Dynamic Path Planning for Reconfigurable Rovers using a Multi-layered Grid

J. Ricardo Sánchez-Ibáñez[a,*], Carlos J. Pérez-del-Pulgar[a], Martin Azkarate[b], Levin Gerdes[b], Alfonso García-Cerezo[a]

[a] *Departamento de Ingeniería de Sistemas y Automática, Escuela de Ingenierías Industriales, Universidad de Málaga, Campus de Teatinos, C/ Doctor Ortiz Ramos, S/N, 29071 Málaga, Spain*
[b] *Automation and Robotics Section, European Space Agency, European Space Research and Technology Centre, Keplerlaan 1, 2201 AZ Noordwijk, Netherlands*

Abstract

Autonomy on rovers is desirable in order to extend the traversed distance, and therefore, maximize the number of places visited during the mission. It depends heavily on the information that is available for the traversed surface on other planet. This information may come from the vehicle's sensors as well as from orbital images. Besides, future exploration missions may consider the use of reconfigurable rovers, which are able to execute multiple locomotion modes to better adapt to different terrains. With these considerations, a path planning algorithm based on the Fast Marching Method is proposed. Environment information coming from different sources is used on a grid formed by two layers. First, the Global Layer with a low resolution, but high extension is used to compute the overall path connecting the rover and the desired goal, using a cost function that takes advantage of the rover locomotion modes. Second, the Local Layer with higher resolution but limited distance is used where the path is dynamically repaired because of obstacle presence. Finally, we show simulation and field test results based on several reconfigurable and non-reconfigurable rover prototypes and a experimental terrain.

Keywords: Path planning, Planetary exploration, Rover

1. Introduction

Autonomy is an essential capability for rovers to explore the surface of other planets. The distances from Earth entail big latencies in communications between the rover and the terrestrial ground station. As an example, there is a radio transmission delay of several minutes between Earth and Mars (Lester and Thronson, 2011; Lester et al., 2017). Therefore, direct teleoperation arises as a difficult task to be carried out remotely from Earth. Besides, communications with rovers at the red planet generally occur only a few times per Martian *sol* (solar day) due to the availability of Deep Space Network antennas, conforming a limited time-slot for providing commands and retrieving data from the rover (Bajracharya et al., 2008). These facts are contrary to the necessity of increasing the number of scientificly interesting places visited by rovers. Providing higher autonomy would allow them to traverse longer distances. However, new issues arise since rovers tackle a high uncertainty when they are traveling, i.e., they may encounter unexpected situations, mostly due to terrain shape and/or composition, as well as the existence of stones. These issues affect the traversability for the vehicle. The improper evaluation of the terrain could lead to a fatal situation of the vehicle, compromising the mission as a result. This was the case of the *Spirit* rover, which got stuck in loose sand, making it impossible to continue driving (Ono et al., 2015) and thus bringing the mission to an end. By using traversability information, autonomy can be improved thanks to the use of path planning algorithms, which allow the vehicle to compute onboard a safe path from one location to another.

*Corresponding author
Email address: ricardosan@uma.es (J. Ricardo Sánchez-Ibáñez)

Path planning has been used in Mars exploration missions along with rovers *Spirit*, *Opportunity* and *Curiosity*. A path planning approach, based on two levels (global and local), was deployed on these rovers (Maimone et al., 2007). The main reason behind it is to use data relative to rover surroundings while also considering information relative to the location of elements, such as obstacles, in other areas. As global planning algorithm, the Field-D* was used, initially introduced by Ferguson and Stentz (2006b). With this algorithm, a potential field is computed on a regular grid starting from the goal position to the rover location. Main particularity of this method is the use of an interpolation technique that assigns the values of this potential field to each node, based on the values of its already visited neighbors. By considering that edges connecting nodes are crossable by the path, it results to be smoother than previous methods like D*, where paths are strictly restricted to go through node locations. This algorithm has also re-planning capability, meaning the path can be updated during rover traverse in case the cost of any node is modified. This can happen any time the rover detects an obstacle on its way using its onboard sensors. Besides, this algorithm has been adapted to the use of multi-resolution grids (Ferguson and Stentz, 2006a) to minimize computational resources, saving computation in areas where the level of detail can be simplified. However, it is not clear neither the computational cost of this algorithm nor the steps to extract a path from it, leaving this operation to another algorithm that only focuses on the local planning. In the case of Mars rovers, the *GESTALT* local planner has been used (Carsten et al., 2007). It basically generates a series of arcs starting from the vehicle position, which are later evaluated according to the potential field created by the global planner. As result, the arc with the best evaluation is chosen to be followed by the rover.

An alternative path planning solution to Field-D* is the Fast Marching Method (FMM), originally introduced by Sethian (1999). It is a numerical method that solves the so-called *eikonal* equation, which is an expression defining the behavior of a wave that propagates over a continuous 2D scalar function – also extendable to 3D. Unlike Field-D*, only a quadratic expression can be used along each node of the grid to compute a continuous potential field. Such a potential field represents the arrival time of the wave at each location of the grid. Besides, in contrast to Field-D*, the computational cost of FMM is clearly stated, this being $O(\zeta log\zeta)$, where ζ is the number of nodes the grid is composed of. After computing the potential field, the path is extracted by just making use of a gradient descent method on it (Kimmel and Sethian, 2001; Liu and Bucknall, 2015). However, this path planning solution is not originally meant to include a re-planning capability. Previous research (Philippsen et al., 2008) has focused on the modification of FMM to make it dynamic, obtaining as result an algorithm called E*. Nevertheless, it does not consider the use of maps with multiple resolutions. An example of FMM using a multi-resolution map, but not dynamic re-planning, can be seen in the work of Petres et al. (2005), where an already-known environment is modeled with different levels of detail, obtaining as result sub-optimal but fast-computed paths.

In most cases, path planning algorithms consider traversability data to decide whether a path should go through a certain area or not. It heavily depends on the underlying physics of the terrain-vehicle interaction, i.e. terramechanics. Shape and composition of terrains determine the dynamic behavior of any body in contact with it (e.g. friction and slipping effects). The kinematic configuration of the vehicle, together with the distribution of its masses and inertias, affects the forces exerted on the surface, influencing on traction, as well as on the energy required to move the vehicle. While the terrain features cannot be changed, the rover kinematic configuration could be adapted to it. This is the case for vehicles categorized as *reconfigurable*, which are able to perform several locomotion modes, each one adapted to a particular terrain. A notable example of this feature is the ExoMars rover, which is being conceived to search for signs of life on Mars in the future *ExoMars 2020* mission, lead by the European Space Agency (Vago et al., 2015, 2017). The particularity, with respect to other rovers, is the use of additional joints on top of its legs. Such joints are initially meant to deploy the wheels once the rover has landed on Mars, but later, they can be further used to improve traction on loose soil by the use of a locomotion mode called *wheel-walking* (Woods et al., 2009; Patel et al., 2010). In this sense, Azkarate et al. (2015) performed some experiments using an *ExoMars* rover prototype that demonstrated this statement. Other authors analyzed the performance of a similar locomotion mode, called *push-pull* (Creager et al., 2015), on loose soil as well, getting to

the same conclusions. These modes of locomotion may be helpful in situations where otherwise, being only capable of executing the standard roving or *normal driving* locomotion mode, would result in the rover raising its power consumption or even getting entrapped. Having a good knowledge of the locomotion-terrain relation for a particular rover, a path planning algorithm could take advantage of this information, finding even more optimal paths. However, the design of an algorithm that takes into consideration different locomotion modes is still being investigated. In previous works, path planning algorithms have been developed aimed at the reconfiguration of the vehicle chassis. For example, Brunner et al. (2012, 2014) proposed algorithms to find optimal paths adapting the chassis to overcome obstacles in the form of stairs, while Miró et al. (2010) carried out a research focused on maintaining the stability of the vehicle using FMM. However, none of these works takes into consideration a reconfigurable rover for long-range operations, such as those found in planetary exploration missions.

This paper proposes a path planning algorithm based on FMM that works with a multiple resolution grid made up of two layers. The role of these layers is fully explained in section 2. The first one is used to compute the overall path using a cost function based on the locomotion performance according to the terrain features. The objective is to find the path that minimizes the power consumption, taking advantage of the use of multiple locomotion modes. The second layer is used to perform a novel repairing process that dynamically modifies the path, with the aim of avoiding any obstacle detected by the rover on the spot. Later on, simulation results are provided in section 3 followed by results from a field test. They serve as a way to analyze the main particularities of the proposed algorithm within several planetary exploration situations. From these results, in section 4 are extracted a series of conclusions and some ideas for future work.

