{\gamma}_{ij}$. In this way, when $\vec{\psi}$ coincides with $\vec{\gamma}_{ij}$ then $\beta(\vec{\psi}, \vec{\gamma}_{ij})$ returns a value of zero. Figure 5 depicts how the radius of this ellipse, which corresponds to the inverse of the cost $1/Q(\tilde{x}_{ij}, \vec{\psi})$, varies with $\beta(\vec{\psi}, \vec{\gamma}_{ij})$.

$$0 = \frac{[\cos_\beta \ \sin_\beta]^T \begin{bmatrix} -C^{\mathrm{a}}_{ij}C^{\mathrm{d}}_{ij} & 0 \\ 0 & -{C^{l}_{ij}}^2 \end{bmatrix} \begin{bmatrix} \cos_\beta \\ \sin_\beta \end{bmatrix}}{Q(\tilde{x}_{ij}, \vec{\psi})^2} + \\ + \frac{\begin{bmatrix} C^{\mathrm{a}}_{ij} - C^{\mathrm{d}}_{ij} & 0 \end{bmatrix} \begin{bmatrix} \cos_\beta \\ \sin_\beta \end{bmatrix}}{Q(\tilde{x}_{ij}, \vec{\psi})} + 1 \,, \quad \beta = \beta(\vec{\psi}, \vec{\gamma}_{ij}) \tag{11}$$

The shape of the ellipse varies with the slope gradient α_{ij} through three functions: the ascent cost C^{a}_{ij}, the lateral cost C^{l}_{ij} and the descent cost C^{d}_{ij}. Each of them correspond to what the real cost $\tilde{Q}(\tilde{x}_{ij}, \vec{\psi})$ returns at certain values of $\beta(\vec{\psi}, \vec{\gamma}_{ij})$: $\pm\pi$ (12), $\pm\pi/2$ (13) and 0 (14).

$$C^{\mathrm{a}}_{ij} = \tilde{Q}(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = \pm\pi) \tag{12}$$

$$C^{l}_{ij} = \tilde{Q}\left(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = \pm\frac{\pi}{2}\right) \tag{13}$$

$$C^{\mathrm{d}}_{ij} = \tilde{Q}(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = 0) \tag{14}$$

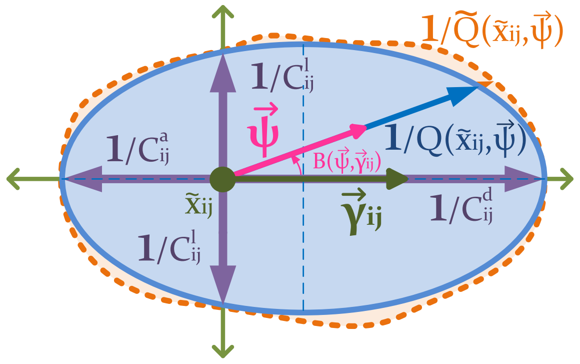

$1/\tilde{Q}(\tilde{x}_{ij},\vec{\psi})$
$1/C^{l}_{ij}$
$\vec{\psi}$
$1/C^{a}_{ij}$
$B(\vec{\psi},\vec{\gamma}_{ij})$
$1/Q(\tilde{x}_{ij},\vec{\psi})$
$\tilde{x}_{ij}$
$\vec{\gamma}_{ij}$
$1/C^{d}_{ij}$
$1/C^{l}_{ij}$

Fig. 5 Elliptical inverse of the anisotropic cost function $Q(\tilde{x}_{ij}, \vec{\psi})$. It serves to approximate the inverse of the real cost of the robot, $\tilde{Q}(\tilde{x}_{ij}, \vec{\psi})$, while complying with the requirements of the OUM (convex, continuous, closed). The robot has a heading direction $\vec{\psi}$ that has an angle $\beta(\vec{\psi}, \vec{\gamma}_{ij})$ with respect to the aspect $\vec{\gamma}_{ij}$

The terms $\cos_\beta$ and $\sin_\beta$ can be substituted by the expressions in (15), given the fact that the ellipse in Fig. 5 is symmetrical in the axis where $\vec{\gamma}_{ij}$ is located. With these expressions in mind, the terms in (11) can be rearranged to leave this equation in the explicit form shown in (16).

$$|\cos_\beta| = \vec{\psi} \cdot \vec{\gamma}_{ij} \quad , \quad |\sin_\beta| = ||\vec{\psi} \times \vec{\gamma}_{ij}|| \tag{15}$$

$$Q(\tilde{x}_{ij}, \vec{\psi}) =$$

$$\sqrt{\left(\frac{C^{\mathrm{a}}_{ij}+C^{\mathrm{d}}_{ij}}{2}\right)^2\left(\vec{\psi}\cdot\vec{\gamma}_{ij}\right)^2+\left(C^{l}_{ij}\,||\vec{\psi}\times\vec{\gamma}_{ij}||\right)^2}$$
$$-\frac{C^{\mathrm{a}}_{ij}-C^{\mathrm{d}}_{ij}}{2}\,\vec{\psi}\cdot\vec{\gamma}_{ij} \qquad (16)$$

3.2 Energy minimizing cost function

The next step is to re-define $Q(\tilde{x}_{ij},\vec{\psi})$ by means of the robot energy consumption according to its dynamics. Equation (17) serves to model this consumption when climbing any slope. It is based on a model created in previous work to represent the current consumption of a robot with n wheels and wheel radius r [28]. Here we assume that the same voltage is supplied to all motors, so the current consumption is proportional to the power. The motor torque constant is referred to as K_m, which serves to translate the torque into the current.

$$I(\rho_{ij},\sigma_{ij},\alpha_{ij})=nK_m r\frac{mg(\rho_{ij}\cos_{\alpha_{ij}}-\sin_{\alpha_{ij}})}{\cos_{\alpha_{ij}}(1-\sigma_{ij})} \qquad (17)$$

The terms that appear in the fraction are explained as follows. The numerator is the Drawbar Pull Resistance Force. It represents the resulting force that drags the robot. It comes as a function of the specific resistance coefficient ρ_{ij}, the mass of the robot m and the gravity acceleration g. Rowe and Ross [17] use a similar expression to model the same. The coefficient ρ_{ij} estimates the ratio between the normal force, generated by the terrain surface, and the drawbar pull, produced by making the wheels roll.

