Algorithms for rough terrain trajectory planning

Alain Haït, Thierry Simeon, Michel Taïx

ALGORITHMS FOR ROUGH TERRAIN TRAJECTORY PLANNING

Alain HAÏT, Thierry SIMEON and Michel TAÏX

LAAS-CNRS, 7 avenue du Colonel-Roche, 31077 Toulouse, France
{hait,nic,taix}@laas.fr

Abstract— This paper deals with motion planning on rough terrain for mobile robots. The aim is to develop efficient algorithms, suitable for various types of robots. On rough terrain, the planned trajectory must verify several *validity constraints* : stability of the robot, mechanical limits and collision avoidance with the ground. Our approach relies on a static and kinematic model of the robot. Efficient geometric algorithms have been developed, taking advantage of each vehicle's specificities. Motion planning relies on incremental search in the discretized configuration space and uses efficient heuristics based on terrain characteristic to limit the size of search space. Simulation results present trajectories planned in a few seconds. The second part takes into account uncertainties to improve trajectory robustness: uncertainties on the terrain model and the position of the robot. The adaptation of the previous algorithms allow to find robust trajectories, without excessive time increase.

Key words: trajectory planning, mobile robots, rough terrain, uncertainties

1. INTRODUCTION

Autonomous navigation on natural terrains is a complex and challenging problem with potential applications ranging from intervention robots in hazardous environments to planetary exploration.

Success of Sojourner rover mission on Mars gave a new fervor to planetary exploration with robots. Future exploratory missions will include more autonomous for complex tasks : pick up samples, long range displacements, etc. To perform such tasks on outdoor terrain, especially for planetary exploration, a high level of autonomy is required because the robot travels long distances without human intervention. For example the future mars rovers can only communicate twice per day and must perform tasks between them equivalent in distance to the entire Sojourner mission. In this context autonomous navigation on natural terrain is still a complex and challenging problem .
Several systems for outdoor navigation have been developed. The *Rocky* robots [23] from Jet Propulsion Laboratory can reach specified positions, using a behavior control approach [17, 11]. *Nomad* [1] from Carnegie Mellon University operated more than 200 kilometers using various navigation modes (principally teleoperation). Another system from CMU, *Navlab* performed autonomous navigation on natural terrain [25].

When terrain is known the motion planning problem can be formulated as an optimization problem [22]. The terrain is represented by a spline surface and the robot by a point mass. The cost function optimized is function of path distance and traversability measure.

An other approach used sensor information to incrementally replan optimal path with partial information at the beginning. The sensor information update a grid-based terrain model to find locally optimal path [24].

In [16], [27] the rover as to work in unknown natural terrain and the path planning is an iterative sensor-based approach. Target waypoint are plan and simple sensor-motion,

motion-to goal or boundary-following ensure global convergence.

Simulated dynamic interaction have been studied in [4] without motion planning. Dynamic behaviours and interaction wheel-ground have been modelised in [18] [14] and used in [7] for motion planning. Physical based model are also used with genetic algorithms [10] to select actions sequence.

[26] compare genetic path planner with fuzzy description terrain to global optimization planner with parametric terrain surface. A fuzzy approach with traversability index is used in [21].

The adaptative navigation approach developed at LAAS within the framework of the EDEN experiment [6] demonstrated autonomous short-range navigation in a natural environment gradually discovered by the robot. The approach combines various navigation modes (*reflex, 2D and 3D*) in order to adapt the robot behavior to the complexity of the environment. The selection of the adequate mode is performed by a specific planning level, the navigation planner ([15]), which reasons on a global qualitative representation of the terrain built from the data acquired by the robot.

Motion safety is a crucial problem for any exploration task on known or unknown terrain. It requires placement validity checking all along the trajectory. Accurate models of the robot, the terrain and their interactions are necessary to guarantee validity, but generally involve time-consuming computations. For example, an approximate cell decomposition of the admissible configuration space [9] [8] satisfying stability and mecanical constraints can be build with expensive time.

The contribution of this paper is to propose efficient algorithms for motion planning on rough terrain [12] . It is a compromize between global motion planner [22] but for simple point robot description and appropriate fine physical models [7] with long time computing.

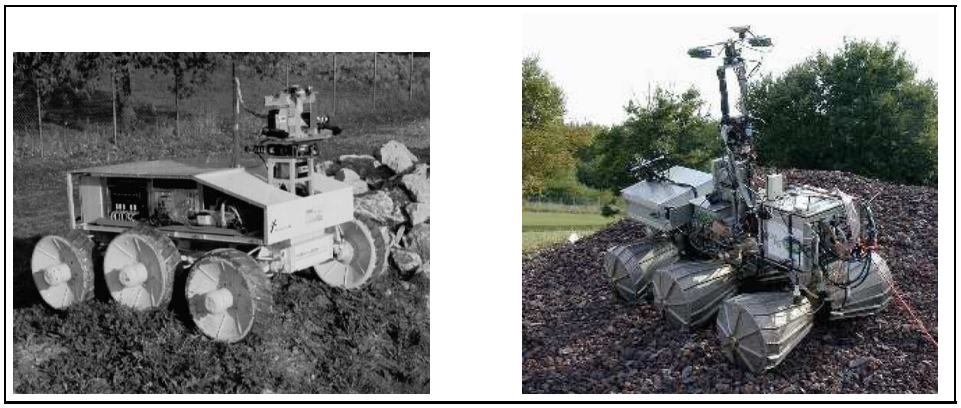

Figure 1: LAAS mobile robots

Our objective is twofold :

- *performance time* should be realistic, compatible with the requirements of exploration missions. Robot placement and validity checking durations are improved, and path search use efficient heuristics based on the terrain characteristics. The approach is based on precomputed models, exploiting the robot's kinematics and the model of the terrain. This approach is illustrated on two types of robots, and could be applied on other types.

