3Dana: Path Planning on 3D surfaces

Pablo Muñoz, María D. R-Moreno and Bonifacio Castaño

Abstract An important issue when planning the tasks that a mobile robot has to reach is the path that it has to follow. In that sense, classical path planning algorithms focus on minimizing the total distance, generally assuming a flat terrain. Newer approaches also include traversability cost maps to define the terrain characteristics. However, this approach may generate unsafe paths in realistic environments as the terrain relief is lost in the discretisation. In this paper we will focus on the path planning problem when dealing with a Digital Terrain Model (DTM). Over such DTM we have developed 3Dana, an any-angle path planning algorithm. The objective is to obtain candidate paths that may be longer than the ones obtained with classical algorithms, but safer. Also, in 3Dana we can consider other parameters to maximize the path optimality: the maximum slope allowed by the robot and the heading changes during the path. These constraints allow discarding infeasible paths, while minimizing the heading changes of the robot. To demonstrate the effectiveness of the algorithm proposed, we present the results for the paths obtained for real Mars DTMs.

1 Introduction

Classical path planning algorithms used by on-ground operators are based on (i) flat terrains with free or blocked cells or (ii) terrain models that usually are produced merging different terrain characteristics (slope, rocks, quicksands, etc.) into a single layer: a traversability cost map. These maps provide the estimated effort required to cross an area of the map with a unique number. The merging process simplifies the path planning algorithm, but at the expense of lost specific information that can be

Pablo Muñoz and María D. R-Moreno
Departamento de Automática, Universidad de Alcalá, e-mail: {pmunoz,mdolores}@aut.uah.es

Bonifacio Castaño
Departamento de Matemáticas y Físicas, Universidad de Alcaá, e-mail: bonifacio.castano@uah.es

useful during the path search. For instance, analysing slopes, rocks or other terrain characteristic independently will allow the path planning algorithm to obtain safer paths, avoiding certain slopes, rocks concentration or other constraints, rather than only minimize the path cost as done till now.

In this paper we present a new path planning algorithm called 3D Accurate Navigation Algorithm (3Dana). It is based on heuristic path planning algorithms such as A* [1], Theta* [2] and S-Theta* [3]. 3Dana is developed having in mind its potential application to exploration robots (e.g. rovers) operating in planetary surfaces. Then, constraints such as the relief of the terrain or the heading changes for a given path are relevant parameters to be considered. 3Dana exploits a Digital Terrain Model (DTM) that provides an accurate abstraction of the terrain relief. Then, the algorithm can avoid paths that overcomes a maximum slope defined by the human operator. It also considers the heading of the robot during the path search, generating smoother routes. Using 3Dana, human operators can select the best performance path considering different parameters, while keeping the safety of the robot.

The next section provides the description of the path planning problem and the notation used for this paper. Sec. 3 presents a brief revision of path planning algorithms. Following, sec. 4 defines the DTM employed to model the surface used by our algorithm. Then, the 3Dana path planning algorithm is presented. Sec. 6 shows an empirical evaluation of the algorithm on real Mars maps. Finally, conclusions and future work are outlined.

2 Path planning notation and representation

The most common discretisation of the environment is a 2D representation formed by a uniform regular grid with blocked and unblocked cells [4]. The edges of the cell represents the nodes over which the robot traverses, being constant the distance between adjacent nodes (except for diagonal moves in an 8-connected grid). Then, each node represents a position with coordinates (x_i, y_j) in the map. In the following, we represent nodes using a lower-case letter and for the sake of simplicity we will symbolize any node p as a coordinate pair (x_p, y_p). Also, we assume that each node can have an elevation value z_p, so we obtain a DTM discretised as a grid. To define the path planning problem, we assume that s and g are the start and goal nodes of respectively. Then, a candidate path will be a set $(p_1, p_2, ..., p_{n-1}, p_n)$ with initial node $p_1 = s$ and goal $p_n = g$. A path is valid iff it does not cross a blocekd cell. Besides their geometrical values, each node has four attributes required by heuristic search algorithms:

- $p.G$: the cumulative cost to reach the node p from the initial node.
- $p.H$: the heuristic value, i.e, the estimated distance to the goal node.
- $p.F$: the node evaluation function: $p.F = p.G + p.H$.
- $p.parent$: the reference to the parent node. The parent of a node p must be a reachable node from the p node.

Finally, some algorithms can deal with traversability cost maps. A cost map is an extension of the introduced grid in which each cell has an associated cost. This cost is used to represent a terrain characteristic and, sometimes, the cost is generated combining different parameters, e.g. rocky and hazardous areas, slopes, etc. During the path search, the cell cost is used as a multiplicative factor applied to the length of the path that traverses over such cell. Using cost maps, some algorithms are able to avoid potentially hazardous areas. However, exploiting a combination of parameters into a single value, could lead to miss information that can be useful separately.