2. Path Planning

The proposed path planning algorithm is described in this section. Figure 1 shows a schematic containing its different parts.

The first one is the *Multi-layered Grid*, which contains two layers, each of them with different resolution and range. First, the *Global Layer* uses information provided by satellites to determine the distribution of the different types of terrain present in the mission area. Orbital images, such as the Digital Elevation Maps (DEMs) from HiRISE (McEwen et al., 2007) or the thermal images from Mars Reconnaissance Orbiter THEMIS (Fergason et al., 2006), provide useful data relative to morphology and physical properties of the surface. From DEMs, slopes can be easily computed as well as the location of non-traversable areas, while composition of terrain is not trivial to estimate at first. However, the research carried out by Cunningham et al. (2019) shed light on the problematic of estimating terramechanic parameters in advance using thermal images. This may prove useful to estimate as well the performance of certain locomotion modes by using models previously defined, such as those created in a previous work (Pérez-del Pulgar et al., 2017).

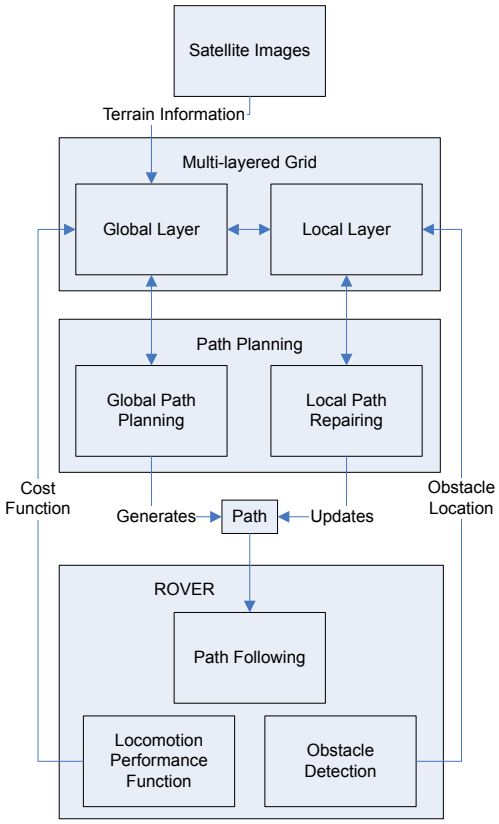


Figure 1: Schematic containing the different parts and interfaces of the proposed path planning algorithm.

Secondly, the *Local Layer* employs information relative to the obstacles detected by the rover during its traverse. As will be seen later, the *Local Layer* is created by subdividing the nodes of the *Global Layer* and updated with data provided by

the onboard sensors. These layers are used by two processes, *Global Path Planning* and *Local Path Repairing* respectively. The first process has the aim of finding a path connecting two points: the rover position and the location of the desired destination. The criteria to determine the shape of this path is based on the available locomotion modes on the vehicle and their performance on the different identified terrains. The *Local Path Repairing* has the function of updating such path whenever necessary in order to avoid those obstacles the rover finds in its way. This path is then used by the rover navigation systems, such as the *Path Following* proposed by Filip et al. (2013), which are out of the scope of this paper.

2.1. Multi-layered Grid

Resolution of maps affects both the computational power needed to work with them and the required size of memory storage. For the same storage, a lower resolution allows to make use of maps with higher range. This is useful to cover a larger area for the mission. However, the drawback of this is that elements like stones that are too small for that resolution cannot be represented, and still these need to be taken into account during the rover traverse. That is why it is essential to make use of local maps with higher detail, which entails an increment in the memory size proportional to the area covered. This paper proposes a way to use information describing the environment using multiple sizes and resolutions. This is thanks to a grid that contains two layers as can be seen in Figure 2. These two layers are named *Global Layer* and *Local Layer*, being the first one meant to be used with low resolution but large areas (e.g. orbital maps) and the second one with high resolution maps covering just certain areas visited by the vehicle (e.g. maps of the surroundings of the rover obtained from its sensors). Each of these layers are formed by nodes, which are basically square areas with a center point and distributed uniformly over the grid. *Global Nodes* and *Local Nodes* are contained in the *Global Layer* and the *Local Layer* respectively. Although the *Local Layer* overlaps the *Global Layer*, its extension is only defined by the number of *Global Nodes* that are subdivided into *Local Nodes*. In other words, the area covered by the *Local Layer* is less or equal to the area covered by the *Global Layer*. In Figure 2, for example, only four *Global Nodes* are subdivided, covering *Local Layer* lesser area than *Global Layer*. Distance between two neighboring *Global Nodes* is represented by the constant Λ, which is, also, the side length of the *Global Node* square area. Analogously, λ is the constant defining the gap between *Local Node* neighbors.

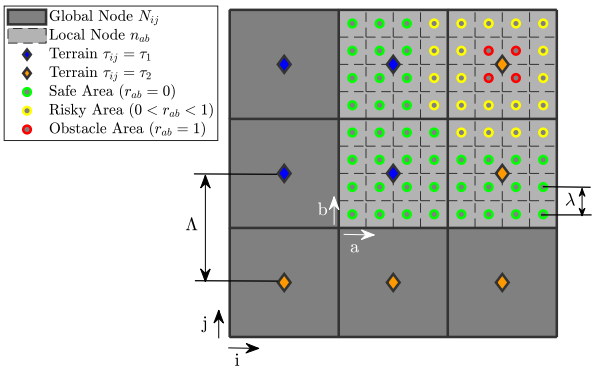


Figure 2: Illustrative image of the multi-layered grid. Two layers are overlapped: *Global Layer*, composed by *Global Nodes*, and *Local Layer*, composed by *Local Nodes*. Each *Global Node* indicates the type of terrain that is contained in its area (τ_1 and τ_2 in this example case) and occupies the same area as a finite number of *Local Nodes*, each of them providing an indication of its state relative to nearby obstacles.

Global Nodes are meant to be used along terrain information deduced from orbital images. For any *Global Node* N_{ij}, where i, j refers to the horizontal and vertical coordinates in the *Global Layer*, τ_{ij} indicates the type of terrain located inside the area of such node. This type is classified within a finite set according to its associated features, in a similar way some terrain classifiers work (Brooks and Iagnemma, 2012; Rothrock et al., 2016). In this way, by just knowing τ_{ij}, terramechanic parameters affecting locomotion performance, such as friction and/or slippage, can be directly associated to it.

Unlike *Global Nodes*, *Local Nodes* do not contain information relative to the type of terrain, but information about their state with respect to nearby obstacles. For this purpose the parameter r_{ab}, also referred here as *risk*, is contained in any *Local Node* n_{ab}, being a and b the horizontal and vertical coordinates relative to the *Global Node* N_{ij} from which is obtained. This parameter uses values going from two limits, 0 and 1. Lower limit or 0 means the node is located on the *safe area*, while a value of 1 is equivalent to an obstacle. Intermediate values correspond to nodes located on the surroundings of the obstacles and they are computed via an operation called *Risk expansion*, which is later detailed.

2.2. Global Path Planning

Initial phase of the proposed path planner is the use of FMM on the *Global Layer*. The main idea behind the *Global Path Planning* is to obtain a matrix T that forms a potential field. Each value T_{ij} of this matrix is associated to a *Global Node* N_{ij} and indicates the value of the *Total Cost* required to go from its location to the final destination. In other words, T_{ij} is the minimum amount of *Cost* integrated along the curve connecting both locations, i.e., the optimal path. In this case the *Cost* C_{ij} consists of a scalar value that represents the difficulty for the vehicle to traverse the area of a *Global Node* N_{ij}. It can be deduced that the value of the *Total Cost* at the *Global Node* corresponding to the destination, N_{dest}, is zero ($T_{dest} = 0$), since from there it is not required for the rover to move to another position until the goal location is changed. From N_{dest} the values of *Total Cost* of the neighboring *Global Nodes* are iteratively computed by using the *eikonal* Equation (1). This equation basically defines how the values of the *Total Cost* are increased as the FMM visits further *Global Nodes* from N_{dest}. In other words, functioning of FMM can be seen as the numerical viscous solution of a wave propagation, expanding from the *Global Node* where $T_{ij} = 0$ (initial condition/desired destination) at a rate determined by the *Cost* values of each node, C_{ij}.