- we look for *robust trajectories*, *i.e.* that remain safe in spite of uncertainties of real world. Uncertainties on the terrain model due to sensor acquisition and on robot placement due to imperfect robot control are introduced in the algorithms without major modification.

This paper is organized as follow. Section 2 presents the different models and constraints taken into account for autonomous motion planning. Section 3 describes the geometric algorithms for efficient placement and constraint checking for two types of mobile robot (see Fig. 1). The next section presents the path planning approach with simulation results. The uncertainties on the terrain model and robot position can be deal with the same method in section 5. The paper concludes with a discussion of ideas for future work.

2. PROBLEM STATEMENT

Given the models of the terrain surface and the vehicle geometry, the problem is to find a feasible motion between two configurations, while respecting a set of constraints related to the safety of the motion.

2.1 Models

The terrain is described by ruled surface patches defined from an elevation map in z associated with a regular grid in (x, y). Elevation z at a point (x, y) is unique and is computed from the elevations of the four points of the grid around it (see Fig.2) :

$$z(x, y) = f(z_{(i,j)}, z_{(i+1,j)}, z_{(i,j+1)}, z_{(i+1,j+1)})$$

The model of the terrain is obtained by fusion of local maps built from 3D sensor

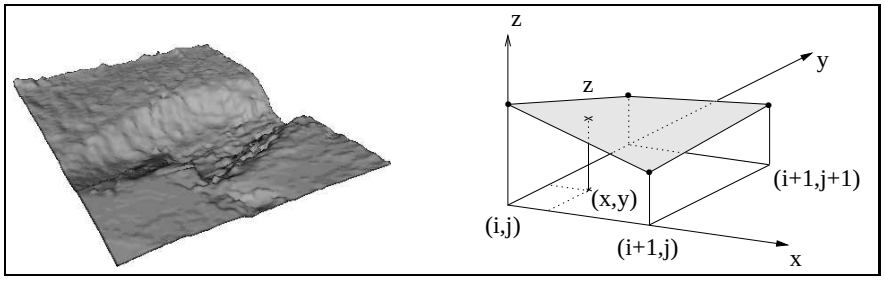

Figure 2: Model of terrain and elementary patch

data (*e.g.* rangefinder [19]). This method provides an evaluation of uncertainty on the elevation z. Uncertainty is given as an interval δz associated with each point of the grid. Section 5 presents how uncertainty is handled in our approach. We consider wheel-driven mobile robots whose articulated chassis and/or suspensions allow to adapt their shape to the terrain irregularities. Given a local frame $\mathcal{R}_{robot}$ fixed on the body of the robot, the configuration of the robot (with refer to a global frame on the terrain) is defined by $(6 + n)$ parameters :

- the position/orientation of $\mathcal{R}_{robot}$ (6 parameters) ;
- n parameters corresponding to the joints and suspensions of the robot.

However these parameters are not independent. On the assumption that the robot velocity is low and the terrain is not deformed by the vehicle, the *placement* of the robot on the terrain is defined by :

$$p = (q, r(q))$$

where the configuration $q(x, y, \theta)$ represents the horizontal position/heading of the robot in the global frame, and $r(q)$ are the parameters associated to the values of the joints or suspensions and the vertical position of $\mathcal{R}_{robot}$. The dependencies between the parameters of r and q come from the contact relations between each wheel and the terrain. The search space is thus reduced to the three dimensional configuration space $CS = (x, y, \theta)$.

2.2 Constraints

In order to guarantee the safeness of the placements, some *validity constraints* are defined :

- stability of the robot ;
- collision avoidance between the body of the robot and the terrain ;
- mechanical constraints expressing that the contact between the wheels and the terrain can be maintained without exceeding the limits of the articulations and suspensions.

We denote by C_{free} the subset of CS where the validity constraints are verified.
To take into account motion without sliding a kinematic constraint is applied on the direction of the robot's velocity.

2.3 Motion planning

The motion planning problem can be classically formulated as the problem of finding a path connecting two given configurations and lying in C_{free} and must also verify the kinematic constraint.

This paper presents the motion planner developed at LAAS and adapted to various types of robots.

3. GEOMETRIC ALGORITHMS FOR EFFICIENT PLANNING

In this section we present some algorithms related to robot placement and validity tests. Since these functions are much used during planning, we must pay a particular attention on their duration. In that aim, specific algorithms adapted to the geometry of the robots have been developed.

3.1 Robot placement

Given the configuration $q(x, y, \theta)$, the placement function determines the elevation of $\mathcal{R}_{robot}$'s origin and the values of the passive joints and suspensions of the robot. Due to the terrain irregularities, this determination is not obvious. An analytical solution of placement would be time-consuming. Consequently, iterative algorithms have been developed in order to provide a good approximate placement within short time.