3 Related works

The objective of path planning is to obtain paths between different points in an environment that can be partially or completely known. There are several variations to solve this problem such as Rapidly exploring Random Trees (RRT) [5] or Probabilistic Road Maps (PRM) [6]. However, we will focus on path planning methods based on heuristic search algorithms as 3Dana inherits from these algorithms.

The most representative heuristic search algorithm is A* [1]. A* has been so widely used because is simple to implement, very efficient and has lots of scope for optimization [7]. But it has an important limitation: it is based on a graph search. Typically, an eight-connected graph is used, which implies a restriction in the path headings to multiples of $\pi/4$. Thus, A* generates a sub-optimal paths with zig-zag patterns. For the heuristic computation A* uses the *Octile* distance [2].

In order to avoid the zig-zag patterns a new family of path planning algorithms called *any-angle* has appeared. These algorithms are based on A* as well, and called *any-angle* since the paths generated are not restricted to to multiples of $\pi/4$. The most representative is Theta* [2]. During the search, for a given position p, Theta* evaluates the line of sight between the successors of p and the parent of the current node, $q = p.parent$, i.e., if the straight line between p and q cross or do not cross blocked cells. Given a node $t \in successors(p)$, if there is line of sight between t and q, the parent of t will be q instead of p (as happens in A*). If there are obstacles between q and t, Theta* behaves as A*. This allows removing the intermediate node (p), smoothing the path during the search process, while maintain a free-obstacle path. As the parent of a node is no longer restricted to be an adjacent node, the path generated can have any heading change, i.e., the A* heading restriction of $\pi/4$ is overcome. However, the line of sight checking is performed frequently, which has a significant computational overhead. In any case, the paths generated are never longer than the obtained with previous approaches. As there is no angle restriction, Theta* obtains better results using the *Euclidean* distance (the straight line distance) as the heuristic function as opposed to the Octile one. Also, as a consequence of the expansion process, Theta* only performs heading changes at the corners of the blocked cells.

A recent algorithm developed from Theta* is Smooth Theta* (S-Theta*) [3]. It aims to reduce the amount of heading changes that the robot should perform to reach the goal. To do this, S-Theta* includes a new term, $\alpha(t)$. This value gives a measure of the deviation from the optimal trajectory to achieve the goal, considering the heading of the robot at each step during the path search. In an environment without obstacles, the optimal path between two points is the straight line. Therefore, applying the triangle inequality, any node that does not belong to that line will involve both, a change in the direction and a longer distance. Therefore, $\alpha(t)$ causes that nodes far away from that line will not be expanded during the search. S-Theta* reduces the number and amplitude of heading changes with respect to Theta*, but at the expense of slightly longer path lengths. Besides, S-Theta* can produce heading changes at any point, not only at the vertex of the obstacles as Theta* does.

Above algorithms try to minimize the total distance and the heading changes for a path in uniform flat environments. Although it is possible to exploit them in robotics, when applying to natural scenarios, e.g., planetary exploration robots, they do not provide enough safe routes. Path generated can cross rocky or high-hills areas that the robot will never be able to reach. Algorithms such as the D* family (the most representative one is Field D* [8]) works with cost maps aiming to provide safer paths in realistic environments. The objective of these algorithms is not to minimize the distance travelled but to minimize the path cost. In particular, the NASA rover drivers employ a variation of Field D* to plan the operations of the MER and MSL.

In a similar direction, Garcia et al. [9] presented a path planning architecture that uses a cost map that combines the elevation, slope and roughness of the terrain. These parameters are acquired using a laser scanner and processed using a fuzzy engine to generate the cell costs. Then, a path planning module generates the path based on a modified version of A*. However, this approach inherits quite unrealistic paths generation since they exploit A*.

Another possibility is to employ a DTM, as Ishigami et al. [10] do. They present a technique to evaluate the motion profiles of a rover when it has to follow the shortest path generated by the Dijkstra algorithm [11]. Based on a DTM, they are able to evaluate the path considering the rover dynamics and the wheel-soil contact model. This allows generating really safe paths, but the process is very time consuming. The algorithm also imposes a constraint that previous approaches do not have: it requires a specific model of the robot.

Following the DTM representation, Page et al. [12] exploit a 3D triangle mesh for the terrain discretisation. Using such DTM, their algorithm generates a path following either the valleys or ridges of the terrain depending on the criteria selected by the user. Although the idea is quite novel, it is not clear how the algorithm performs in real scenarios.

4 Terrain interpolation

We have presented a terrain discretisation based on a 2D representation that can be easily enhanced to a 3D terrain or DTM considering the elevation at each node. If we have a DTM, the ground representation is defined as a set of $k = n \times m$ spatial points

$(x_i, y_j, z_{i,j})$, where $1 \leq i \leq n$ and $1 \leq j \leq m$. The value $z_{i,j}$ represents the height of the terrain over the node located at the position (x_i, y_j). We assume a regular grid, i.e, the distance between two nodes (x_i, y_j) and (x_{i+1}, y_j), $1 \leq i \leq n-1$; $1 \leq j \leq m$ is constant. As well, we have the same distance for two nodes (x_i, y_j) and (x_i, y_{j+1}), $1 \leq i \leq n$; $1 \leq j \leq m-1$. A possible graphical representation can be seen in fig. 1.