Two terms are in the denominator. First, $\cos_{\alpha_{ij}}$ arises to account for the fact that we are solving a two-dimensional path planning problem. In other words, the 2.5D elevation map is projected onto the 2D plane and the aforementioned robot speed v takes different values in the 2D projection when climbing or descending through a slope, i.e. changing its Z-coordinate. This can be understood better by checking on Fig. 4a, where the blue circle on the slope takes the form of an ellipse in its projection onto the XY-plane. Second, the slip ratio σ_{ij} is the difference between 1 and the ratio between v_{ij} and the estimated speed that is commanded to the wheels. In other words, σ_{ij} takes a value of zero when v_{ij} and the commanded speed are the same. Its value gets close to one as the commanded speed increases.

The functions $C^{\mathrm{a}}_{ij}(\tilde{x}_{ij})$, $C^{l}_{ij}(\tilde{x}_{ij})$ and $C^{\mathrm{d}}_{ij}(\tilde{x}_{ij})$ are defined using (17) to make the anisotropic cost function $Q(\tilde{x}_{ij},\vec{\psi})$ consider the path planning criterion of electric charge minimization. They are expressed in Eqs. (18), (19) and (20), respectively. All of them consider the robot speed v_{ij}.

$$C^{\mathrm{a}}_{ij}=\frac{I(\rho_{ij},\sigma_{ij},\alpha_{ij})}{v_{ij}} \qquad (18)$$

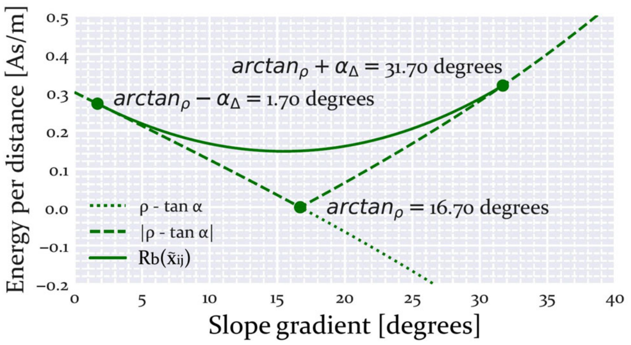


Fig. 6 Use of Bezier curve function $R_{\mathrm{b}}(\tilde{x}_{ij})$ to comply with the non-zero positive cost specification in a smooth way. $\rho_{ij}=0.3$, $\alpha_\Delta=15°$ and $\alpha^{\mathrm{zero}}_{ij}=\mathrm{arctan}_{\rho_{ij}}$

$$C^{l}_{ij}=\frac{I(\rho_{ij},\sigma_{ij},0)}{v_{ij}} \qquad (19)$$

$$C^{\mathrm{d}}_{ij}=\begin{cases} nK_m r\dfrac{mgR_{\mathrm{b}}(\tilde{x}_{ij})}{v_{ij}(1-\sigma_{ij})}, & \alpha_{ij}\in(\alpha^{\mathrm{zero}}_{ij}-\alpha_\Delta,\alpha^{\mathrm{zero}}_{ij}+\alpha_\Delta) \\ \left|\dfrac{I(\rho_{ij},\sigma_{ij},-\alpha_{ij})}{v_{ij}}\right|, & otherwise \end{cases} \qquad (20)$$

The function $C^{l}(\tilde{x}_{ij})$, expressed in (19), takes a value of zero for the α_{ij} input. This is because, whenever the robot goes in a direction perpendicular to the aspect γ_{ij}, the value of elevation does not change, i.e. the robot does not ascend or descend. On the other hand, a special treatment is given to $C^{\mathrm{d}}(\tilde{x}_{ij})$, expressed in (20). The effect of gravity pulls the robot and reduces its energy consumption when it descends. Here, it is assumed the vehicle cannot recharge itself, so the energetic cost should always be present in the form of a positive value. Besides, it may be also desirable to prevent the robot from braking when descending through slopes, so the loss of energy in the form of heat is avoided [17]. When the robot descends, having a value of $-\alpha_{ij}$ as input that acknowledges this robot pose, the current function from (17) could return zero or even negative values. This is incompatible with the OUM and bi-OUM requirements. Therefore, this situation is dealt with by using the Bezier function $R_{\mathrm{b}}(\tilde{x}_{ij})$ that is present in (20). This function acknowledges the angle of slope gradient in which the robot would start gaining energy, $\alpha^{\mathrm{zero}}_{ij}$. From this slope gradient, the robot starts braking to keep its velocity, avoiding any acceleration. $R_{\mathrm{b}}(\tilde{x}_{ij})$ preserves continuity and smoothness while making the cost $C^{\mathrm{d}}(\tilde{x}_{ij})$ return always positive values that increase with the slope gradient α_{ij}. This is showcased in Fig. 6.

Given the expression in (17), the angle $\alpha^{\mathrm{zero}}_{ij}$ would be equal to $\mathrm{arctan}_{\rho_{ij}}$. The absolute value is used to define $C^{\mathrm{d}}(\tilde{x}_{ij})$ in (20) so this function increases the cost with val-

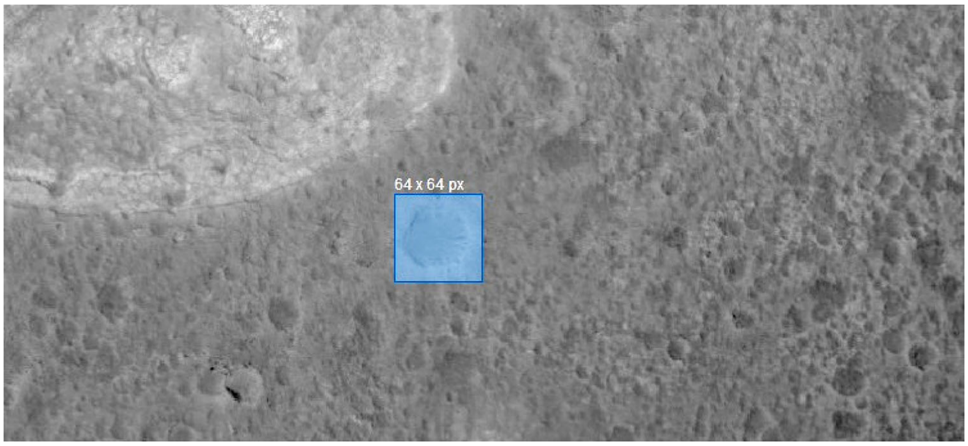


Fig. 7 Selection of region from the DEM of a crater. The elevation used is half of the original. Preparation of the map used for the simulation tests with the anisotropic planner. It is based on the shape of a real crater on the Martian surface. The resolution of the DEM is 1 m