3.1.1 Example of a rigid body

Model of the robot The body of the robot is modeled by a polyhedron R. We define a local frame $\mathcal{R}_{robot} = (G, \vec{u}, \vec{v}, \vec{w})$, where G is the gravity center of the robot, and $(\vec{u}, \vec{v}, \vec{w})$ are respectively its longitudinal, lateral and vertical axis (see Fig. 3).

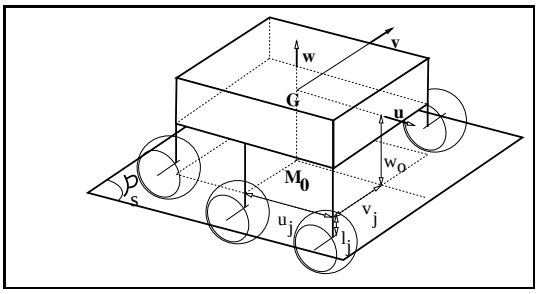

Figure 3: Model of the robot

The n wheels are attached to R by passive suspensions which are modeled by springs. We assume that, when all the springs are in their steady state, the wheels belong to the same plane $\mathcal{P}_{\mathcal{S}}$ perpendicular to w axis, at height w_0 in the local frame. Then, the coordinates of wheel j are $(u_j, v_j, w_0 + l_j)$ where u_j and v_j are some fixed values related to the geometry of the robot, and l_j is the algebraic extension of the spring ($l_j = 0$ being its natural length). That means that the springs always keep a vertical orientation in the robot's frame.

Placement The position of the robot's body is defined by the 6-dimensional vector $p(x, y, z, \theta, \phi, \psi)$, where (x, y, z) are the coordinates of point G, and (θ, ϕ, ψ) are respectively the horizontal orientation of **u** axis, the roll angle and the pitch angle.
The placement of the robot results from its weight and the reaction of the ground exerted through the springs. The static equilibrium state is reached when the total energy of the robot is minimized. In the following, we consider a simple energy function, including only the compression energy of the springs: $\mathcal{E} = \sum_{j=1}^{n} k {l_j}^2$ where k is the stiffness coefficient of the springs.
Moreover, we only consider the placements which keep all the wheels in contact with the terrain. Let us denote by $L_j(p)$ the function which associates to a given value of the vector p, the value of the spring extension l_j such that the wheel j is in contact (without intersecting) with the terrain (see Fig. 4). We denote by C_j the contact point between wheel j and the terrain.
Therefore, the energy can be expressed as a function of the placement $\mathcal{E}(p) = \sum_{j=1}^{n} k L_j^2(p)$ and for a given configuration $q(x, y, \theta)$, the remaining parameters of vector p result from the minimization of this function. Vector $(z(q), \phi(q), \psi(q))$ is the solution of:

$$min_{z,\phi,\psi} \sum_{j=1}^{n} L_1^2(p) \tag{1}$$

The spring extensions $l_j(q)$ are obtained from the evaluation of the functions $L_j(p)$ for the computed placement.
Algorithm **Place_robot**(x, y, θ) allows to compute efficiently an approximate placement. The method consists in applying iteratively a least square algorithm to improve an initial estimation of the parameters z,ϕ and ψ.

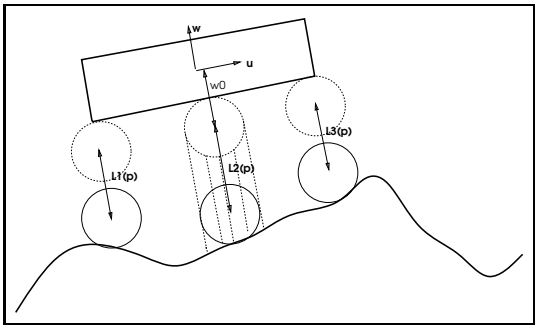

Figure 4: Definition of $L_j(p)$

Each iteration i is aimed to decrease the value of energy $\mathcal{E}(p_i)$. Let z_i,ϕ_i and ψ_i denote the value of the placement parameters at the beginning of iteration i. For the corresponding placement p_i, the algorithm $L_j(p_i)$ allows to compute the value $\mathcal{E}_i = \mathcal{E}(p_i)$ and the n contact points C_j between the wheels and the terrain. The least square method is then used to obtain the equation of the plane $\mathcal{P}_i$ which minimizes the quadratic mean of the distances to the points C_j (step 10.). The new values of the placement parameters are deduced from this cdplane: ϕ_i and ψ_i are computed such that the axis $\vec{w}$ of $\mathcal{R}_{robot}$ coincides with the plane normal (step 11.). z_i is computed such that all the wheels contact this plane for a null deformation of the springs (see Fig. 5).

1: **Place_robot**(x, y, θ)
2: $i := 0$
3: $z_0 := z_{terrain}(x, y)$; $\phi_0 := 0$; $\psi_0 := 0$;
4: $p_0 := (x_G, y_G, \theta, z_0, \phi_0, \psi_0)$;
5: $\forall j \in [1; n],\ L_j(p_0) \Rightarrow C_j$
6: $\mathcal{E}_0 := \sum_{j=1}^{n} k {L_j}^2(p_0)$;
7: **repeat**
8: $\quad i := i + 1$;
9: $\quad$ determination of $\mathcal{P}_i$ from $C_{j,j\in[1,n]}$;
10: $\quad$ determination of $z_i, \phi_i, \psi_i \Rightarrow p_i$;
11: $\quad \forall j \in [1; n],\ L_j(p_i) \Rightarrow C_j$
12: $\quad \mathcal{E}_i := \sum_{j=1}^{n} k {L_j}^2(p_i)$;
13: **until** $\mathcal{E}_i < \mathcal{E}_{i-1}$
14: **Return**$(p_{i-1}, L_j(p_{i-1}))$;

Convergence is not guaranted for *Place_robot* but the algorithm give good result in practice. On very irregular terrain the placement is not correct, but in this case the placement is impossible because all wheels are not in contact with the terrain.