Given a DTM, the elevation of each node is known, so, algorithms restricted to move between nodes (e.g. A*) can be used without problems. If we want to traverse from $(x_i, y_j, z_{i,j})$ to $(x_{i+1}, y_j, z_{i+1,j})$, we can obtain the path length by applying the Pythagoras theorem. Instead, *any-angle* algorithms can traverse among non-adjacent nodes. Thus, it is required to compute the elevation of points that do not belong to the rectangular grid. For example, a movement between nodes $(x_i, y_j, z_{i,j})$ and $(x_{i+\gamma}, y_{j+\delta}, z_{i+\gamma,j+\delta})$ for arbitrary $\gamma > 1, \delta > 1$ and $\gamma \neq \delta$, implies to cross more than one cell and traverse in between nodes. For such coordinates we do no have the elevation: we need to interpolate the elevation for these points to obtain the most approximate distance travelled by the robot if we want to exploit an *any-angle* algorithm.

To get the elevation of a node (x_u, y_u) that not belongs to the rectangular grid (x_i, y_j), $1 \leq i \leq n, i \in \mathbb{N}$; $1 \leq j \leq m, j \in \mathbb{N}$, we will employ a lineal interpolation when $x_u = x_i$ for some value i or $y_u = y_j$ for some j. First, we need to consider the shape belonging to the cell formed by the nodes $(x_i, y_j, z_{i,j})$, $(x_{i+1}, y_j, z_{i+1,j})$, $(x_i, y_{j+1}, z_{i,j+1})$ and $(x_{i+1}, y_{j+1}, z_{i+1,j+1})$. Usually, four points in the space are not guaranteed to fit to a plane. As we see in other approaches, it is possible to model the DTM as a triangular mesh. In this regard, we assume that each cell is composed of four triangles whose base is the connection between two nodes, while the cathetus are the joint between each node and the central point of the cell, (x_c, y_c, z_c), as shown in fig. 2. The altitude at the central point is computed as the mean value of the four nodes that comprise the cell $(x_i, y_j, z_{i,j})$, $(x_{i+1}, y_j, z_{i+1,j})$, $(x_i, y_{j+1}, z_{i,j+1})$ and $(x_{i+1}, y_{j+1}, z_{i+1,j+1})$. Calculating the distance travelled using this terrain representation is computationally expensive (as explained next), but unambiguous.

With this in mind, we need to compute the length for any arbitrary pair of nodes. This implies:

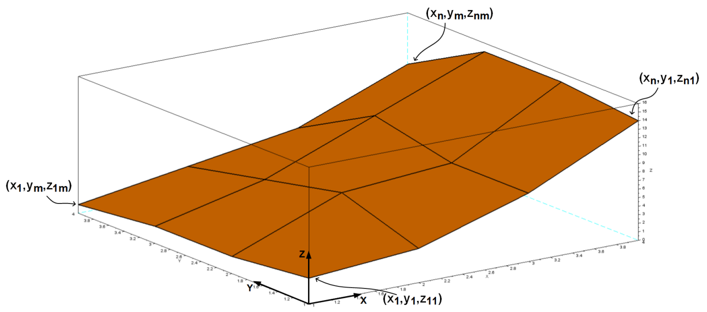

Fig. 1 DTM representation.

1. Obtaining the list of crossed cells.
2. For each cell, we need to calculate:
   (a) The coordinates in which the path enters and exits the cell.
   (b) The coordinates in which the path intersects with the segments that conforms the four triangles.
3. Obtaining the elevation for all the previous coordinates. If the coordinate is not a node, we need to interpolate its altitude.

We can determine the cells crossed by a straight line using a Cohen-Sutherland clipping algorithm [13]. As well, such algorithm can be used to compute the entry and exit point to the cell. Then, we have to calculate the intersection points between the path and the four triangles that conform the cell. In this point there are different possibilities depending on the entry and the exit points.

Assuming that the movement does not start and end in a node, we need to interpolate the altitude at different points. First, consider a point (x_u, y_u, z_u) that belongs to the x axis as show in fig. 2. Its first coordinate is $x_u = x_i$ for some $1 \leq i \leq n$, then (x_u, y_u) will belong to the straight line between the two grid points (x_i, y_j) and (x_i, y_{j+1}). Next, we can linearly interpolate (using the triangle similarity) the altitude z_u for the point (x_u, y_u) as in eq. 1. This allows us to interpolate the elevation of any point between two adjacent nodes with the same x coordinate. For the case of interpolating the elevation between two adjacent nodes with the same y coordinate the procedure is equivalent and not shown here.