ues of slope gradient higher than $\alpha_{ij}^{\text{zero}} + \alpha_{\Delta}$. The use of an absolute value was used by Rowe and Ross [17] to penalize paths that required the robot to brake and not accelerate. Here, α_{Δ} is a custom configurable angle margin. $R_{\text{b}}(\tilde{x}_{ij})$ marks three points using this margin: at $\alpha_{ij}^{\text{zero}} - \alpha_{\Delta}$, at $\alpha_{ij}^{\text{zero}}$ and at $\alpha_{ij}^{\text{zero}} + \alpha_{\Delta}$. A Bezier curve is the basis of $R_{\text{b}}(\tilde{x}_{ij})$ to not only preserve continuity but also smoothness while penalizing braking.

4 Experiments

This section presents an evaluation of the anisotropic cost function $Q(\tilde{x}_{ij}, \vec{\psi})$, presented in Sect. 3 for path planning purposes. In planetary exploration missions, rovers may drive not only on horizontal surfaces but also on inclined ones. For instance, the Spirit rover was commanded over weeks to climb the Husband summit on Mars and later descend it [29]. For this reason, we prepared and carried out a numerical simulation using the model of a martian unstructured environment containing a crater. This paper focuses on executing only the bi-OUM algorithm to produce a series of paths in the mentioned scenario. This is because a past comparative study already demonstrates how bi-OUM generates better solutions and even in a faster way than others compatible with anisotropic cost functions, like OUM, RRT* and Genetic Algorithms [26].

The purpose of this experiment is twofold. First, it is of interest to analyse how the terrain affects the performance of the anisotropic cost function to find energy-minimizing paths. Second, to figure out how significant is the use of an anisotropic function in contrast with an isotropic function. The latter is usable by the Fast Marching Method (FMM), another PDE Solving method introduced in Sect. 1 that employs much lower computational complexity: $O(n_{\text{nodes}} \log(n_{\text{nodes}}))$ [30]. All the code used to produce the paths of all the tests is written using the Python language and is available online.[1]

The simulation test was carried out using the Digital Elevation Model (DEM) of a crater. This DEM is based on the shape of a real crater located close to where the Spirit rover landed on Mars. The data was obtained from the High Resolution Imaging Science Experiment (HiRISE)[2] repository and was adapted to make it present slopes up to 20 degrees. The main reason to do this is that the original presents pronounced slopes that would be non-traversable, and the key point in this simulation test is to have a large variety of traversable inclined surfaces. Figure 7 shows the 80×80 m portion of elevation data that was extracted from the original DEM. Figure 8 presents the elevation data describing the shape of this crater together with some contour lines to ease the visualization. The slope gradient (maximum inclination) is shown in Fig. 9 for all the points on the map.

First of all, we analyse how the specific resistance ρ_{ij} and the slip ratio σ_{ij} influence the energy consumption estimation provided by the anisotropic cost function. It is worth mentioning that, in a similar way to isotropic cost functions, the constant parameters that multiply the whole function, acting as gains, do not affect the location of the waypoints of the resulting path. These parameters only modify proportionally the values of total cost assigned to the nodes. Since the value returned by the specific resistance ρ_{ij} can be a real number higher than 0 and lower than 1, the simulation test is performed using a discrete set of constant values of ρ_{ij}: 0.15, 0.3, 0.45, 0.6, 0.75 and 0.9. In other words, the planner is executed several times, using one out of the six different values of ρ_{ij} each time for all nodes $\tilde{x}_{ij} \in \tilde{\Omega}$.

With regards to the slip ratio σ_{ij}, it is defined after two models constructed from experimental data and introduced by Sutoh et al. [31]. These models were constructed from

[1] https://github.com/spaceuma/CAMIS_python.

[2] http://www.uahirise.org/dtm/dtm.php?ID=ESP_023957_1755 Accessed on 6th December 2021.

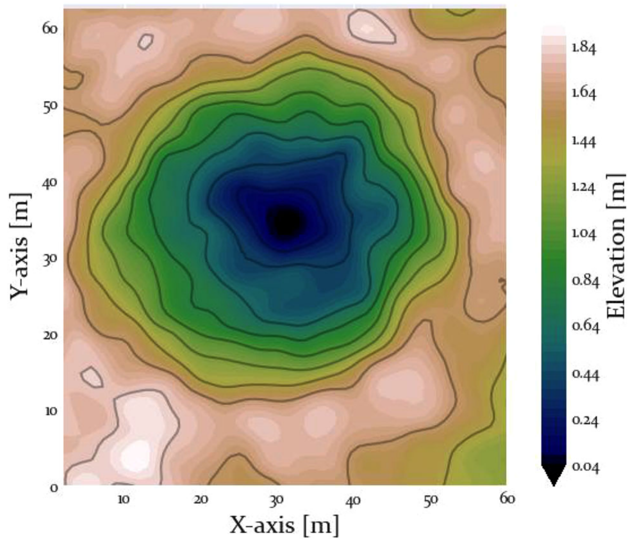


Fig. 8 Elevation map

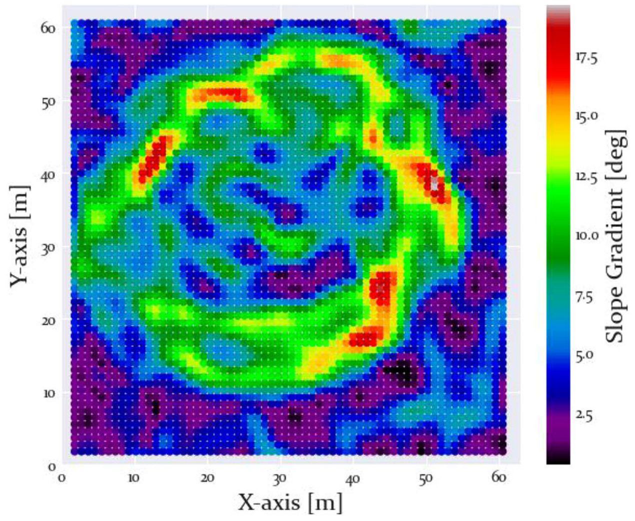


Fig. 9 Slope gradient map

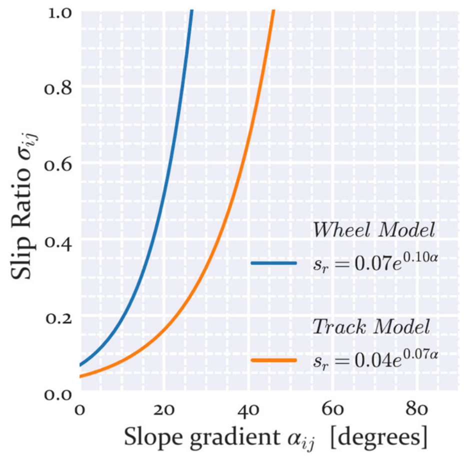


Fig. 10 Slip ratio function of the two models (*Wheel* and *Track*) found in the literature [31], used in the numerical simulation tests