$$||\nabla T_{ij}|| = C_{ij} \tag{1}$$

From the resulting potential field, the optimal path can be later extracted, connecting the location of the rover and the goal position. Since it is intended to minimize the energy required by the vehicle to perform its traverse and reach the desired destination, this parameter is the one considered here as the *Total Cost*. Thus, the *Cost* is also defined here in energetic terms and is shown in Equation (2). Since it is the spatial derivative of the *Total Cost*, and energy is the integration of power over time, *Cost* is here equivalent to the ratio between the power P and the velocity v of the rover, which in this case is considered to be a constant value. In this way, if the *Cost* (power/speed) is integrated over the length of any path obtained using FMM, the value of the *Total Cost* (energy) is obtained as result. The Power function P in Equation (2) considers the use of rovers with reconfiguration capability, providing the value of the instantaneous power depending on the locomotion mode l, the type of terrain τ_{ij} and the value v of speed used. It can be built upon models and/or experimentation such as in the work of Pérez-del Pulgar et al. (2017). Locomotion mode chosen to traverse the *Global Node* N_{ij}, l_{ij}, is the one that makes C_{ij} take the minimum value and is contained in the set L of all the available modes in the rover.

$$C_{ij} = \min_{l \in L} \frac{P(l, \tau_{ij}, v)}{v} \quad \Rightarrow \quad l = l_{ij} \tag{2}$$

Following the work of Sethian (1999), the *eikonal* Equation (1) is discretized by means of finite differences, having as result Equation (3). Tx_{ij} and Ty_{ij} are the values of the *Total Cost* of the horizontal and vertical neighbors of the *Global Node* N_{ij} respectively. The criteria to choose which neighbor is used with respect to each axis is shown in Equations (4) and (5).

$$\left(\frac{T_{ij} - Tx_{ij}}{\Lambda}\right)^2 + \left(\frac{T_{ij} - Ty_{ij}}{\Lambda}\right)^2 = C_{ij}^2 \tag{3}$$

$$Tx_{ij} = \min\{T_{i-1j}, T_{i+1j}\} \tag{4}$$

$$Ty_{ij} = \min\{T_{ij-1}, T_{ij+1}\} \tag{5}$$

The quadratic equation (Equation (3)) may however provide more than one solution for T_{ij}. It is important to ensure that the upwind condition in Equation (6) is always true while propagating the Fast Marching wave, so as to ensure no local minimums are created in the process.

$$(T_{ij} > Tx_{ij}) \vee (T_{ij} > Ty_{ij}) \tag{6}$$

The final implementation of the propagation equation, complying with the upwind condition (6), is shown in Equation (7). As can be deduced, in those cases where the discretized *eikonal* does not ensure the condition in (6), an alternate way to compute T_{ij} is used, corresponding basically to the *Dijkstra* method.

$$T_{ij} = \begin{cases} \frac{Tx_{ij} + Ty_{ij} + \sqrt{2(\Lambda C_{ij})^2 - (Tx_{ij} - Ty_{ij})^2}}{2}, \\ \qquad |Tx_{ij} - Ty_{ij}| \leq \Lambda C_{ij} \\ \min\{Tx_{ij}, Ty_{ij}\} + \Lambda C_{ij}, \\ \qquad otherwise \end{cases} \tag{7}$$

Next, it is given here an explanation about how FMM visits each *Global Node* to execute Equation (7), as well as the respective pseudo-code used, which can be found in Algorithm 1. Initially the state of all *Global Nodes*, $N_{ij}.s$, is labeled as *Far*, meaning the algorithm has not yet reached them.

Algorithm 1: Global FM Propagation

$T_{ij} = \infty, \ \ \forall N_{ij} \in$ Global Layer
$N_{ij}.s = Far, \ \ \forall N_{ij} \in$ Global Layer
$T_{dest} \leftarrow 0$
$N_{dest}.s = Accepted$
$FWlist \leftarrow \{\}$
$FWlist \leftarrow N_{ij} \ \ \forall N_{ij} \in N_{dest}.nb \mid C_{ij} \neq \infty$
$N_{ij}.s \leftarrow Considered, \ \ \forall N_{ij} \in N_{dest}.nb \mid C_{ij} \neq \infty$
$T_{ij} \leftarrow \min\{T_{ij}, Eq.(7)\}, \ \ \forall N_{ij} \in N_{dest}.nb \mid C_{ij} \neq \infty$
repeat
  $N_{next} \leftarrow N_{ij} \mid \min_{N_{ij} \in FWlist} T_{ij}$
  Erase N_{next} in $FWlist$
  $N_{next}.s \leftarrow Accepted$
  $FWlist \leftarrow N_{ij} \ \ \forall N_{ij} \in N_{next}.nb \mid N_{ij}.s = Far$
  $N_{ij}.s \leftarrow Considered, \ \ \forall N_{ij} \in N_{next}.nb \mid N_{ij}.s = Far$
  $T_{ij} \leftarrow \min\{T_{ij}, Eq.(7)\}, \ \ \forall N_{ij} \in N_{next}.nb \mid N_{ij}.s = Considered$
until $(N_{rover}.s = Accepted) \vee (FWlist = \{\})$;

One of these *Global Nodes*, N_{dest}, has its state labeled as *Accepted* and its value of *Total Cost*, T_{dest}, is set to zero. The state *Accepted* means its value of *Total Cost* is already computed and definitive. Then, a list called *FWlist* is initialized, empty at first, and is meant to contain *Global Nodes* with the state labeled as *Considered*. These *Considered Global Nodes* form the front wave of the propagation process, acting as frontier between *Far* and *Accepted Global Nodes*. The next step is to label all *von Neumann* neighbors of N_{dest}, $N_{dest}.nb$, as *Considered* (except those containing obstacles, which can be distinguished by its corresponding value of cost C_{ij} being equal to ∞) and introduce them into *FWlist*, computing for them preemptive values of *Total Cost* using Equation (7). From this point, a loop is continuously executed until either *FWlist* gets empty or the *Global Node* containing rover location, N_{rover}, is labeled as *Accepted*. This loop consists of a series of steps as follows: first, the *Global Node* from *FWlist* with the lowest value of *Total Cost*, N_{next}, is extracted, then it is labeled as *Accepted* and finally, for each of its *von Neumann* neighbors, $N_{next}.nb$, the value of *Total Cost* is re-computed in case it is yet preemptive and, also, they are introduced into *FWlist* if their state is *Far*, updating them as *Considered* in the process.

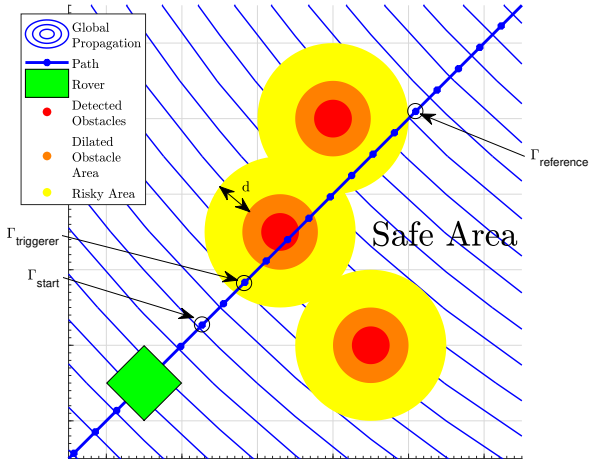
Once the potential field of T values is computed, the next step is the extraction of the path that minimizes the *Total Cost*. To do this, a gradient descent method is applied over the field using Equation (8), starting from a waypoint placed at the location of the rover. In order to better distinguish between waypoints computed using *Global Layer* from those using *Local Layer*, the first ones are referred to as *Global Waypoints* and the second ones as *Local Waypoints*. Γ_k is the k *Global Waypoint* of the trajectory. The number of *Global Waypoints* will depend on both the length of the resulting path and the chosen step size ρ (which is a value lower than 1). The last *Global Waypoint* is placed further than a certain distance (1.5 Λ in our case) of the goal position since the gradient at that location degenerates because of the discretization method used.

$$\Gamma_k = \Gamma_{k-1} - \rho \nabla T_{k-1} \ \ \forall k = 1, 2, ... \qquad (8)$$

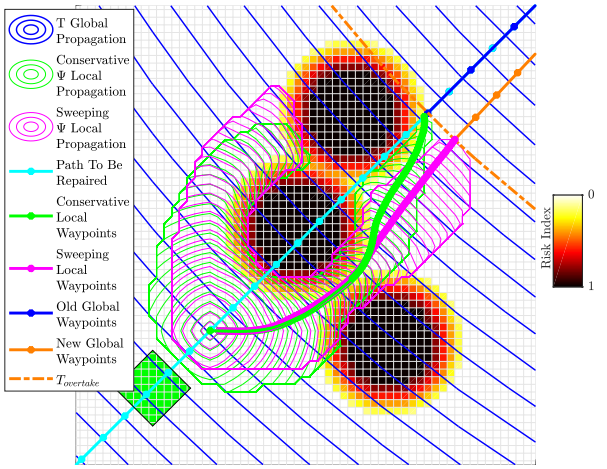
Finally, the computation of ∇T_{k-1} is done using linear interpolation with the values of T of the nearby 4 *Global Nodes*, which are also computed using finite differences.