$$z_u = z_{i,j} + \frac{z_{i,j+1} - z_{i,j}}{y_{j+1} - y_j}(y_u - y_j) \tag{1}$$

So, let (x_u, y_u, z_u) be the entry point to the cell and let (x_v, y_v, z_v) be the exit point that are now known. Considering that (x_u, y_u, z_u) belongs to the line $\overline{(x_i, y_j), (x_i, y_{j+1})}$ we can exit through one of the other three sides of the cell. Depending on the exit point, it is possible that we need to cross two or three planes of the cell. The possibilities are the following:

(a) Exit at the side defined by the line $\overline{(x_i, y_j), (x_{i+1}, y_j)}$. Two planes are crossed.
(b) Exit at the opposite side of a point that belongs to $\overline{(x_{i+1}, y_j), (x_{i+1}, y_{j+1})}$ constrained to $y_v < z_c$. Then three planes are crossed, and the one not crossed is the one formed by the nodes $(x_i, y_{j+1}), (x_c, y_c), (x_{i+1}, y_{j+1})$. This is the example presented in fig. 2.
(c) Exit at the opposite side of a point belong to $\overline{(x_{i+1}, y_j), (x_{i+1}, y_{j+1})}$ constrained to $y_v \geq z_c$. Then three planes are crossed, and the one not crossed is formed by the nodes $(x_i, y_j), (x_c, y_c), (x_{i+1}, y_j)$.
(d) Exit at the side defined by the line $\overline{(x_i, y_{j+1}), (x_{i+1}, y_{j+1})}$. Two planes are crossed.

We can also consider entering the cell at a point within $\overline{(x_i, y_j)}$, $\overline{(x_{i+1}, y_j)}$. This case is equivalent to the previously presented and can be solved using symmetry. Next, we briefly present how to compute the traversed length in a cell for case (b).

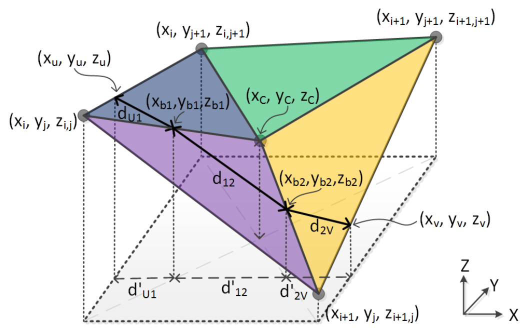

Fig. 2 Representation of a DTM cell.

Cases (a) and (d) are easier to compute and case (c) is symmetric to (b). We do not provide the mathematical model as it is excessively large to be presented here.

We need to compute the points in which the straight line that the path follows, $\overline{(x_u, y_u, z_u), (x_v, y_v, z_v)}$, changes from one plane to another in the cell. For (b) case three planes are cut, so we need to compute two points. Let (x_{b1}, y_{b1}, z_{b1}) and (x_{b2}, y_{b2}, z_{b2}) be these points as fig. 2 shows. To obtain these points we need the coordinates (x, y) and then interpolate the elevation at such point. The way to obtain the intersect points is to employ the equation of the line and obtain the points in which the lines $\overline{(x_u, y_u), (x_v, y_v)}$ and $\overline{(x_i, y_j), (x_{i+1}, y_{j+1})}$ and/or $\overline{(x_i, y_{j+1}), (x_{i+1}, y_j)}$ (diagonals) intersect.

Once we have (x_{b1}, y_{b1}, z_{b1}) and (x_{b2}, y_{b2}, z_{b2}) we can obtain their altitude using triangle similarity. We can use eq. 1 to do this, using the correct triangles to interpolate the altitude. If we want to obtain the elevation of the point (x_{b1}, y_{b1}, z_{b1}), we need to evaluate the triangle formed by such point, the node with coordinates $(x_i, y_j, z_{i,j})$ and the center of the cell, (x_c, y_c, z_c) Finally, we can compute the distance travelled through the cell using the Pythagoras Theorem for each path segment.

Using this representation we can also obtain the slope of the terrain. To compute the slope, we need to obtain the normal vector of each plane that comprises the cell. Considering the point (x_u, y_u, z_u) that belongs to the plane formed by the points $(x_i, y_j, z_{i,j})$, (x_c, y_c, z_c) and $(x_{i+1}, y_j, z_{i+1,j})$, we can obtain the normal vector, $\overrightarrow{n_\pi}$, of the plane as in eq. 2. This normal vector forms an angle, α_z, with the Z axis that gives us the slope of the terrain. Thus, the slope for such plane can be obtained as in eq. 3. For each cell, we have four α_z, i.e., we have four slopes. Computing the slope during the path search enables the algorithm to avoid dangerous path.