two locomotion subsystems: one using wheels and the other using tracks. These two mechanisms were simulated on an inclined sandbox, making them work with lunar regolith simulant [32]. Figure 10 depicts the two functions of slip ratio σ_{ij} that depend on the slope gradient α_{ij}. Moreover, they are expressed in Equation (21). With the increase of this slope gradient, both slip ratio functions return higher values for both models, but at different rates. The slip ratio of the Wheel model increases faster than that of the Track model.

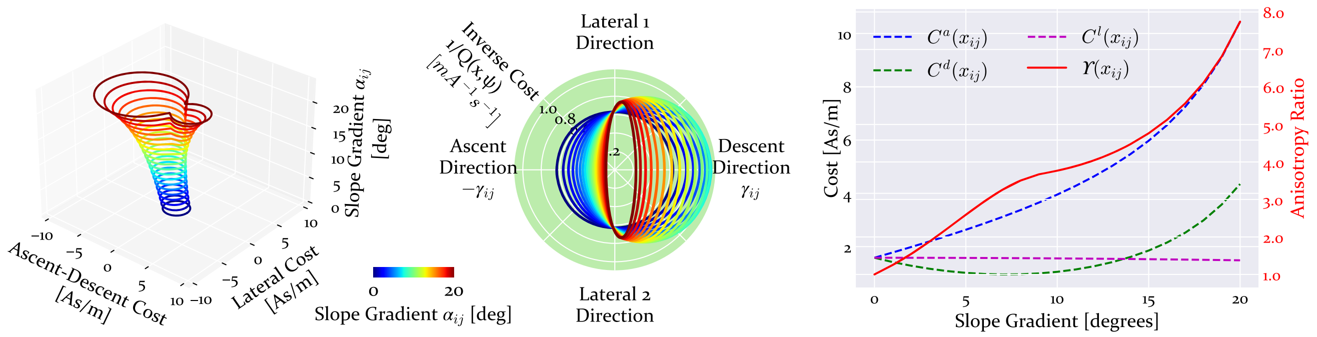
$$\sigma_{ij} = \begin{cases} 0.07e^{0.1\alpha_{ij}} & \text{if Wheel Model} \\ 0.04e^{0.07\alpha_{ij}} & \text{if Track Model} \end{cases} \tag{21}$$

In this way, there are six constant values to define the specific resistance ρ_{ij} and two functions that depend on α_{ij} to define the slip ratio σ_{ij}. Moreover, it is here created an isotropic cost function that takes the highest value of cost from the anisotropic cost function as shown in (22).

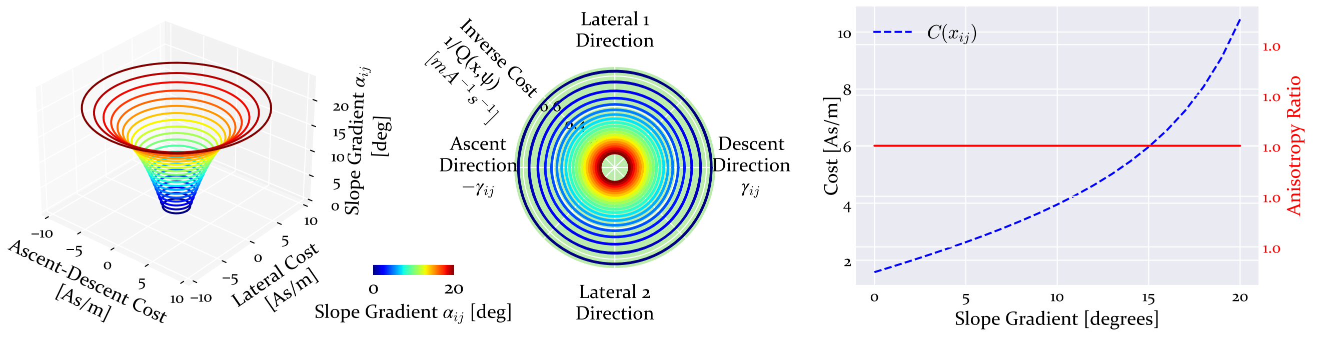
$$C(\tilde{x}_{ij}) \equiv \max_{\vec{\psi}}\{Q(\tilde{x}_{ij}, \vec{\psi})\} \tag{22}$$

For a better understanding of how the anisotropic and isotropic cost functions are defined in this simulation test, Fig. 11 is provided. It contains 4 subfigures with 3 plots each. The left plot depicts the anisotropic cost function using a 3d representation. This representation resembles a vertical cylinder: the base axes serve to construct a polar plot while there is also a vertical axis pointing upwards (the slope gradient α_{ij}). The middle plot depicts the inverse of this cost function using a 2d polar plot. In this case, the information about the slope gradient α_{ij} is only represented by the use of different colours, as the vertical axis from the previous plot is projected. This view is similar to the one shown in Fig. 5. The right plot serves to represent the directional cost functions $C^{\mathrm{a}}_{ij}(\tilde{x}_{ij})$, $C^{l}_{ij}(\tilde{x}_{ij})$ and $C^{\mathrm{d}}_{ij}(\tilde{x}_{ij})$ as well as the anisotropy $\Upsilon(\tilde{x}_{ij})$ (in red).