3.1.2 Example of an articulated chassis

Model of the robot. Let us consider a three-axle articulated robot (see Fig.6). We define a local frame $\mathcal{R}_{robot}(M, \vec{x}, \vec{y}, \vec{z})$ where M is the center of the middle axle, $\vec{x}$ is the longitudinal axis of the robot and $\vec{z}$ the vertical axis. Each axle is articulated around the $\vec{x}$-axis; $\varphi_{<axle>}$ is the roll angle of an axle, measured with respect to the horizontal $\vec{y}$-axis. The axles are linked by two rigid bodies articulated around the $\vec{y}$-axis; ψ_F and ψ_R are the pitch angles of the front and the rear body, measured with respect to the

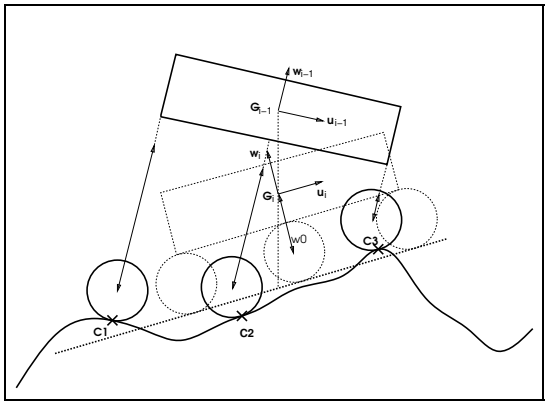

Figure 5: Iterative computation

horizontal $\vec{x}$-axis.

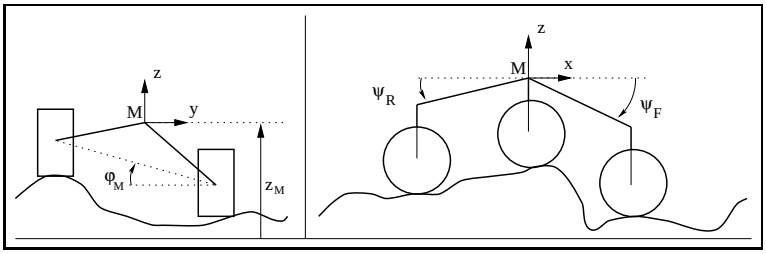

Figure 6: Model of an articulated robot

For this locomotion system, configuration $\mathbf{q} = (x_M, y_M, \theta)$ and the other placement parameters of $\mathbf{r}$ are the roll angles $\varphi_F, \varphi_M, \varphi_R$ of the axles, the pitch angles ψ_F, ψ_R of the bodies and the elevation z_M of the middle axle. As mentioned earlier, the role of the placement algorithm is to compute the vector $\mathbf{r}(\mathbf{q})$ associated with a given configuration $\mathbf{q}$. After considering the case of a single axle, we show how this basic algorithm can be used for the placement of the chassis.

Placement of a single axle. Given the (x, y) position of the axle center and its orientation θ, the placement basically consists in finding the φ angle for which the right and left wheels, W_r and W_l have the same vertical distance to the terrain (afterwards, the elevation z of the center is easily deduced).
Let $dist = F(\varphi)$ be the distance from W_l to the terrain when W_r is put in contact. Then we have:

$$F(\varphi) = 2l \sin \varphi - (z_l(\varphi) - z_r(\varphi)) \tag{2}$$

where $z_l(\varphi), z_r(\varphi))$ are the elevations of the wheels contacting the terrain for the xy-positions of their center associated with the angle φ (see Fig. 7). The placement is obtained for the solution φ_{sol} of the equation $F(\varphi) = 0$. This equation cannot be solved analytically since $z_l(\varphi), z_r(\varphi)$ closely depend on the terrain geometry. However one can

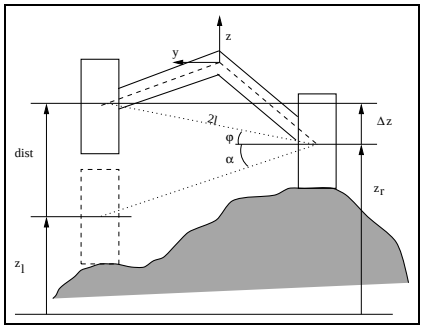

Figure 7: Placement of an axle

remark that for the case of a flat terrain having a slope α, we would have:

$$z_l(\varphi) - z_r(\varphi) = \cos\varphi \tan\alpha \tag{3}$$

and that the solution φ_{sol} would equal the slope α. The algorithm **Place_axle**(x, y, θ) presented below uses this remark to iteratively refine an approximation of φ_{sol}. At iteration i, evaluating $z_l(\varphi) - z_r(\varphi)$ for a value φ_i and applying Eq. 3, allows to get the slope α which would be the solution for a planar approximation of the terrain (step 6). The next value φ_{i+1} is computed from φ_i and α such that the line passing through $(\alpha, F(\alpha))$ and $(\varphi_i, F(\varphi_i))$ cuts the horizontal axis ($F(\varphi) = 0$) at φ_{i+1} (step 9).