$$\begin{aligned}
&\overrightarrow{n_\pi} = (A, B, C) \;\; with: \\
&A = (y_c - y_j) \cdot (z_c - z_{i+1,j}) - (z_c - z_{i,j}) \cdot (y_c - y_j) \\
&B = (z_c - z_{i,j}) \cdot (x_c - x_{i+1}) - (z_c - z_{i+1,j}) \cdot (x_c - x_i) \\
&C = (x_c - x_i) \cdot (y_c - y_j) - (y_c - y_j) \cdot (x_c - x_{i+1})
\end{aligned} \tag{2}$$

$$\alpha_z = \arccos \frac{C}{\sqrt{A^2 + B^2 + C^2}} \tag{3}$$

5 3Dana algorithm

The 3D Accurate Navigation Algorithm, abbreviated as 3Dana, is a path planning algorithm developed to obtain safer routes based on heuristic search over a DTM. 3Dana is an evolution of the A* search algorithm, and it takes advantage of the newest *any-angle* path planning algorithms such as Theta* or S-Theta*. Its application scope is those mobile robots in which the elevation of the surface can affect their mobility. The main features of 3Dana are:

- Evaluation of the path cost using the terrain altitude. 3Dana performs path planning over realistic surface models, using the DTM as explained in the previous section. The length of a movement is a function of the distance between two points given their altitudes.
- Evaluation of heading changes during the search. Just like the S-Theta* algorithm [3], 3Dana calculates the necessary turns needed to reach the next position taking into consideration the current heading of the robot and the position of the goal. This allows obtaining smoother routes.
- Evaluation of the terrain slope. 3Dana avoids paths that exceed the maximum slope allowed by the robot. This allows obtaining safer and feasible paths.

During the path search 3Dana maintains a list of reachable nodes, i.e., the *open* list. Such list is ordered by the F value of the nodes. If that list is empty, all the reachable nodes from the start position have been evaluated and none is the goal position. Then, there is no feasible path between the desired points. Otherwise, the first node from the *open* list, that is, the most promising one, is extracted. If that node is the objective, the algorithm returns the path between the start and the goal through backtracking of the parents nodes from the goal to the start. Otherwise, a successor function returns a set with the visible adjacent nodes for the current node.

Instead of the previous algorithms, 3Dana uses node re-expansion, which means that all nodes will be analysed instead if they are previously expanded. This may leads to better paths, but increasing the runtime due to the possibility of expanding the same node several times. This is required when dealing with elevation maps. For example, the algorithm can reach a node by climbing a hill, that can be more expensive that surrounding it. This usually implies to expand more nodes, and thus, that path is discovered later during node re-expansion.

For the heuristic function, 3Dana uses a variation of the *Euclidean* distance to take into consideration the altitude difference between any two nodes p and t. We call this heuristic *EuclideanZ*, and is computed as in eq. 4. The objective of this heuristic is to prioritize nodes without (or with lower) elevation changes. The *EuclideanZ* heuristic in 3D scenarios is admissible and consistent.

$$EuclideanZ(p,t) = \sqrt{(x_t - x_p)^2 + (y_t - y_p)^2 + (z_t - z_p)^2} \tag{4}$$

Besides the heuristic described, 3Dana also uses the value provided by the $\alpha(t)$ function, inherited from S-Theta*. This value measures the heading changes necessary to reach the next node as function of the current node's parent and the goal position. Alpha is computed as in eq. 5. Using $\alpha(t)$ in the node's heuristic allows us to consider the heading changes during the search process, delaying the expansion of nodes that require a high turn to be reached. The function gives a value in the interval [0°, 180°]. If we apply a weight factor to this heuristic we can determine the relevance of the heading changes in our path. Small weights implies a soft restriction, whereas larger weights tend to make the algorithm trends to follow smoother paths with low number of heading changes, in spite of the distance travelled. Generally the weight used, called α_w, takes values in the range 0 (heading changes are not considered during search) and 1. Experimentally, we have realised that values higher than 1 usually do not reduce the heading changes.

$$\alpha(p,t,g) = \arccos \frac{\text{dist(p, t)}^2 + \text{dist(p, g)}^2 - \text{dist(t, g)}^2}{2 \cdot \text{dist(p, t)} \cdot \text{dist(p, g)}} \cdot \alpha_w \tag{5}$$
$$\text{with dist(p, t)} = \sqrt{(x_t - x_p)^2 + (y_t - y_p)^2}$$

As 3Dana is an *any-angle* algorithm, we need to perform the line of sight checking during the search. When evaluating the line of sight, we perform the path length computation and the slope analysis. This procedure is divided in two phases: first, we compute the points in which the line that connects the two nodes intersects with the horizontal or vertical axis; and second, we compute the length of each segment formed by two consecutive points.

For the first step, we have implemented an algorithm to compute the axes intersection points. For two given nodes, p_0 and p_n, the algorithm returns a list of points that intersect the axes and belong to the line that connects p_0 and p_n.

Once the points list has been calculated, the second step is to treat the segments formed by each pair of consecutive points obtained in the previous phase. So, to compute the length traversed for two non-adjacent nodes (e.g., p_0 and p_n) and check if there is a line of sight between them, we need to evaluate the intermediate positions. For all pairs of consecutive points of the list, p_i and p_{i+1}, we need to perform the process described below:

1. Evaluate if the cell that contains the segment $\overline{p_i p_{i+1}}$ is an obstacle. If it is, the algorithm returns that the path between p_0 and p_n is blocked.
2. If we have defined a maximum slope allowed, we compute the terrain slope of the cell using the normal vector (as in eqs. 2 and 3). If the slope is higher than the maximum defined, such path is not considered.
3. Compute the length of the segment $\overline{p_i p_{i+1}}$ using the DTM data. For this, we use the formulation shown in sec. 4. The path length consider the elevation change using the four planes that describes the cell crossed by the segment under consideration.