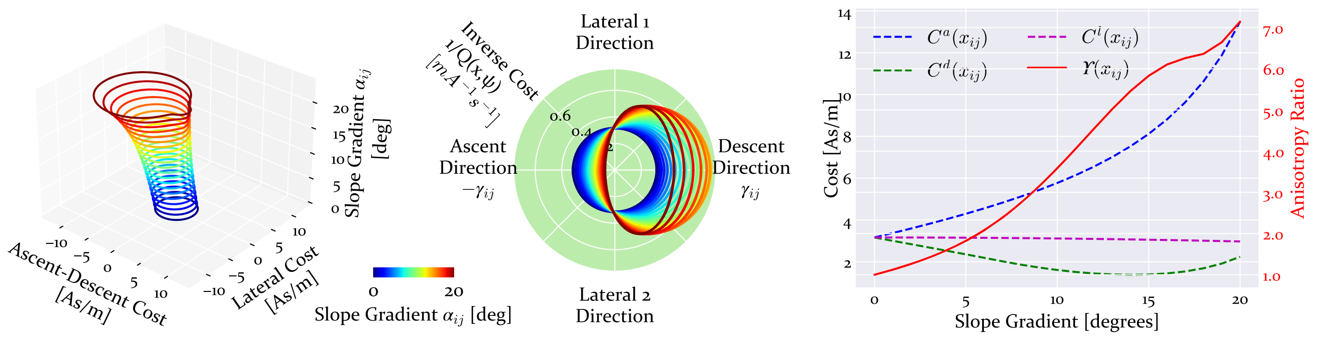
Figure 11a, b corresponds to the case where $\rho = 0.15$ and the slip ratio model is the Wheel one. The difference between them is that the first one uses the anisotropic cost function while the second uses an isotropic one. As can be checked, the cost of the isotropic is equal for all directions of the robot, while in the anisotropic case there are differences according to this direction. For values of α_{ij} close to 20 degrees, the inverse of the anisotropic cost takes the shape of an elongated ellipse, while the isotropic cost remains as a circle. In the isotropic case, this circle becomes smaller as the slope gradient increases, while in the anisotropic case this inverse shrinks in the ascent-descent directions, where the relative angle $\beta(\vec{\psi}, \vec{\gamma}_{ij})$ between the heading $\vec{\psi}$ and the aspect $\vec{\gamma}_{ij})$ directions is either 0 or 180 degrees. The last two subfigures show the anisotropic cost function with $\rho_{ij} = 0.3$ and each of them uses a different slip model. Figure 11c shows the



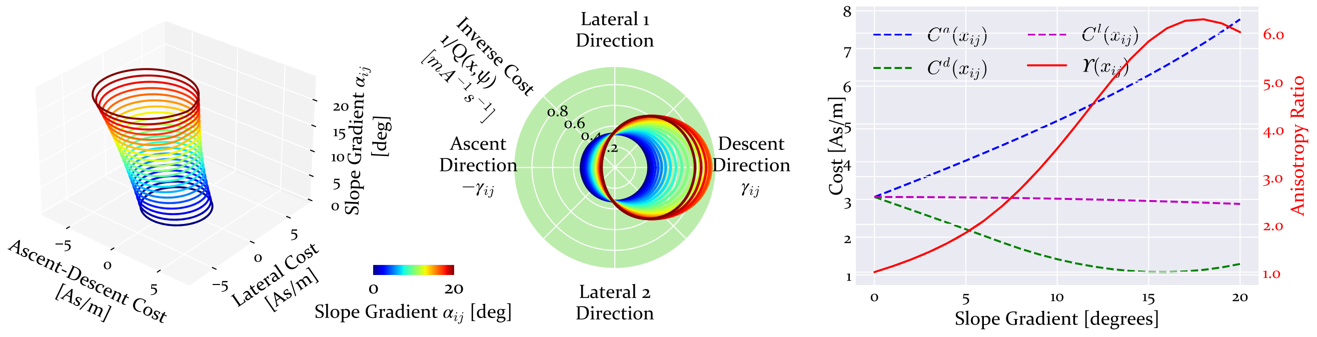
(a) $\rho_{ij} = 0.15$, Wheel model, anisotropic.



(b) $\rho_{ij} = 0.15$, Wheel model, isotropic.



(c) $\rho_{ij} = 0.3$, Wheel model, anisotropic.



(d) $\rho_{ij} = 0.3$, Track model, anisotropic.

Fig. 11 Values of cost returned by each anisotropic cost function. *Left* 3d plot of the values returned by the cost function, where two horizontal axes correspond to the ascent-descent and lateral directions, while the vertical axis is the slope gradient. *Middle* Polar plot of the inverse to the cost function as in Fig. 5, given several values of slope gradient. *Right* directional cost functions and anisotropy. The latter uses the red axes