2.3. Local Path Repairing

Although the *Global Path Planning* stage can initially provide a path, the information it is based on can be too imprecise to ensure the rover advances safely. This is mainly due to the possibility of encountering elements such as rocks that were not visible due to the low resolution of the global map. Therefore, the rover must have a means to update the path shape whenever its sensors detect an obstacle. However, the resolution of the *Global Layer*, usually in the order of the size of the rover, can be too big to represent it with the proper level of detail. Thus, this paper proposes a new stage, called *Local Path Repairing*, which basically consists of the application of a heuristic version of FMM, called FM*, on the *Local Layer*, using a small area of the map but with higher resolution. As a result, a section of path is computed, which serves to avoid obstacles as well as to guide the rover to a safe *Global Waypoint* from which it can continue its traverse. Depending on the criteria chosen to determine such *Global Waypoint*, we can distinguish between two possible approaches for the *Local Path Repairing*: the *Sweeping approach*, which results on computing an all-new path, or the *Conservative approach*, which tries to continue the previous planned path as soon as possible.



(a) Example case where the rover (green) encounters three obstacles (in red) on its way that were not considered during the *Global Path Planning* stage.



(b) On the *Local Layer*, obstacles are mapped into the grid and the risk values r of the areas around them are computed. Next, depending on the chosen approach (either *Conservative* or *Sweeping*), FM* is executed to create the local section of the repaired path.

Figure 3: The *Local Path Repairing* process is composed by several operations. First, when obstacles are in the way of the rover the process is triggered (a). *Local Nodes* are created and an operation called *Risk Expansion* computes the values of r for them (b). Finally, a heuristic FMM propagation is computed using these r values to get a new path.

Figure 3 shows a series of operations that are carried out during the *Local Path Repairing* stage. The first step consists of determining whether the path must be repaired or not each time a new obstacle is detected. As can be seen in Figure 3a, the area occupied by obstacles is dilated by a certain amount depending on the dimensions of the rover, ensuring that the vehicle will not collide with them while following the resulting path. Then, the criterion used to trigger the rest of the steps is whether any of the *Local Nodes* containing an obstacle or part of it is located under a distance threshold d to any *Global Waypoint*. In that case, the current path is considered to be too close to obstacles, so the rest of steps are executed. The distance threshold d is chosen complying with Equation (9), so as to minimize the error committed by not checking intermediate points between consecutive *Global Waypoints*. The *Global Waypoint* $\Gamma_{triggerer}$ is marked as the first one (following the order in which the rover arrives at each of them) that is considered to be close to an obstacle so as to trigger the repairing process. Then, from that *Global Waypoint* another prior to it is searched for, called Γ_{start}. It can be either a *Global Waypoint* placed further than distance d from $\Gamma_{triggerer}$, or even the vehicle position, in case it is closer to $\Gamma_{triggerer}$. Γ_{start} acts as the reference position from which the rest of the path is re-computed, meaning all *waypoints* prior to it are not modified in the repairing process. Another *Global Waypoint*, $\Gamma_{reference}$, is also considered later, and is basically the first waypoint that is placed further than distance d after the path has gone close to the obstacle. Figure 3a depicts three possible areas each *Local Node* can be part of: *Obstacle area*, *Risky area* and *Safe area*. *Obstacle area* refers to those *Local Nodes* containing obstacles, while *Risky area* are those located under distance d to the previous area. On the other hand, *Safe area* is formed by those *Local Nodes* located further than distance d from obstacles.

$$\rho \leq d/\Lambda \tag{9}$$

Before computing the FM* propagation on the *Local Layer* to retrieve the new path, it is necessary to define the values of r_{ab} for each *Local Node* n_{ab}. This is done by executing a process called *Risk Expansion*, in which a gradient is created around obstacles to serve as a repulsive potential field and make the resulting path get further from them. The main idea behind this has been applied in other research (Petres et al., 2005; Valero-Gomez et al., 2013). The pseudo-code of this process can be found in Algorithm 2. While the *Local Nodes* contained in the *Obstacle area* are labeled as *Accepted* and have their value of *risk* set to 1, this parameter is initialized to 0 for the rest, whose state is

Algorithm 2: Risk Expansion

$r_{ab} = 1, \ \ \forall n_{ab} \in$ Obstacle Area
$r_{ab} = 0, \ \ \forall n_{ab} \notin$ Obstacle Area
$FWlist \leftarrow \{\}$
$FWlist \leftarrow n_{ab}, \ \ \forall n_{ab} \in$ Local Layer
  $\mid (any\ n' \in n_{ab}.nb, r' = 0) \wedge (r_{ab} = 1)$
repeat
  $n_{next} \leftarrow n_{ab} \mid \max_{n_{ab} \in FWlist} r_{ab}$
  Erase n_{next} in $FWlist$
  $n_{next}.s = Accepted$
  for $n_{ab} \in n_{next}.nb \mid n_{ab}.s \neq Accepted$ **do**
    $r' \leftarrow Eq.(11)$
    if $r' > r_{ab}$ **then**
      $r_{ab} \leftarrow r'$
      if $n_{ab}.s = Far$ **then**
        $n_{ab}.s \leftarrow Considered$
        $FWlist \leftarrow n_{ab}$
until $FWlist = \{\}$;

set to *Far*. Then, all those *Local Nodes* that are within the *Obstacle Area* but have neighbors with $r = 0$ are introduced into the list *FWlist*. From these nodes is expanded a wave that will set values of r going from 1 to 0, stopping at those *Local Nodes* whose value of r is already higher than the one the wave intends to set. The *eikonal* Equation used for *Risk expansion* is provided in (10), as well as its discretized version in (11). In Figure 3b can be checked how values of r between 0 and 1 are set forming the repulsive potential fields, contained inside the *Risky areas*.

$$||\nabla r_{ab}|| = -\frac{1}{d} \tag{10}$$

$$r_{ab} = \begin{cases} \frac{rx_{ab}+ry_{ab}-\sqrt{2(\lambda/d)^2-(rx_{ab}-ry_{ab})^2}}{2}, \\ \qquad\qquad |rx_{ab} - ry_{ab}| \leq \lambda/d \\ \max\{rx_{ab}, ry_{ab}\} - \lambda/d\ , \\ \qquad\qquad otherwise \end{cases} \tag{11}$$

After the *Risk Expansion*, the values of r are used to build *Local Cost* c_{ab} as seen in (12) for any *Local Node* n_{ab}. This cost parameter is used in the *eikonal* Equation (13), whose discretized version is shown in (14). This equation is used for the propagation of the FM* wave starting from the *Local Node* closest to Γ_{start}, as seen in Figure 3b. As result, a potential field is obtained, from which is intended to later extract a series of *Local Waypoints* to rebuild the path. This potential field is formed by values of parameter t_{ab} for each *Local Node* n_{ab}, which is analogous to the *Total Cost* in the *Global Layer*.