1: **Place_axle**(x, y, θ)
2: $i := 0$; $\varphi_0 := 0$;
3: $dist_0 := F(\varphi_0)$;
4: **while** $dist_i > \varepsilon$ **do**
5: {evaluation: flat terrain}
6: $\alpha := atan2((z_l(\varphi_i) - z_r(\varphi_i))/\cos\varphi_i)$;
7: $dist_\alpha := F(\alpha)$;
8: {computation of φ_{i+1}}
9: $\varphi_{i+1}(\alpha, dist_\alpha, \varphi_i, dist_i)$;
10: $dist_{i+1} := F(\varphi_{i+1})$;
11: $i := i + 1$;
12: **end while**
13: **return** $\varphi_{sol} := \varphi_i$; z_{sol};

Tests performed [12] on highly irregular terrains show that function $F(\varphi)$ varies very smoothly and that the risk of local minima is very limited between $\varphi = 0$ and the solution φ_{sol}. In many cases, the solution is obtained after one iteration.

Placement of the chassis. The placement consists of three steps: the middle axle is first placed by using the algorithm presented above. The parameters of the front and rear axles (elevation z and angles φ and ψ) are then determined by another algorithm based on the same principle, but adapted in order to introduce the link with the middle axle as an additional constraint.

Consider the case of the front axle (the placement of the rear axle is exactly the same). Knowing the elevation z_M of the middle axle, we can compute, for a given value of parameter ψ_F, the coordinates (x_F, y_F, z_F) of the front axle's center F. Applying **Place_axle**(x_F, y_F, θ) determines φ_F and the elevation z of F when the front axle is placed on the terrain, regardless of the link with the middle axle.
Let $G(\psi_F) = z_F - z$ denote the difference between both elevations. The constraint imposed by the link is verified for the solution of $G(\psi_F) = 0$. The values of the parameters ψ_F and φ_F are computed successively by an iterative algorithm **Place_other_axle** similar to the previous one. Each iteration first evaluates ψ_{i+1} regardless to the link with the middle axle: β is determined from z, z_M and the distance d between both axles. Then we compute ψ_{i+1} such that the line passing through $(\beta, G(\beta))$ and $(\psi_i, G(\psi_i))$ cuts the horizontal axis $(G(\psi) = 0)$ at ψ_{i+1}.

1: **Place_other_axle**(x_M, y_M, z_M, θ)
2: $i := 0$; $\varphi_0 := 0$; $\psi_0 := 0$;
3: $dist_0 := G(\psi_0)$;
4: **while** $dist_i > \varepsilon$ **do**
5: {evaluation: no link with the middle axle}
6: $\beta := atan2(\frac{dist_i + d.sin\psi_i}{d})$;
7: $dist_\beta := G(\beta)$;
8: {computation of ψ_{i+1}}
9: $\psi_{i+1}(\beta, dist_\beta, \psi_i, dist_i)$;
10: $dist_{i+1} := G(\psi_{i+1})$;
11: $i := i + 1$;
12: **end while**
13: **return** ψ_{sol}; $\varphi_{sol} := \varphi_i$; z_{sol};

Note that in this case, the evaluation of $G(\psi_F)$ requires a call to the function **Place_axle** which is then used once for the initialization and twice for each step of the algorithm. Again the solution is generally found in only one iteration.
Figure 8 shows an example of placement computed by the algorithm.

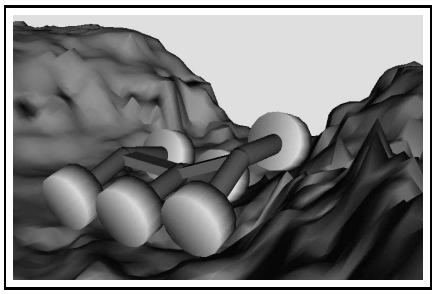

Figure 8: Example of placement

Placement optimization. The motion planner extensively uses the placement algorithms when evaluating the validity of the elementary motions generated during the search. Moreover the placement of the chassis requires itself several calls to the function **Place_axle**. The aim of this preprocessing step is to significantly reduce the computational cost of the placement algorithm when used by the planner. It consists in slicing the

orientation parameter, and in computing two surfaces denoted $\mathcal{S}_\varphi(x,y)$ and $\mathcal{S}_z(x,y)$ for each slice θ_i. These surfaces respectively correspond to the roll angle and the elevation of the axle when it is placed at position (x,y), with an orientation θ_i. Like the terrain, both surfaces are represented by ruled surfaces patches defined from a elevation map in φ (or z) associated with a regular grid in (x,y). They are computed by applying the function **Place_axle** at every point of the grid.

The online placement of the chassis simply computes the φ (or z) parameter for a position (x,y,θ) of the axle, as follows:

- for θ lying between θ_i and $\theta_{i+1} = \theta_i + \Delta\theta$ of the slicing, let $\mathcal{S}_\varphi^i$ and $\mathcal{S}_\varphi^{i+1}$ denote the associated φ-surfaces.
- $\varphi_i = S_\varphi^i(x,y)$ and $\varphi_{i+1} = S_\varphi^{i+1}(x,y)$.
- the roll angle φ is obtained by a linear interpolation: $\varphi = (\varphi_{i+1} - \varphi_i)\frac{\theta-\theta_i}{\theta_{i+1}-\theta_i} + \varphi_i$

Figure 9 presents a terrain model and the associate surfaces S_z and S_φ at orientation $\theta = 0$ and $\theta = 90°$ (represented as terrain models).