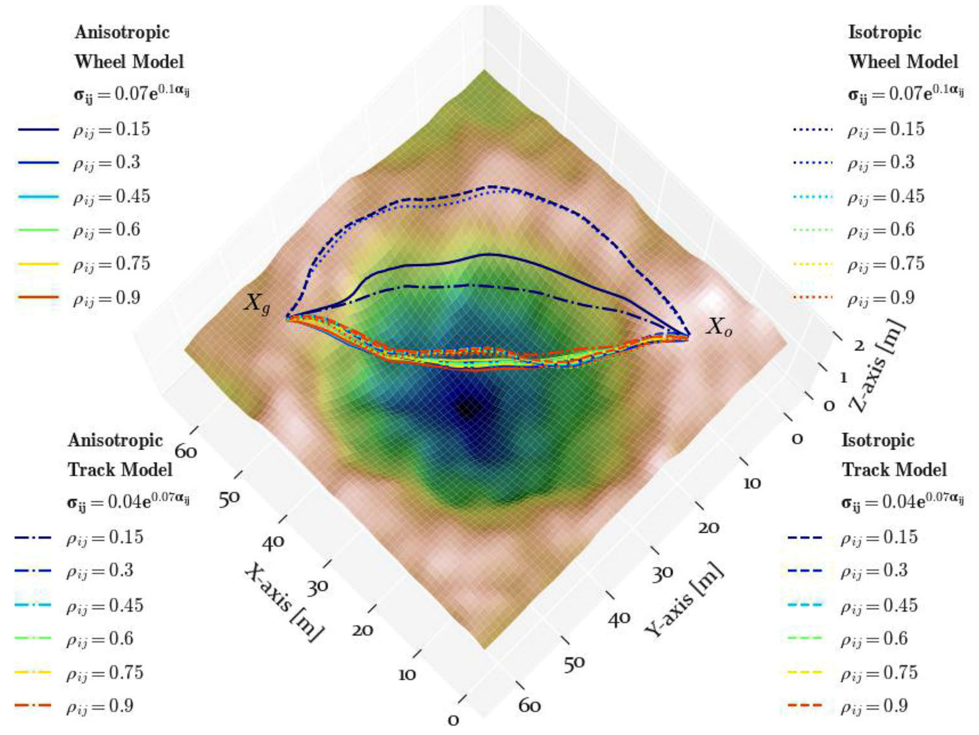


Fig. 12 3d view of the scene with the resulting paths. Results from the first test using anisotropic and isotropic cost functions. The resulting paths connecting two locations, $\tilde{x}_o$ to $\tilde{x}_g$, are depicted. The origin is located at (10 10) m while the goal is at (55 50) m. Note that the 3d view is rotated to provide a better perspective of the obtained paths. The grid $\tilde{\Omega}$ used is a hexagonal regular one, with a resolution Λ of 0.5 m

use of the Wheel model, while Fig. 11d shows the use of the Track model. As can be checked, the cost in the first one is higher in the ascent and descent functions (see right images) due to the higher values of slip ratio (see Fig. 10), while the lateral cost remains the same. This also creates a different anisotropy for values of slope gradient close to 20. The difference in anisotropy is more significant by comparing the first and third subfigures, Fig. 11a, c, where an increase in ρ_{ij} entails higher anisotropy.

With the defined anisotropic and isotropic cost functions based on different configurations of terramechanic functions, a series of paths are planned. Figure 12 depicts these paths connecting two locations of interest: the origin $\tilde{x}_o = x_o$ and the goal $\tilde{x}_g = x_g$. As can be seen, those paths created with low values of ρ_{ij}, between 0.15 and 0.3, go through different places than the rest. The isotropic cost functions with low ρ_{ij} get further from the slopes and surround the crater, taking more distance to reach the goal. The paths with low ρ_{ij} and generated using an anisotropic cost function traverse laterally the slopes, keeping the same elevation. This is because although the ascent and descent costs are high, the lateral cost is still lower (see Fig. 11a). To do a deeper insight into how the total cost function is calculated, Figs. 13 and 14 depict the solution calculated by two configurations: one anisotropic and one isotropic. In Fig. 13 the anisotropy makes the wave propagation from the origin enter the crater. This is because the planner acknowledges that the descent cost is cheaper than the lateral and ascent costs as also seen in the third row of Fig. 11. On the contrary, the isotropic cost from (22) prevents the wave from propagating towards the crater slopes, as it does not address the differences in cost according to direction. For this reason, in the anisotropic case (with $\rho_{ij} = 0.3$ and using the Wheel model) the path traverses the crater, while the isotropic planner finds another path that circumvents the crater by sticking to horizontal surfaces as much as possible. Therefore, the consideration or not of the anisotropy entails different results. Figures 15 and 16 serve to highlight the implications of these differences.

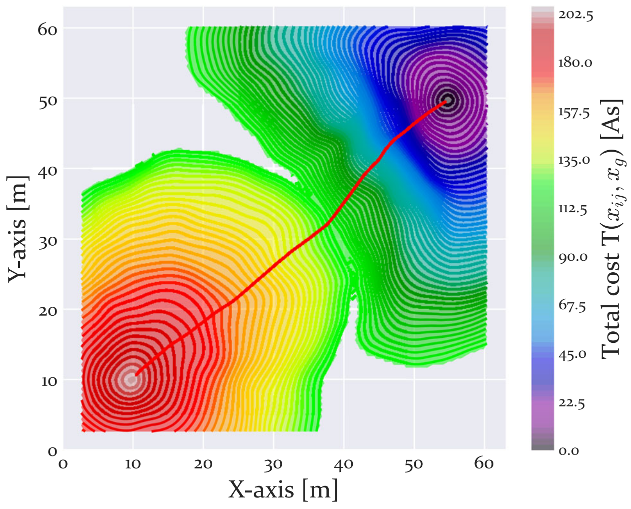


Fig. 13 Total cost calculated using anisotropic cost with $\rho_{ij} = 0.3$ and the Wheel slip model

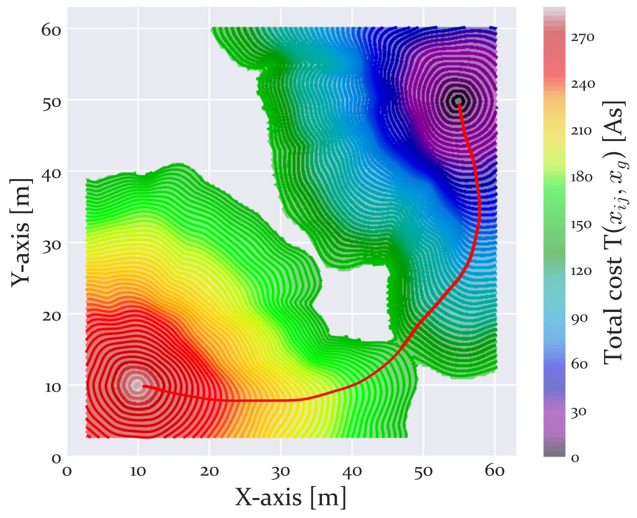


Fig. 14 Total cost calculated using isotropic cost with $\rho_{ij} = 0.3$ and the Wheel slip model

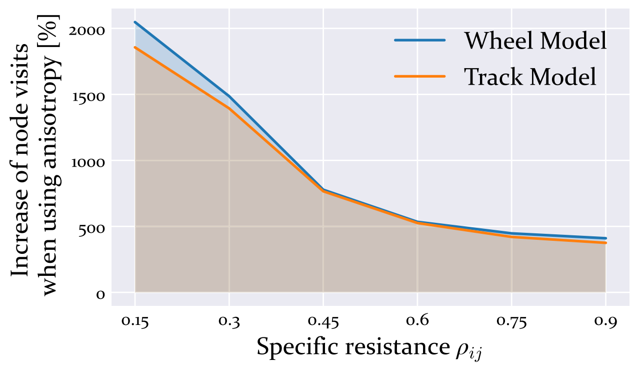


Fig. 15 Increase in the number of visits to nodes when planning using anisotropic cost instead of isotropic cost. Each visit is an update of total cost and characteristic direction values using Eqs. (6), (7) and (8)