Algorithm 3: Local FM* Propagation

$t_{ab} = \infty, \ \ \forall n_{ab} \in$ Local Layer
$n_{ab}.s = Far, \ \ \forall n_{ab} \in$ Local Layer
$n_{start} \leftarrow n\ nearest\ to\ \Gamma_{start}$
$t_{start} \leftarrow 0$
$n_{start}.s = Accepted$
$FWlist \leftarrow \{\}$
Add n_{ab} to
  $FWlist \ \ \forall n_{ab} \in n_{start}.nb \mid n_{ab}.s = Far$
$n_{ab}.s \leftarrow Considered, \ \ \forall n_{ab} \in n_{start}.nb \mid$
  $n_{ab}.s = Far$
$t_{ab} \leftarrow \min\{t_{ab}, Eq.(13)\}, \ \ \forall n_{ab} \in n_{start}.nb \mid$
  $n_{ab}.s \neq Accepted$
repeat
  $n_{next} \leftarrow n_{ab} \mid \min_{n_{ab} \in FWlist} h_{ab}$
  Erase n_{next} in $FWlist$
  $n_{next}.s = Accepted$
  $FWlist \leftarrow n_{ab} \ \ \forall n_{ab} \in n_{next}.nb \mid n_{ab}.s =$
   Far
  $n_{ab}.s \leftarrow Considered, \ \ \forall n_{ab} \in n_{next}.nb \mid$
   $n_{ab}.s = Far$
  $t_{ab} \leftarrow \min\{t_{ab}, Eq.(13)\}, \ \ \forall n_{ab} \in$
   $n_{next}.nb \mid n_{ab}.s \neq Accepted$
until n_{next} *satisfies either (15) or (17)*;

$$c_{ab} = 1 + r_{ab} \tag{12}$$

$$||\nabla t_{ab}|| = c_{ab} \tag{13}$$

$$t_{ab} = \begin{cases} \frac{tx_{ab}+ty_{ab}+\sqrt{2(\lambda c_{ab})^2-(tx_{ab}-ty_{ab})^2}}{2}\ , \\ \qquad\qquad |tx_{ab} - ty_{ab}| \leq \lambda c_{ab} \\ min\{tx_{ab}, ty_{ab}\} + \lambda c_{ab}\ , \\ \qquad\qquad otherwise \end{cases} \tag{14}$$

In Algorithm 3 is contained the pseudo-code of the FM* propagation using the *Local Layer*. It works similarly to the propagation in *Global Path Planning*, but with the main difference of working in this case with *Local Nodes* instead of *Global Nodes*. Another difference is the requirement to fulfill in order to stop the propagation loop. In the case of *Local Path Repairing* it consists of reaching a safe *Local Node* n_{next} that satisfies a certain stop

condition. Such condition depends on whether it is desired to strictly keep the rover close to the initially computed path or not. Therefore, we can find up to two different approaches that also affect the heuristic function to use during the computation of the propagation wave: *Sweeping* and *Conservative* approaches (pink and green coloured lines respectively in Figure 3b).

2.3.1. Sweeping Approach

In this case it is prioritized to reach a *Local Node* from which the rest of *Global Waypoints* are computed again. In other words, from the location of such *Local Node* the gradient descent method introduced in (8) can be applied on the *Global Layer* again using the values of *Total Cost*. The main idea behind this approach is that the *Local Node* where the local propagation stops satisfies the condition (15).

$$T_{next} \leq T_{overtake} \tag{15}$$

where T_{next} is the value of *Total Cost* corresponding to the *Local Node* n_{next} obtained by interpolating with the values of *Total Cost* of the surrounding *Global Nodes*. $T_{overtake}$, on the other hand, is the value of *Total Cost* corresponding to the *Global Waypoint* $\Gamma_{reference}$. Therefore, by using this approach it is searched for a position beyond the obstacle to be avoided keeping track of the *Total Cost* values computed in the *Global Path Planning* process. Due to the fact that the gradient descent method always makes the path go towards lower values of *Total Cost*, the new *Global Waypoints* will be placed further from the obstacle. Then, since in this case it is more important to search through nodes with values of *Total Cost* lower than $T_{overtake}$, the heuristic function (16) is used. χ is the distance traversed in the path from T_{start} to $T_{overtake}$. It is worth mentioning that this function contemplates the possibility of encountering nodes that, although they meet condition (15), may not be safe, meaning the FM* on the *Local Layer* should continue propagating.

$$h_{ab} = t_{ab} + \max\left\{0, \frac{T_{ab} - T_{overtake}}{T_{start} - T_{overtake}}\chi\right\} \tag{16}$$

2.3.2. Conservative approach

Unlike the previous approach, this one consists of making the local path rejoin with one of the existing *Global Waypoints* placed beyond the obstacle and risky areas. This approach is more restrictive since the local propagation can only stop at the *Local Node* $n_{reference}$, which is the closest to the location of $\Gamma_{reference}$, as denoted in (17).

$$n_{next} = n_{reference} \tag{17}$$

As can be checked in the example case introduced in Figure 3b, this rejoining tends to be less smooth than the one done under the *sweeping approach* since the local propagation does not follow the descending direction of the *Total Cost* values. It can be deduced that this strategy is not focused on searching for the optimal solution in terms of distance and/or power consumption, but instead for ensuring that the rover will stick to the original planned path. This can be useful whenever during a planetary exploration mission it is prioritized that the rover follows the original path as closely as possible. Besides, there are some advantages in terms of computation with respect to the previous approach: it demands less data storage, since the rover is no longer dependant on the information provided by the *Global Layer*. Also, it is not required to interpolate the values of *Total Cost* of each *Local Node*, so this computation process is also avoided. Finally, as has been stated, the computation of the *Local Path Repairing* under this approach searches for the shortest path to rejoin to the previous path while avoiding surrounding obstacles. Therefore, the corresponding heuristic function considers the distance to the position of $n_{reference}$, as can be checked in equation (18). In this way, the *Local Path Repairing* prioritizes the expansion of the propagation wave towards those nodes closer to $n_{reference}.p$.

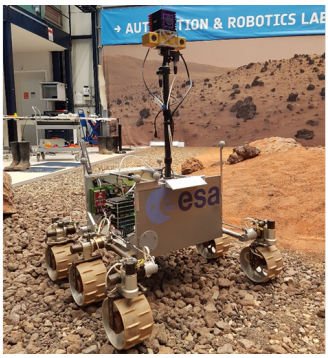
$$h_{ab} = t_{ab} + |n_{ab}.p - n_{reference}.p| \tag{18}$$

3. Results

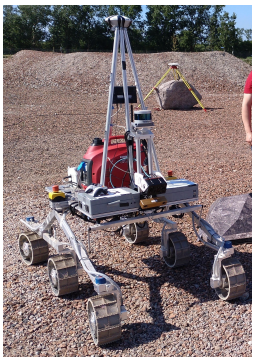
Once the proposed path planning algorithm has been detailed, it is validated by means of simulations carried out using MATLAB software and a field test. The code used for the simulations and the real experiment can be found on GitHub repositories [1][2]. The purpose of the first simulation test is to analyze the validity of the *Global Path Planning* on long-range navigation. In particular, data related

[1] `https://github.com/spaceuma/ARES-DyMu_matlab`

[2] `https://github.com/ESA-PRL/planning-path_planning`

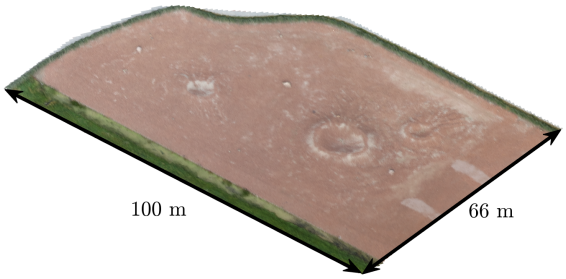


(a) ExoMars Testing Rover (ExoTeR).

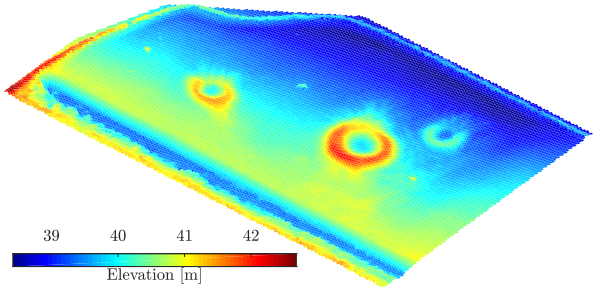


(b) Heavy Duty Planetary Rover (HDPR).

Figure 4: Close-up images of the rovers used for the simulations (a) and the field test (b).



(a) 3d model using an orthonormal image as texture.



(b) Digital Elevation Map.

Figure 5: Overview of the experimental terrain used to perform the simulation and field tests.

to the locomotion performance of the reconfigurable rover shown in Figure 4a is used. This rover, called *ExoTeR*, (Azkarate et al., 2015) is capable of executing two locomotion modes: *Normal driving* and *Wheel-walking*. The first mode is the usual locomotion used by rovers: by rolling the wheels the vehicle advances. On the other hand, *Wheel-walking* consists of deploying and retrieving the legs so as to improve traction, as already stated in the Section 1. The last simulation serves to validate the use of the *Local Path Repairing* process under the *sweeping approach* on one of the paths computed before. Later on, results from a field test carried out with a single-locomotion rover, shown in Figure 4b, are also introduced, showing the performance of *Local Path Repairing* under the *conservative approach*. This mobile platform has similar dimensions as the ExoMars rover and is also equipped with a rocker-bogie locomotion system. Its purpose is to replicate the sensor architecture that will be located onboard the rover to be sent to Mars (Hewitt et al., 2018).