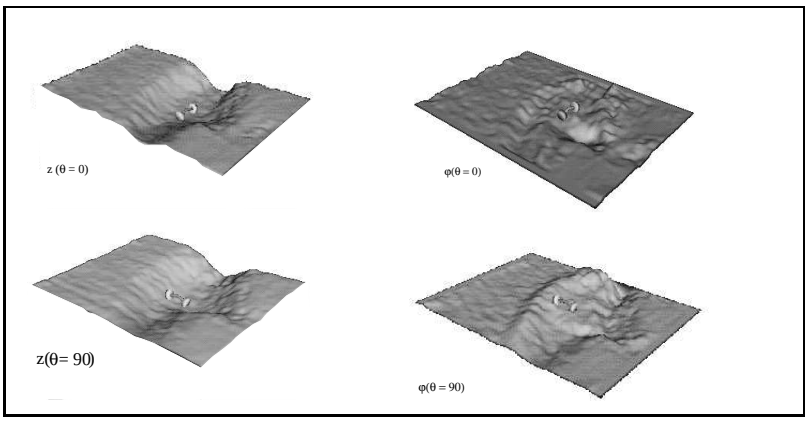

Figure 9: Placement surfaces

Computation time of placement and preprocessing step are given in §4.3.

3.2 Validity constraints

We consider now the problem of checking whether a given configuration q belongs to C_{free} or not. The placement parameters must verify the validity constraints : stability, mechanical limits and collision avoidance.

3.2.1 Stability of the robot

Case of a rigid body. The robot does not tip-over, which requires that the projection of its gravity center remains inside the convex hull of the projections of all the contact points (called the support polygon). The shape of this polygon is clearly a function of ϕ, ψ and of the lengths $(l_j)_{j=1\ldots n}$. However, we assume that the less stable position is obtained when all the springs are the longest. The stability thus only depends on ϕ and

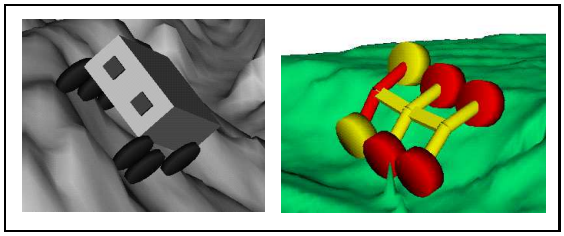

Figure 10: Stability constraint

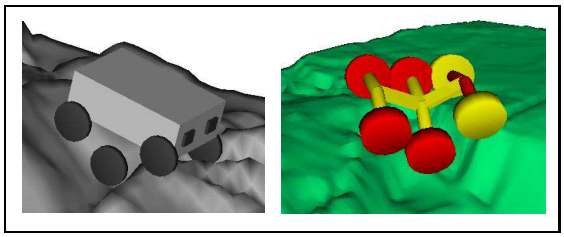

Figure 11: Mechanical constraints

ψ, and the subset $S \subset]-\frac{\pi}{2}, \frac{\pi}{2}[^2$ verifying this condition can be easily determined. Thus, the stability constraint is verified for a given q iff:

$$(\phi(q), \psi(q)) \in S$$

In order to avoid sliding, and to simplify constraint checking, a subset $S'[-\phi_{max}, \phi_{max}] \times [-\psi_{max}, \psi_{max}] \subset S$ is used.

Case of an articulated robot. The constraint used to check that some static stability conditions (*e.g.* no tip-over and absence of sliding) are satisfied simply consists in verifying that the global roll and pitch angles of the robot do not exceed some predefined stability limits (see Fig. 6).
The global roll angle of the robot is defined by the average of the three roll angles of the axles.

$$\left| \frac{\varphi_F + \varphi_M + \varphi_R}{3} \right| < \varphi_{StabMax}$$

The global pitch angle is defined by the angle between the horizontal plane, and the line going through the centers of the front/rear axles. The pitch stability constraint is:

$$\psi_{StabMin} < \frac{\psi_R - \psi_F}{2} < \psi_{StabMax}$$

3.2.2 Mechanical constraints

Case of a rigid body. Let L_{max} be the maximal deformation allowed for the springs. A configuration q is safe iff:

$$\forall j \in [1, n], |l_j(q)| < L_{max}$$

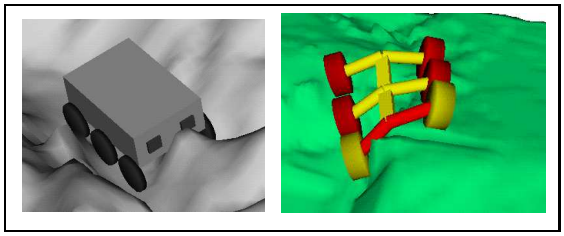

Figure 12: Collision

Case of an articulated robot. The mechanical constraint correspond to angular limits for the relative angle between two successive axles

$$|\varphi_M - \varphi_R| < \varphi_{max} \quad \text{and} \quad |\varphi_M - \varphi_F| < \varphi_{max}$$

and for the angle made by the two bodies (concave or convex configuration of the robot)

$$\psi_{Min} < \psi_F + \psi_R + \pi < \psi_{Max}$$

3.3 Collision detection

The last validity constraint guarantees that, except the wheels, the other parts of the locomotion system do not collide with the terrain irregularities.