When considering a maximum slope during the path search, 3Dana can generate safer routes. Consider the paths presented in fig. 3, both paths are obtained using 3Dana over the same DTM, but varying the maximum slope allowed. The left path does not consider slope limitations so then, it is highly undesirable since it crosses a

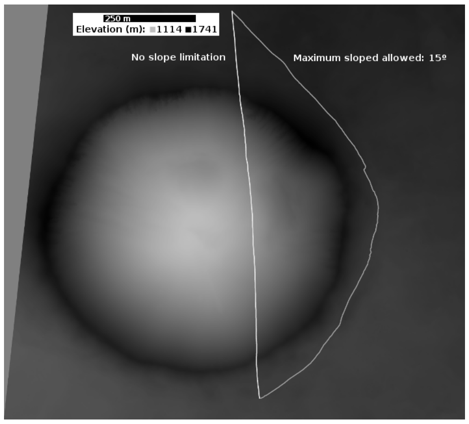

Fig. 3 No slope limited path (center path) vs limited slope path (right path) over a DTM.

crater. Meanwhile, when the maximum slope is setted to 15°, the algorithm avoids the crater and surrounds it.

The implementation of 3Dana allows safely generating candidate paths for mobile robots. Providing a DTM of the environment, we can select the best paths based not only in the distance travelled (considering also the elevation changes), but also reducing the heading changes and avoiding terrains with excessive slopes.

6 Experimental results

In this section, we test the behaviour of the 3Dana algorithm using high resolution DTMs from Mars. The elevation data is obtained by the Mars Reconnaissance Observer (MRO) spacecraft, which provides elevation data with a vertical accuracy of 25 centimetres [14][1]. The maps used here have a resolution of 2 meters, i.e., we have a uniform grid with elevation points every two meters. The objective of the experiments is to generate different paths when varying the constraints, i.e., we evaluate paths with different heading changes consideration (modifying the α_w factor) and various maximum slopes allowed. In order to provide a comparative, we have adjusted the A* algorithm to work with the DTMs following the same model presented in sec. 4 and including the altitude difference in the *Octile* heuristic.

For the experiments we have taken into consideration the following parameters: (1) the path length, (2) the total number of accumulated degrees by the heading changes (total turn), (3) the CPU time or search run-time, and, (4) the number of expanded nodes during the search. The execution is done on a 2.5 GHz Intel Core i7 with 8 GB of RAM under Ubuntu 14.04.

[1] The DTMs are publicly accessible in http://uahirise.org/dtm

The first map considered presents a central structure and layered bedrock in a 25-kilometer diameter crater[2]. The total area covered is near 40 km^2. The dimension in nodes is 3270x6636. In this map we set the initial point to the coordinates (700,500) and the goal in (2800,6000). Then, we have run A* and 3Dana with different configurations. These configuration entails no heading changes consideration ($\alpha_w = 0$), and two more values $\alpha_w = 0.5$ and $\alpha_w = 1$. Also, we have considered three possible values for the maximum slope allowed in the path: no limitation, 10° and 20°. The results obtained for the different paths are presented in table 1, while some paths are represented in fig. 4.

Given the data, we can observe that the path length and the heading changes obtained by 3Dana are better than the values obtained by A* when we do not consider maximum slope neither heading changes. As we consider maximum slopes for the

Table 1 Data for DTEED_017147_1535. In bold: best path length + total turns for each slope.

Alg.	Max. slope	α_w	Length (m)	Turn (°)	Time (sec.)	Expanded nodes
A*	-	-	11010	31770	1120	2497903
3Dana	-	-	**10189**	**751**	776	2503970
A*		-	11427	39915	1023	3032953
	20°	0.0	10822	4918	1842	4087662
3Dana		0.5	**10887**	**3742**	4112	3870873
		1.0	11022	3720	5670	3990782
A*		-	15434	73260	833	5213506
	10°	0.0	14703	10815	1617	5514326
3Dana		0.5	**14979**	**10355**	2485	5384046
		1.0	15193	10923	3147	5391994