All of these tests were done using the Martian-looking environment shown in Figure 5a. It consists of a real experimental terrain[3] that is located close to the European Research and Technology Centre (ESA-ESTEC) in Noordwijk, The Netherlands. A Digital Elevation Map obtained from it was used for the simulations, and it is also the scenario where the field test took place.

3.1. Long-range navigation

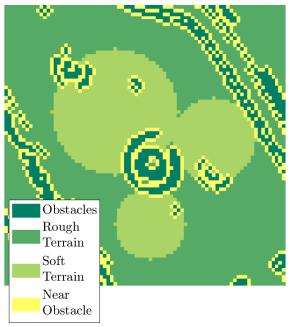
The first experiment consists of executing the *Global Path Planning* stage to obtain the optimal path to reach a certain destination. It is executed several times, starting from various positions over the map. To do this, the *Global Layer* is built using $\Lambda = 1\,\mathrm{m}$, storing values of elevation from a DEM (see Figure 5b). Then, since the terrain contains craters and big rocks, the value of slope is computed at each *Global Node*, considering it an obstacle if it is too pronounced. Besides, as can be seen in Figure 6a, there are two traversable areas, each one corresponding to one kind of terrain: *Rough* and *Soft*. Moreover, a third type of terrain is introduced here, named *Near Obstacle*. This type serves to smooth cost transition between obstacles and traversable area, since cost of the first one is considered as infinite. In this way, resulting paths go further from obstacles, which is desirable as well to avoid the use of gradient descent method near them. Then, the FMM propagation is executed. To do this, the values of C_{ij} for each *Global Node* N_{ij} are chosen according to the Table 1, depending on the type of terrain and the chosen locomotion mode. These values come from previous work (Pérez-del Pulgar et al., 2017), in which simulation models were used to analyze the power consumption of each locomotion mode. Two terrain parameters were considered for this purpose: μ and s. First of them depends

[3]DMS Coordinates: 5212'55.0"N 425'39.1"E

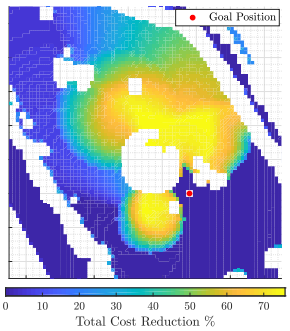
τ	μ	s	$C(l_{ww})$	$C(l_d)$
Rough	0.07	0.05	0.236	**0.088**
Soft	0.45	0.5	**0.236**	1.074
N.O.	–	–	**0.472**	**2.148**
Obstacle	–	–	∞	∞

Table 1: Table from which the values of cost C_{ij}, given in kW s/m, are set for any *Global Node* N_{ij}, depending whether the locomotion mode is *Normal driving* (l_d) or *Wheel-walking* (l_{ww}), as well as terrain parameters μ and s.

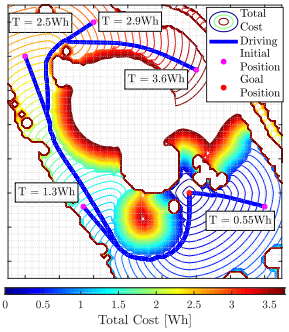
on the normal force applied by the rover on the surface, being the *rolling resistance* for *Normal driving* and the *friction resistance* for *Wheel-walking*. The second parameter, s, is the slip ratio experienced by the rover while it advances i.e., odometry vs ground velocity. Two cases are considered to better clarify the importance of taking more than one locomotion mode into account when planning paths: in one case, only *Normal driving* is available and in the other case *Wheel-walking* is also used. In this way, a typical rover with just one locomotion mode is compared to a reconfigurable rover. Figure 6b shows a plot where it is indicated for each *Global Node* the percentage of *Total Cost* that is reduced by considering the *Wheel-walking* mode. Since this is the best locomotion mode to use in *Soft* terrain, it is obvious that *Total Cost* is reduced within the area containing this type of terrain. Nevertheless, it can be checked how in some parts of the area containing *Rough* terrain the *Total Cost* is also reduced. This is mainly due to the positioning of the different terrains and the obstacles with respect to the destination location. The resulting paths starting from certain positions in both cases are shown in Figures 6c and 6d. In the second case, these paths are more likely to go through the *Soft* terrain thanks to having *Wheel-walking* available. Moreover, in both figures the computed matrix of *Total Cost* is provided, along with annotations about the numerical values corresponding to each path in both cases. As can be deduced, the *Total Cost* associated to some of those paths is lower in the case of using two locomotion modes than in the first case. Therefore, the results from this experiment justify the use of a cost function based on locomotion-terrain interaction along with the FMM for long-range navigation, specially for those vehicles that are capable to adapt to certain types of terrain with different locomotion modes.



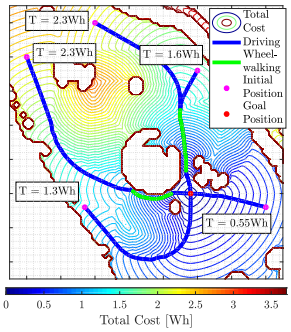
(a) Distribution of the different types of terrain over the *Global Layer*.



(b) Percentage of *Total Cost* reduced considering also *Wheel-walking*.



(c) *Global Path Planning* only considering *Normal driving*.

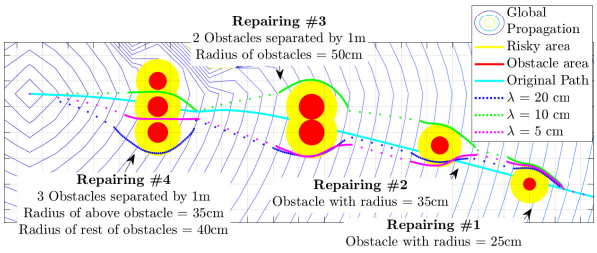


(d) *Global Path Planning* considering *Normal driving* and *Wheel-walking*.

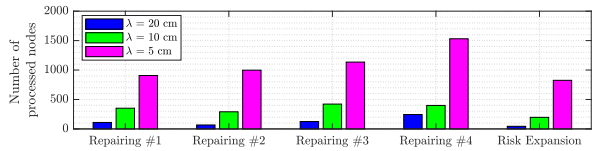
Figure 6: Results from the first experiment. The distribution of the different terrains is shown in (a). The *Global Path Planning* is executed twice: one considering just *Normal driving* and other also taking *Wheel-walking* into account. In this way, *Total Cost* to arrive at the destination is reduced for some areas (b). The paths starting from different positions in both cases are also provided (c), (d).

3.2. Obstacle avoidance

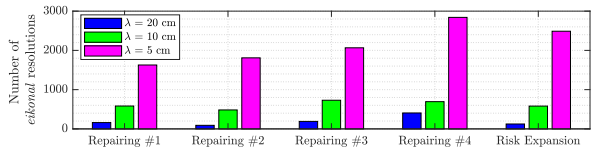
The second experiment is focused on analyzing the performance of the *Local Path Repairing* process under the *sweeping approach*. In order to do this, the path starting from the bottom right corner of the map in both cases is considered as the path to be repaired. Then, it is introduced a series of small obstacles as can be checked in Figure 7a, corresponding to different situations the rover may encounter with during a mission. The area that is supposed to be occupied by these obstacles consists of circles with diverse radii. Since the dimensions of *ExoTeR* are $70 \times 70 \times 40$ cm (Azkarate et al., 2015) and the value of Λ is 1 m, the obstacle areas are set to have a radius between 25 and 50 cm. The chosen value of distance d is 50 cm, which com-



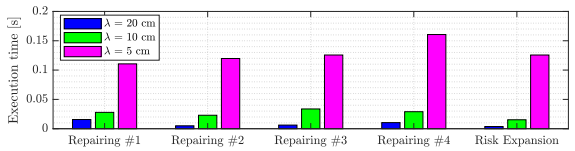
(a) Resulting Paths using different values of λ.



(b) Number of processed *Local Nodes*.



(c) Number of times *eikonal* is used.



(d) Execution time in seconds.