3.3.1 Hierarchical model of the terrain

In order to efficiently check the collision between the robot and the terrain, we construct a hierarchical model (a quadtree) from the model of the terrain (see Fig. 13). The root of the quadtree is the cell corresponding to the whole terrain, and it is recursively subdivided into four smaller cells. To each of these cells, we associate the mean plane to the points of the corresponding part of terrain (with least square method), and the minimal and maximal distance to this plane. Thus a cell can be represented by a parallelepiped which upper and lower faces are parallel to the mean plane (see Fig. 14).

Subdivision stops in the following cases :

- when the size of the cell is equivalent to the dicretization of the terrain grid ;
- when the height h of the cell (*i.e.* the difference between the maximal and the minimal distance to the plane) is lower than a given limit h_0.

An advantage of this method is that the cells are not systematically subdivided : in smooth areas (even with slope), the structure remains compact.

3.3.2 Collision checking

The quadtree is used to detect collision between polygonal faces and the terrain. This test is thus performed several times with the faces that represent the body of the robot. The following procedure is recursively applied, starting from the root of the quadtree :

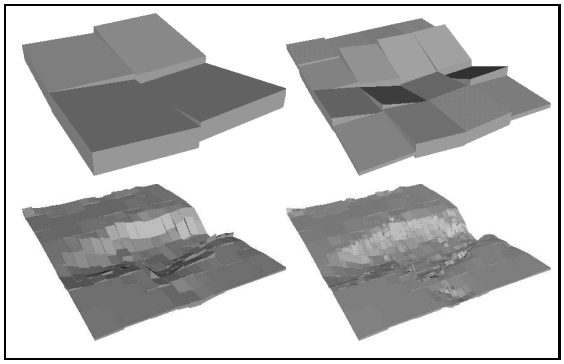

Figure 13: The hierarchical model (levels 1,2,4,6)

- if the polygonal face P does not intersect the parallelepiped associated to a given cell, then we are certain that P does not collide with the terrain (face **a** on Fig.14) ;
- similarly, when P intersects both upper and lower face of the parallelepiped we can directly conclude the existence of a collision (face **b** on Fig.14) ;
- otherwise, we need to consider a finer approximation of the terrain by analyzing the four descendant of this node.

When a leaf of the quadtree is reached and does not allow to conclude, a more expensive test with the corresponding patch surface is applied.

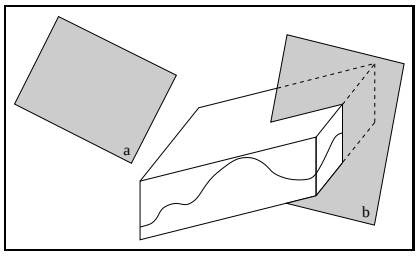

Figure 14: Collision between faces and a cell

Collision is checked with the lower faces of the polyhedron representing the robot's body and/or axles (see Fig.15)

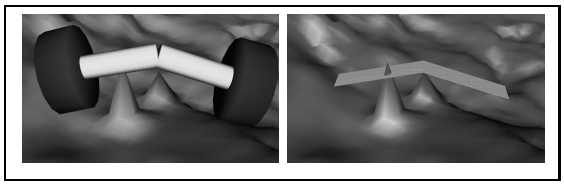

Figure 15: Collision of an axle with the terrain

4. PATH PLANNING

4.1 Principle

The planning principle was originally proposed in [2]. The idea is to build incrementally a graph $\mathcal{G}$ of discrete configurations in C_{free}. These configurations can be reach from q_i by applying sequences of elementary controls during a short time interval.
The configuration space is decomposed into an 3D-array of small cells of equal size and a node of the graph represents a CS cell. Each arc of the graph corresponds to a trajectory portion between two nodes, obtained by applying a given control. These controls are line segment or arc of circle of minimal radius for the robot in the plane (x, y). The discrete values of the linear and angular velocity v and ω are :

$$(v, \omega) \in \{-V_{lin}, V_{lin}\} \times \{-\Omega, 0, \Omega\}$$

These controls respect the non-holonomic constraint in the plane and represent a good approximation on the rough surface.

In order to limit the size of the graph each new cell is marked and one node is at most generated in each cell. Figure 16 details the successors developed by applying the

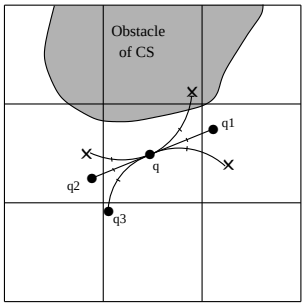

Figure 16: Incremental graph

controls from the node q. A node is created when a new cell is reached, and if the cell has not been visited yet and the portion of trajectory verifies the validity constraints.
For example in the figure 16, only the nodes $q1$, $q2$ and $q3$ are created from node q. To guarantee the trajectory validity, the constraints are tested at discrete configurations along each portion of trajectory.

4.2 Graph search

We use a classical A^* algorithm to search the graph $\mathcal{G}$. Search starts at node q_{init}, and nodes are recursively developed until q_{goal} is reached. For each node created, two features are evaluated :

- the *cost* of the path from q_{init} to the current node ;
- the *heuristic function* : estimate of the cost from the current node to q_{goal}.

The node with the best value $cost + heuristic$ is developed first. The heuristic based on terrain characteristics avoid creation of great number of nodes.