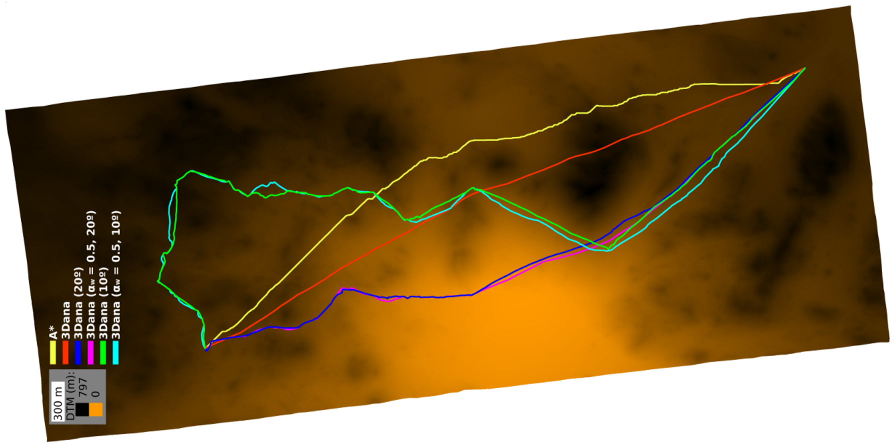

Fig. 4 Paths obtained for the DTEEC_017147_1535 using A* and different configurations of 3Dana.

[2] http://www.uahirise.org/dtm/dtm.php?ID=ESP_017147_1535

path, we can see that as higher is the restriction imposed (i.e., smaller sloped allowed), both, the path length and the total turns increase. This is specially notorious when we restrict the slope to 10°, in which the path length is 1.35 times longer that the path restricted to 20° (without considering heading changes). Regarding to the heading changes, we can appreciate that $\alpha_w = 0.5$ effectively reduces the total turn parameter. However, $\alpha_w = 1.0$ increases the turns respect to its previous configuration. While this parameter works fine in flat environments (see the S-Theta* evaluation [3]), considering the elevation seems to affect negatively. Particularly, attempting to avoid heading changes can lead to follow longer paths, as we can discard preferable paths (as function of the slope) in spite of reducing the turns. Also, we can observe that the runtime increases with higher α_w values. 3Dana requires some time to find paths as a consequence of both, the re-expansion process of the nodes and the computational cost related to the management of the DTM.

The second map considered presents an uplift of a 30-kilometer diameter crater in Noachis Terra[3]. This map coves near 15 km^2. The dimension in nodes is 2960x2561. In this map we set the initial point to the coordinates (800,600) and the goal in (1800,2500). We have run the same experiments, i.e., $\alpha_w \in 0, 0.5, 1.0$ and maximum slopes 10° and 20°. However, in this map, all paths stay above of the 10° slope limitation. Thus, we have evaluated paths with 15° of maximum slope. Table 2 provides the results of these executions and fig. 5 the paths representation.

As for the first case, 3Dana outperforms A* in both path length and total turns when not considering maximum slope and heading changes. As well, we can appreciate that the path length increases from the case with maximum slope of 20° to the case with 15°. If we analyse the heading changes, in this map considering $\alpha_w = 0.5$ slightly decreases the total turns only when the maximum slope is set to 20°. However, in any case, using $\alpha_w = 1.0$ not only increases the path length, but also the total turns. Then, seems that α_w is not well suitable for complex maps.

In fig. 5 (right) we can appreciate the nodes expanded during the search for a maximum slope of 10°. This case is remarkable because 3Dana (as well as A*) is

Table 2 Data for DTEED_030808_1535. In bold: best path length + total turns for each slope.

Alg.	Max. slope	α_w	Length (m)	Turn (°)	Time (sec.)	Expanded nodes
A*	-	-	4800	13995	49	407544
3Dana	-	-	**4493**	**478**	77	720744
A*		-	5990	28080	343	2129666
	20°	0.0	5665	4442	433	2230932
3Dana		0.5	**5776**	**4241**	777	2150387
		1.0	5786	4428	983	2153923
A*		-	7387	41850	244	2712912
	15°	0.0	**7010**	**8569**	409	2789358
3Dana		0.5	7175	9066	749	2721862
		1.0	7450	9061	901	2774165

[3] http://www.uahirise.org/dtm/dtm.php?ID=ESP_030808_1535

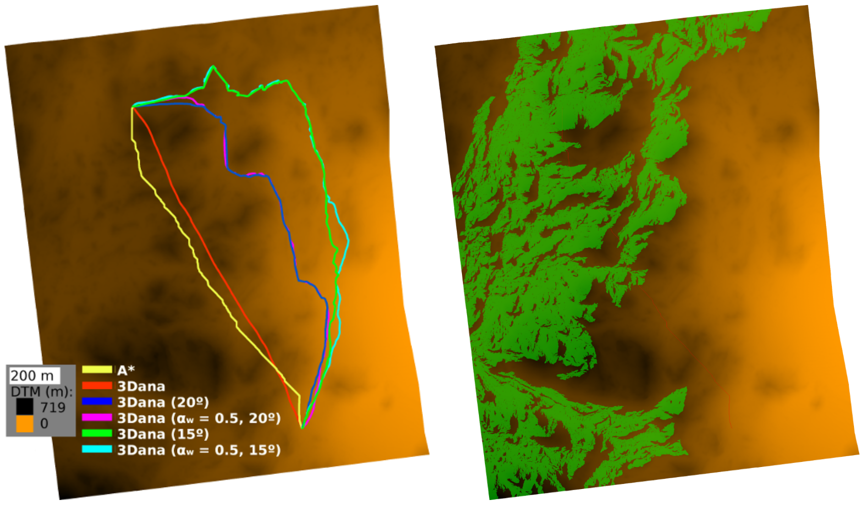

Fig. 5 Paths obtained for the DTEED_030808_1535 using A* and different configurations of 3Dana (left). Area of the map explored (in green) by 3Dana considering a maximum slope of $10^{c}irc$ (no path found in this case).