Figure 7: Results from second experiment. The shape of the repaired paths after executing the *Local Path Repairing* process 4 times is shown (a), as well as information relative to the computational power used, in the form of the number of processed nodes (b), the number of times the *eikonal* equation is used in total and the elapsed time to do the computation.

plies with Equation (9) since the value of ρ used to compute the paths in the previous experiment was 0.4. Then, three cases are considered in this experiment, each of them corresponding to the use of a different value of λ: 5, 10, and 20 cm. In this way the influence of this parameter with respect to the *Global size* Λ on the behavior of the *Local Path Planning* is analyzed. Figure 7a illustrates the differences in the shape of the original path and the ones resulting after the execution of the repairing process for the three cases. By using a *Local Layer* with a higher resolution, i.e., a lower value of λ, the repaired path is smoother. Besides, as can be seen in the fourth repairing, the resulting path can also go through smaller gaps. Nevertheless, a lower value of λ implies a heavier computation. This is demonstrated in Figures 7b, 7c, and 7d for the three cases. In Figure 7b it is measured the number of *Local Nodes* visited for each repairing as well as the average per obstacle during the *Risk Expansion* process. The total number of times the *eikonal* equation has been called is also provided in Figure 7c, being Equation (14) for the repairings and Equation (11) for the *Risk Expansion*. In Figure 7d the execution time can be checked for the three cases. Besides, this data can be contrasted with the one related to the execution of the *Global Path Planning* process in the previous experiment: the total number of *Global Nodes* visited was 10522, the *eikonal* Equation (7) is used a total of 19929 times and the required time for the execution is 0.2 seconds. Comparing these values with the results from the second experiment, it can be observed how the order of magnitude of computing on *Global Layer* is around 10 times longer than on *Local Layer*. It is worth mentioning that *Global Path Planning* in the first experiment is used on a relatively small area with a few hundreds of square meters, while in other cases it could be extended to much larger surfaces, in terms of square kilometers, increasing the computational power needed. However, being the overall area larger does not change the computational power required by *Local Path Repairing*, since it just depends on the relation between Λ and λ and also on the location of the obstacles with respect to the path. This last statement is supported by the results of the second experiment: in the first repairing the path initially goes through the middle of the obstacle, while in the second repairing is not the case, being the computation cheaper. The third repairing is more expensive than the previous ones because of the amount of area occupied by obstacles, which are also placed too close to each other forming a wall. The increase of computation of the fourth repairing with respect to the third one is also justified by the increase in obstacle area as well as risky area. Finally, as can be noticed in all cases, the resulting path continues following the descending gradient of the potential field of *Total Cost* values computed on the *Global Layer* as much as possible, as expected from using the *sweeping approach*. This validates the *Local Path Repairing* as a method to repair the path while keeping track of the global computation done using lower computational resources.

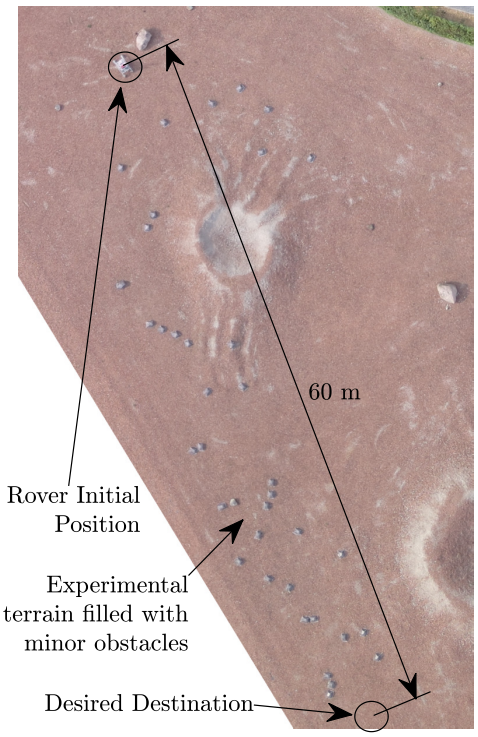


Figure 8: Orthonormal image of the experimental terrain where the field test is executed.

3.3. Field test

While *Local Path Repairing* under the *sweeping approach* has been tested in the previous simulation, it still remains the *conservative approach* to be validated. Thus, a field test is carried out in which this approach is chosen. The main purpose of this test is to replicate a planetary exploration case where a rover has to reach a certain position traversing an area filled with obstacles that were not previously considered. A rover prototype named HDPR (see Figure 4b) is used. Although this platform can make use of just one locomotion mode, *Normal Driving*, only the *Local Path Repairing* process is under the scope of this test.

Figure 8 depicts the setup prepared for this field test. During it, the rover must move from the initial position to the desired destination. On its way, the rover faces several obstacles that are not considered during the *Global Path Planning* process, as seen in Figure 9. These obstacles were placed randomly between both starting and goal positions. This is done to emulate how in the real exploration case the only information available prior to the traverse comes from orbital images. As consequence, minor elements such as rocks are not detected due

Figure 9: HDPR performing its traverse through the experimental field populated by obstacles.

to the resolution of these images. Therefore, the same DEM with the global resolution of 1 meter that was used for the simulations is also employed for this test. However, since no information related to the power consumption of the rover is taken into account, cost values proportional to the slope are used. The magnitude of these cost values on the *Global Layer* can be checked on Figure 10. The inverse of the speed of the rover, being in this case constant with a magnitude of (10 cm/s), is applied to those nodes placed on flat terrain, while the cost of *Global Nodes* placed on more pronounced slopes is penalized. Later on, the *Global Path Planning* process computes the optimal and continuous path to the goal according to this cost information.

Once the path is produced, it is time for the rover to start following it. During the traverse, the rover makes use of an on-board navigation system for the detection of obstacles based on a frontal camera. The functioning of this system is out of the scope of this paper, but thanks to it the rover is capable to map the obstacle on the *Local Layer*. The resolution λ chosen in this case is 10 centimeters. Then, the obstacle area is dilated and the values of *risk* are assigned to the respective *Local Nodes* via the *Risk Expansion* process. Later on, the *Local Path Repairing* process computes the *Local Waypoints* needed to reconnect to the original path while avoiding the obstacles. As stated before, the approach chosen in this case is the *conservative approach*. In Figure 10 are represented the resulting paths that were produced each time the rover found an obstacle on its way. A total of 13 reparations were computed in real time during this traverse. A video showing HDPR performing the test can be found in YouTube [4]. It can be con-

[4] `https://youtu.be/X4mihNTEVGw`

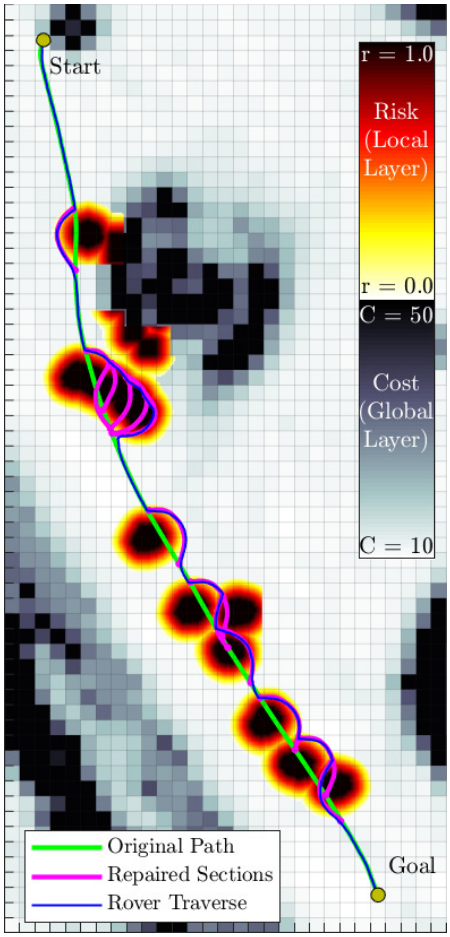


Figure 10: Paths computed during the field test.

cluded how this repairing process finds the path to avoid any obstacle, having to deviate further from the original path when the obstacles form a wall.

4. Conclusions

In this paper a novel path planning has been presented. It works at two scales, global and local, using the so called *Multi-layered Grid*. The grid is composed of two layers, each of them for different planning purposes. The path connecting the rover initial and goal locations is computed on the *Global Layer*, according to the information related to the terrain and the rover locomotion modes. Then, the generated global path plan is dynamically repaired using the *Local Layer* whenever an obstacle is detected by the rover during its traverse.

As regards the global path planner, it has been demonstrated, in a simulation environment, that the use of FMM provides an optimal, continuous and smooth path. Besides, the global path planner has been modified to take into consideration the use of a reconfigurable rover, with more than one locomotion mode. This improvement allows to find new optimal paths by the combination of these locomotion modes, reducing the total cost of performing the traverse from one location to another.