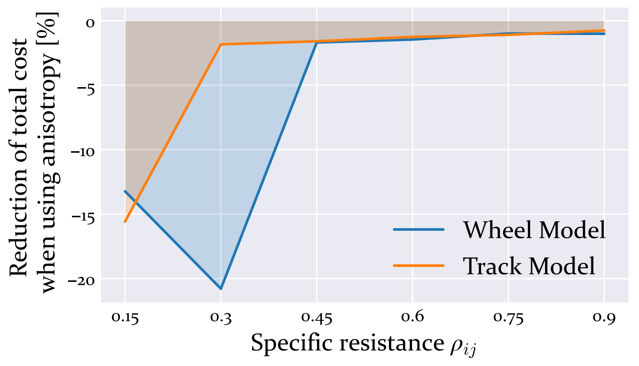


Fig. 16 A measure of how significant is to consider the directional dependency of the cost according to the specific resistance ρ_{ij} and to each slip ratio model. This is calculated according to Eq. (23) given the resulting anisotropic and isotropic total cost values

Figure 15 indicates the increase in the number of node updates when using anisotropic cost in contrast with using isotropic cost. This metric gives an idea about the extra computational load when calculating the solution using anisotropy. As can be denoted, the use of anisotropic cost functions entails a higher of visits to the nodes. This fact goes in line with the computational complexities of anisotropic planners like OUM and isotropic planners like FMM. As a reminder, the computational complexity of OUM is proportional to the overall anisotropy [13]. Figure 15 indicates that for low values of specific resistance ρ_{ij} the number of updates is around 20 times higher than in the isotropic case. This is because the anisotropy increases when the specific resistance is lower, especially when it gets close to zero. This can be checked by comparing the anisotropy plot of Fig. 11a (ρ_{ij} = 0.15) and Fig. 11c ($\rho_{ij} = 0.3$).

Although the increase in computational load is significant, the reduction in total cost produced by considering anisotropy still needs to be measured. Figure 16 shows this reduction for all values of specific resistance and both slip models, calculated using the expression shown in (23).

$$\text{Reduction}[\%] = \frac{T_{\text{aniso}}(\tilde{x}_o, \tilde{x}_g) - T_{\text{iso}}(\tilde{x}_o, \tilde{x}_g)}{T_{\text{iso}}(\tilde{x}_o, \tilde{x}_g)} \times 100 \tag{23}$$

The total cost of paths created using anisotropic cost is $T_{\text{aniso}}(\tilde{x}_o, \tilde{x}_g)$, while $T_{\text{iso}}(\tilde{x}_o, \tilde{x}_g)$ is the total cost of paths created with isotropic cost. As can be checked in Fig. 16, for $\rho_{ij} = 0.15$ the reduction is close to -13% for the Wheel model and -15% for the Track model. For values of specific resistance ρ_{ij} between 0.15 and 0.45, there is some significant difference between the models. The reduction in the Track model decreases rapidly until being around -2.5%. On the contrary, the Wheel Model at $\rho_{ij} = 0.3$ experiences a reduction of around -20.0%.

The exposed data evinces that the use of isotropic cost functions introduces undesired extra total cost in all cases. However, this extra total cost is only significant for $\rho_{ij} < 0.45$ with the Wheel model and for $\rho_{ij} < 0.3$ with the Track model. For this reason, the use of anisotropy in the cost function to address uneven surfaces is only recommended in those circumstances, especially when using a slip model with a high slip ratio such as the Wheel model. The user should evaluate whether the reduction in total cost compensates for the increase in the number of node updates, also quite high for lower values of specific resistance as shown in Fig. 15. For example, for the case in which this planning is carried out offline, the trade-off in question may not be a problem. Yet, the high computational load may be intractable for online planning in the onboard computer of a robotic vehicle.

5 Conclusion

In this paper, we have presented a cost function model aimed at performing anisotropic path planning on terrains containing slopes. By defining this model as the inverse polar function of a displaced ellipse, we make it compatible with the use of anisotropic PDE path planners like the bi-OUM. This cost function considers the energy consumption of the robot according to its heading when it experiences inclination. To better understand the use of the cost function, we present in this paper the results from a simulation test. These results have demonstrated in which situations the anisotropy may be beneficial for making a robot optimally traverse scenarios with slopes. In particular, two slip models were used, in which one of them, the Wheel model, was affected by the inclination more than the other, the Track model. The terrain was considered by not only its shape, through the use of the slope gradient, but also by the use of the specific resistance parameter. Since PDE planners like FMM can use isotropic functions with low computational complexity, they were used to contrast anisotropic ones. The results indicate that the higher the effect of the slip and the lower the value of specific resistance motivate the use of an anisotropic cost function instead of an isotropic one. The benefit of opting for anisotropic cost functions is more prominent in the Wheel model, given a value of specific resistance lower than 0.45.

Finally, we foresee the continuation of this work by studying the use of other PDE path planning methods such as the Fast Sweeping Method (FSM), compatible with anisotropic cost functions [33] and with turning constraints [34]. An improvement to the proposed anisotropic cost function would be the preservation of the stability, as addressed in other work [26], as well as the consideration of non-traversable areas [35]. Another would be the consideration of different functions for the slip ascending and descending as in other approaches [36]. Finally, initial and goal orientations could be addressed by adapting previous approaches done with FMM [37].

Acknowledgements This work was supported by the Andalusian regional government, Spain, under the project entitled “Intelligent multimodal sensor for terramechanic characteristics in Off-Road Vehicles (IMSITER)” with grant reference P18-RT-991.

Funding Funding for open access publishing: Universidad de Málaga/CBUA.