not able to find a path with sucha constraint. We can see that the algorithm expands several nodes, but there is no path that allows safely reaching the desired goal. Then, fig. 5 (right) provides a vision of the reachable areas of the map when we restrict the maximum slope to 10°. This could provide an insight of the terrain that can be useful for human operators during the mission planning.

7 Conclusions

Heuristic search path planning algorithms such as A* or S-Theta* try to minimize the total distance that the robot should travel. Although this criteria has been widely used to compare algorithms, it is not enough if the robot cannot cross certain rocky or cumbersome areas. In this paper we presented a new *any-angle* algorithm named 3Dana. It is designed with the purpose of considering the terrain model and minimizing the heading changes of a path. 3Dana integrates a DTM during the search, which enables to avoid potentially dangerous areas, and then generate safer routes. This is done discarding paths that exceed slopes restrictions imposed by the user. Moreover, 3Dana computes the necessary turns for a path, providing smoother routes.

Experiments performed with Mars DTMs show that 3Dana can generate routes restricted by the slope, avoiding dangerous terrains. Also, some configurations allows properly reducing the heading changes performed during the path. As a future work, we will work to improve the capabilities of the algorithm, e.g., exploiting cost maps as well as the DTM during the path search.

Acknowledgements Pablo Muñoz is supported by the European Space Agency under the Networking and Partnering Initiative *"Cooperative Systems for Autonomous Exploration Missions"* project 4000106544/ 12/NL/PA. The work is supported by MINECO project EphemeCH TIN2014-56494-C4-4-P and UAH 2015/00297/001. Authors want to thanks the reviewers for their valuable comments.

References

1. P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," in *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, July 1968, pp. 100–107.
2. K. Daniel, A. Nash, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," *Journal of Artificial Intelligence Research*, vol. 39, pp. 533–579, 2010.
3. P. Muñoz and M. D. R-Moreno, "S-Theta*: low steering path-planning algorithm," in *procs. of the 32nd SGAI International Conference*, Cambridge, UK, December 2012, pp. 109–121.
4. P. Yap, "Grid-based path-finding," in *Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, vol. 2338. Springer Berlin / Heidelberg, 2002, pp. 44–55.
5. S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.
6. M. Akinc, K. E. Bekris, B. Y. Chen, A. M. Ladd, E. Plaku, and L. E. Kavraki, *Robotics Research. The Eleventh International Symposium*, ser. Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg, 2005, vol. 15, ch. Probabilistic Roadmaps of Trees for Parallel Computation of Multiple Query Roadmaps, pp. 80–89.
7. I. Millington and J. Funge, *Artificial Intelligence for Games*, 2nd ed. Morgan Kaufmann Publishers, 2009.
8. D. Ferguson and A. Stentz, "Field D*: An interpolation-based path planner and replanner," *Robotics Research*, vol. 28, pp. 239–253, 2007.
9. A. Garcia, A. Barrientos, A. Medina, P. Colmenarejo, L. Mollinedo, and C. Rossi, "3D path planning using a fuzzy logic navigational map for planetary surface rovers," in *procs. of the 11th Symposium on Advanced Space Technologies in Robotics and Automation*, 2011.
10. G. Ishigami, K. Nagatani, and K. Yoshida, "Path planning for planetary exploration rovers and its evaluation based on wheel slip dynamics," in *procs. of the IEEE International Conference on Robotics and Automation*, Roma, Italy, April 2007, pp. 2361–2366.
11. E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.
12. D. L. Page, A. F. Koschan, and M. A. Abidi, "Ridge-valley path planning for 3D terrains," in *procs. of the 2006 IEEE International Conference on Robotics and Automation*, Orlando, Florida, USA, May 2006, pp. 119–124.
13. J. Foley, A. van Dam, S. Feiner, and J. Hughes, *Computer Graphics: Principles and Practice*. Addison-Wesley, 1992.
14. R. L. Kirk, E. Howington-Kraus, M. R. Rosiek, J. A. Anderson, B. A. Archinal, K. J. Becker, D. A. Cook, D. Galuszka, P. E. Geissler, T. M. Hare, I. M. Holmberg, L. P. Keszthelyi, B. L. Redding, W. A. Delamere, D. Gallagher, J. Chapel, E. M. Eliason, R. King, and A. S. McEwen, "Ultrahigh resolution topographic mapping of Mars with MRO HiRISE stereo images: Meter-scale slopes of candidate Phoenix landing sites," *Journal of Geophysical Research: Planets*, vol. 113, no. E3, pp. n/a–n/a, 2008.