When new obstacles are detected in the rover planned path, it is repaired by means of two methods that have been tested in a simulation and a real environment. The first one, called *sweeping approach*, has demonstrated it generates the optimal repairing, avoiding sharp turning angles once the obstacle has been overcome. However, this method requires the computation of new *Global Waypoints* towards the goal. On the other hand, the *conservative approach*, which has been validated in a field test, repairs the path by only computing the *Local Waypoints*, trying to reconnect to the global path as soon as possible. This limitation causes the repaired path to be longer and less smooth during the connection to the global path. As an advantage, this approach avoids the use of the *Global Layer* each time the path is repaired. Therefore, the rover does not need to store and process the global layer, i.e. it only uses the initial global path.

As a future work, it is considered to extend this works by including the detection of different kind of terrains in the local path repairing stage. On the other hand, it is also intended to extend the capabilities of the *Global Path Planning* to take into consideration the direction of slopes in order to design an anisotropic path planning algorithm. These issues are proposed as future works.

Acknowledgements

This work was partially supported by the European Space Agency under contract number 4000118072/16/NL/LvH/gp and by the Spanish project DPI2015-65186-R.

References

Azkarate, M., Zwick, M., Hidalgo-Carrio, J., Nelen, R., Wiese, T., Poulakis, P., Joudrier, L., Visentin, G., 2015. First Experimental investigations on Wheel-Walking for improving Triple-Bogie rover locomotion performances, in: 13th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA), ESA-ESTEC. pp. 1–6.

Bajracharya, M., Maimone, M.W., Helmick, D., 2008. Autonomy for mars rovers: Past, present, and future. Computer 41, 44–50. doi:10.1109/MC.2008.479.

Brooks, C.A., Iagnemma, K., 2012. Self-supervised terrain classification for planetary surface exploration rovers. Journal of Field Robotics 29, 445–468. doi:10.1002/rob.21408.

Brunner, M., Brüggemann, B., Schulz, D., 2012. Motion planning for actively reconfigurable mobile robots in search and rescue scenarios, in: IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), IEEE. pp. 1–6. doi:10.1109/SSRR.2012.6523896.

Brunner, M., Brüggemann, B., Schulz, D., 2014. Metrics for path planning of reconfigurable robots in uneven terrain, in: Informatics in Control, Automation and Robotics. Springer, pp. 147–165. doi:10.1007/978-3-319-03500-0_10.

Carsten, J., Rankin, A., Ferguson, D., Stentz, A., 2007. Global Path Planning on Board the Mars Exploration Rovers, in: IEEE Aerospace Conference, IEEE. pp. 1–11. doi:10.1109/AERO.2007.352683.

Creager, C., Johnson, K., Plant, M., Moreland, S., Skonieczny, K., 2015. Push–pull locomotion for vehicle extrication. Journal of Terramechanics 57, 71–80. doi:10.1016/j.jterra.2014.12.001.

Cunningham, C., Nesnas, I.A., Whittaker, W.L., 2019. Improving slip prediction on mars using thermal inertia measurements. Autonomous Robots 43, 503–521. doi:10.1007/s10514-018-9796-4.

Fergason, R.L., Christensen, P.R., Kieffer, H.H., 2006. High-resolution thermal inertia derived from the Thermal Emission Imaging System (THEMIS): Thermal model and applications. Journal of Geophysical Research: Planets 111. doi:10.1029/2006JE002735.

Ferguson, D., Stentz, A., 2006a. Multi-resolution Field D*, in: Intelligent Autonomous Systems. IOS Press, pp. 65–74.

Ferguson, D., Stentz, A., 2006b. Using interpolation to improve path planning: The Field D* algorithm. Journal of Field Robotics 23, 79–101. doi:10.1002/rob.20109.

Filip, J., Azkarate, M., Visentin, G., 2013. Trajectory control for autonomous planetary rovers, in: 12th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA), ESA-ESTEC. pp. 1–7.

Hewitt, R.A., Boukas, E., Azkarate, M., Pagnamenta, M., Marshall, J.A., Gasteratos, A., Visentin, G., 2018. The Katwijk beach planetary rover dataset. The International Journal of Robotics Research 37, 3–12. doi:10.1177/0278364917737153.

Kimmel, R., Sethian, J.A., 2001. Optimal algorithm for shape from shading and path planning. Journal of Mathematical Imaging and Vision 14, 237–244. doi:10.1023/A:1011234012449.

Lester, D., Thronson, H., 2011. Human space exploration and human spaceflight: Latency and the cognitive scale of the universe. Space Policy 27, 89–93. doi:10.1016/j.spacepol.2011.02.002.

Lester, D.F., Hodges, K.V., Anderson, R.C., 2017. Exploration telepresence: A strategy for optimizing scientific research at remote space destinations. Science Robotics 2, 1–2. doi:10.1126/scirobotics.aan4383.

Liu, Y., Bucknall, R., 2015. Path planning algorithm for unmanned surface vehicle formations in a practical maritime environment. Ocean Engineering 97, 126–144. doi:10.1016/j.oceaneng.2015.01.008.

Maimone, M.W., Leger, P.C., Biesiadecki, J.J., 2007. Overview of the Mars Exploration Rovers Autonomous Mobility and Vision Capabilities, in: IEEE International Conference on Robotics and Automation (ICRA), pp. 1–8.

McEwen, A.S., Eliason, E.M., Bergstrom, J.W., Bridges, N.T., Hansen, C.J., Delamere, W.A., Grant, J.A., Gulick, V.C., Herkenhoff, K.E., Keszthelyi, L., et al., 2007. Mars reconnaissance orbiter's high resolution imaging science experiment (HiRISE). Journal of Geophysical Research: Planets 112. doi:10.1029/2005JE002605.

Miró, J.V., Dumonteil, G., Beck, C., Dissanayake, G., 2010. A kyno-dynamic metric to plan stable paths over uneven terrain, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE. pp. 294–299. doi:10.1109/IROS.2010.5650042.

Ono, M., Fuchs, T.J., Steffy, A., Maimone, M., Yen, J., 2015. Risk-aware planetary rover operation: Autonomous terrain classification and path planning, in: IEEE Aerospace Conference, IEEE. pp. 1–10. doi:10.1109/AERO.2015.7119022.

Patel, N., Slade, R., Clemmet, J., 2010. The exomars rover locomotion subsystem. Journal of Terramechanics 47, 227–242. doi:10.1016/j.jterra.2010.02.004.

Petres, C., Pailhas, Y., Petillot, Y., Lane, D., 2005. Underwater path planing using fast marching algorithms, in: Europe Oceans 2005, IEEE. pp. 814–819. doi:10.1109/OCEANSE.2005.1513161.

Philippsen, R., Kolski, S., Macek, K., Jensen, B., 2008. Mobile Robot Planning in Dynamic Environments and on Growable Costmaps, in: IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 1–8.

Pérez-del Pulgar, C.J., Sánchez, J., Sánchez, A., Azkarate, M., Visentin, G., 2017. Path planning for reconfigurable rovers in planetary exploration, in: IEEE International Conference on Advanced Intelligent Mechatronics (AIM), IEEE. pp. 1453–1458. doi:10.1109/AIM.2017.8014223.

Rothrock, B., Kennedy, R., Cunningham, C., Papon, J., Heverly, M., Ono, M., 2016. SPOC: Deep learning-based terrain classification for mars rover missions, in: AIAA SPACE 2016, pp. 1–12. doi:10.2514/6.2016-5539.

Sethian, J.A., 1999. Fast Marching Methods. SIAM review 41, 199–235. doi:10.1137/S0036144598347059.

Vago, J., Witasse, O., Svedhem, H., Baglioni, P., Haldemann, A., Gianfiglio, G., Blancquaert, T., McCoy, D., de Groot, R., 2015. ESA ExoMars program: the next step in exploring Mars. Solar System Research 49, 518–528. doi:10.1134/S0038094615070199.

Vago, J.L., Westall, F., Coates, A.J., Jaumann, R., Korablev, O., Ciarletti, V., Mitrofanov, I., Josset, J.L., De Sanctis, M.C., Bibring, J.P., et al., 2017. Habitability on early Mars and the search for biosignatures with the ExoMars rover. Astrobiology 17, 471–510. doi:10.1089/ast.2016.1533.

Valero-Gomez, A., Gomez, J.V., Garrido, S., Moreno, L., 2013. The path to efficiency: Fast marching method for safer, more efficient mobile robot trajectories. IEEE Robotics & Automation Magazine 20, 111–120. doi:10.1109/MRA.2013.2248309.

Woods, M., Shaw, A., Barnes, D., Price, D., Long, D., Pullan, D., 2009. Autonomous science for an ExoMars Rover–like mission. Journal of Field Robotics 26, 358–390. doi:10.1002/rob.20289.