References

1. Delmerico J, Mintchev S, Giusti A, Gromov B, Melo K, Horvat T, Cadena C, Hutter M, Ijspeert A, Floreano D et al (2019) The current state and future outlook of rescue robotics. J Field Robot 36(7):1171–1191
2. Gonzalez-De-Santos P, Fernández R, Sepúlveda D, Navas E, Armada M (2020) Unmanned ground vehicles for smart farms. In Agronomy, IntechOpen
3. Yang G, Steve C (2017) Review on space robotics: toward top-level science through space exploration. Sci Robot. https://doi.org/10.1126/scirobotics.aan5074
4. Sánchez-Ibáñez JR, Pérez-del Pulgar CJ, García-Cerezo A (2021) Path planning for autonomous mobile robots: a review. Sensors 21(23):7898
5. Krüsi P, Furgale P, Bosse M, Siegwart R (2017) Driving on point clouds: motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments. J Field Robot 34(5):940–984
6. Noreen I, Khan A, Habib Z (2016) Optimal path planning using rrt* based approaches: a survey and future directions. Int J Adv Comput Sci Appl 7(11):97–107
7. Dijkstra EW (1959) A note on two problems in connexion with graphs. Numer Math 1(1):269–271
8. Hart PE, Nilsson NJ, Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. IEEE Trans Syst Sci Cybern 4(2):100–107
9. Carsten J, Rankin A, Ferguson D, Stentz A (2007) Global path planning on board the mars exploration rovers. In: Aerospace conference, 2007 IEEE, pp 1–11
10. Maimone MW, Leger PC, Biesiadecki JJ (2007) Overview of the mars exploration rovers’ autonomous mobility and vision capabilities. In: IEEE international conference on robotics and automation (ICRA) space robotics workshop
11. Daniel K, Nash A, Koenig S, Felner A (2010) Theta*: any-angle path planning on grids. J Artif Intell Res 39:533–579
12. Nash A, Daniel K, Koenig S, Felner A (2007) Theta*: any-angle path planning on grids. In: AAAI vol 7, pp 1177–1183
13. Sethian JA, Vladimirsky A (2003) Ordered upwind methods for static Hamilton–Jacobi equations: theory and algorithms. SIAM J Numer Anal 41(1):325–363
14. Sethian JA (1999) Fast marching methods. SIAM Rev 41(2):199–235
15. Liu Y, Liu W, Song R, Bucknall R (2017) Predictive navigation of unmanned surface vehicles in a dynamic maritime environment when using the fast marching method. Int J Adapt Control Signal Process 31(4):464–488
16. Shum A, Morris K, Khajepour A (2016) Convergence rate for the ordered upwind method. J Sci Comput 68(3):889–913
17. Rowe NC, Ross RS (1990) Optimal grid-free path planning across arbitrarily-contoured terrain with anisotropic friction and gravity effects. IEEE Trans Robot Autom 6(5):540–553. https://doi.org/10.1109/70.62043
18. Sun Z, Reif JH (2005) On finding energy-minimizing paths on terrains. IEEE Trans Rob 21(1):102–114
19. Choi S, Park J, Lim E, Yu W (2012) Global path planning on uneven elevation maps. In: 2012 9th international conference on ubiquitous robots and ambient intelligence (URAI), IEEE. pp 49–45

20. Ganganath N, Cheng C-T, Chi KT (2015) A constraint-aware heuristic path planner for finding energy-efficient paths on uneven terrains. IEEE Trans Ind Inf 11(3):601–611
21. Ganganath N, Cheng C-T, Fernando T, Iu HHC, Chi TK (2018) Shortest path planning for energy-constrained mobile platforms navigating on uneven terrains. IEEE Trans Ind Inform 14(9):4264–4272
22. Howard TM, Kelly A (2007) Optimal rough terrain trajectory generation for wheeled mobile robots. Int J Robot Res 26(2):141–166
23. Inotsume H, Kubota T, Wettergreen D (2020) Robust path planning for slope traversing under uncertainty in slip prediction. IEEE Robot Autom Lett. https://doi.org/10.1109/LRA.2020.2975756
24. Inotsume H, Creager C, Wettergreen D, Whittaker WRL (2016) Finding routes for efficient and successful slope ascent for exploration rovers. In: The international symposium on artificial intelligence, robotics and automation in space (i-SAIRAS)
25. Gruning V, Pentzer J, Brennan S, Reichard K (2020) Energy-aware path planning for skid-steer robots operating on hilly terrain. In: 2020 American control conference (ACC), IEEE, pp 2094–2099
26. Shum A, Morris K, Khajepour A (2015) Direction-dependent optimal path planning for autonomous vehicles. Robot Auton Syst 70:202–214
27. Brunner M, Fiolka T, Schulz D, Schlick CM (2015) Design and comparative evaluation of an iterative contact point estimation method for static stability estimation of mobile actively reconfigurable robots. Robot Auton Syst 63:89–107
28. Pérez del Pulgar Mancebo CJ, Romeo Manrique P, Paz Delgado GJ, Sánchez Ibáñez JR, Azkarate M (2019) Choosing the best locomotion mode in reconfigurable rovers. Electronics 8(7):818
29. Arvidson Raymond E, Bell JF, Bellutta P, Cabrol Nathalie A, Catalano JG, Cohen J, Crumpler Larry S, Des Marais DJ, Estlin TA, Farrand WH et al (2010) Spirit mars rover mission: overview and selected results from the northern home plate winter haven to the side of Scamander crater. J Geophys Res Planets. https://doi.org/10.1029/2010JE003633
30. Petres C, Pailhas Y, Petillot Y, Lane D(2005) Underwater path planing using fast marching algorithms. In: Oceans 2005-Europe, vol 2, pp 814–819. IEEE
31. Sutoh M, Otsuki M, Wakabayashi S, Hoshino T, Hashimoto T (2015) The right path: comprehensive path planning for lunar exploration rovers. IEEE Robot Autom Mag 22(1):22–33. https://doi.org/10.1109/MRA.2014.2381359
32. Wakabayashi S, Sato H, Nishida S-I (2009) Design and mobility evaluation of tracked lunar vehicle. J Terrramech 46(3):105–114
33. Kao C-Y, Osher S, Tsai Y-H (2005) Fast sweeping methods for static Hamilton–Jacobi equations. SIAM J Numer Anal 42(6):2612–2632
34. Takei R, Tsai R (2013) Optimal trajectories of curvature constrained motion in the Hamilton–Jacobi formulation. J Sci Comput 54(2–3):622–644
35. Shum A (2014) Optimal direction-dependent path planning for autonomous vehicles. PhD thesis. University of Waterloo
36. Ding L, Gao H, Deng Z, Guo J, Liu G (2013) Longitudinal slip versus skid of planetary rovers' wheels traversing on deformable slopes. In: 2013 IEEE/RSJ international conference on intelligent robots and systems, pp 2842–2848
37. Liu Y, Bucknall R (2016) The angle guidance path planning algorithms for unmanned surface vehicle formations by using the fast marching method. Appl Ocean Res 59:327–344