4.2.1 Arc cost

The cost assigned to the arc connecting two adjacent nodes is computed from the distance between the two configurations associated to these nodes. This distance is weighted in order to penalize the *dangerous* configurations, *i.e.* configurations for which the placement parameters (angles $\phi(q)$ and $\psi(q)$ or some of the links) are close to their limits. Therefore, the minimum-cost trajectory is a compromise between the distance crossed by the robot and the security along the trajectory.

4.2.2 Search guidance

The choice of a good heuristic function is very important to limit the graph development. In our problem, the terrain characteristics must be taken into account in order to give a good approximate of the path cost between a node and the goal.

Cost bitmap for each point of the terrain grid, the slope and the roughness of a circular domain centered at this point with a radius related to the size of the robot is evaluated. From these values we deduce the local *cost* of the terrain, rough estimation of the difficulty for the robot to cross the domain (see Fig. 17).

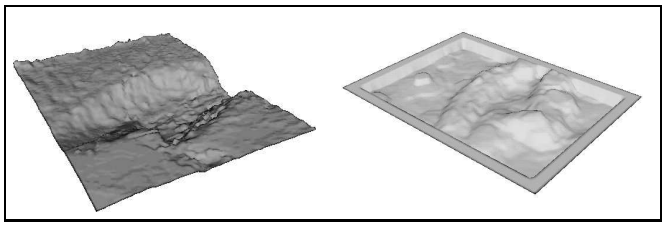

Figure 17: The terrain model and its associated cost bitmap

Potential bitmap The potential bitmap is computed by using a classical wave propagation technique which integrate the cost across the bitmap, starting with a null potential at the goal position of the robot. Propagation is done in priority for the pixels whose combination $(cost + potential)$ is the lowest. This guarantees a compromise between path length and local cost of the terrain.

Heuristic function For a configuration $q(x, y, \theta)$, the heuristic $H(q)$ is computed, as in [3], by a linear combination of the potential evaluated at two control points of the robot.

Close to the goal the Reeds and shepp curves [20] are used to take into account the goal orientation.

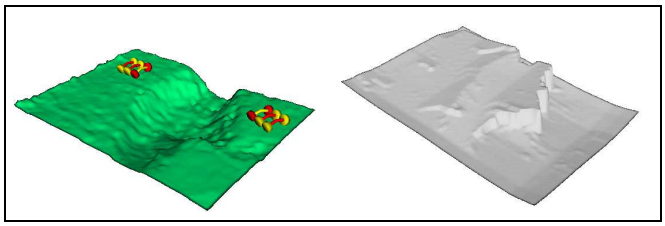

Figure 18: The initial/goal configurations on the terrain and the potential bitmap

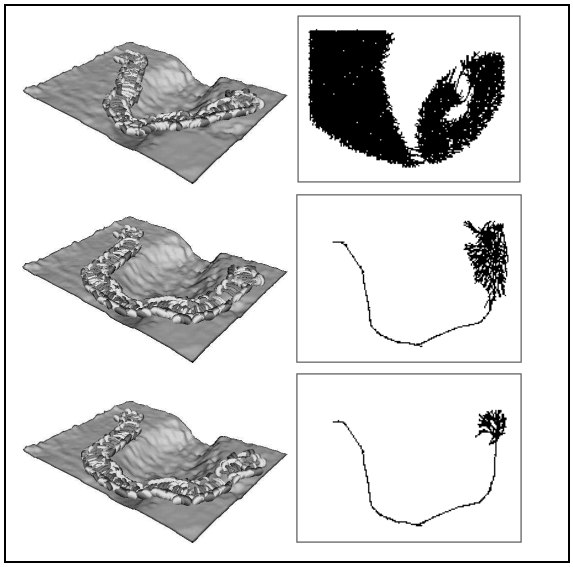

Figure 19: motion planning with various search guidances

4.3 Results

Figure 19 presents the trajectories found for the problem of Fig. 18, and the nodes developped for each case. Three different heuristic functions have been used to guide the search.
First, a simple strait-line distance heuristic (top). This guidance is not efficient for rough terrain, because of the validity constraints, and many nodes are created before the goal is reached. The number of nodes, placements and the computation time is given in the following table.
The two other examples (middle and bottom on Fig. 19) correspond to a search guidance with potential bitmap. The latter includes an additional guidance that takes into account the robot's orientation when it is closed to the goal. The number of nodes and the computation time show that this guidance is much more efficient than the simple strait-line distance guidance.

	top	mid	bot
Nodes created	33004	2278	775
Nodes developed	30579	1583	413
Placements	80885	5425	1757
Time (sec.)	125.1	4.47	1.83

Figure 20 presents the results of preprocessing on various terrains. Computation time is linear with the size (number of points) and quadratic with the resolution of the terrain. Tests have been performed first for a given resolution (25cm between two consecutive points), and then for a given size of terrain (20000 points).

4.4 Smoothing

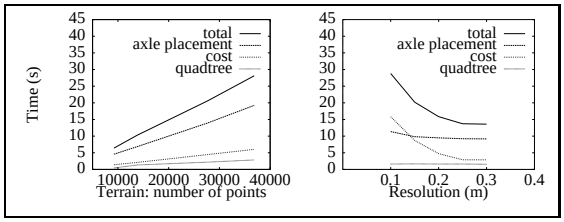

Figure 20: Preprocessing: computation time

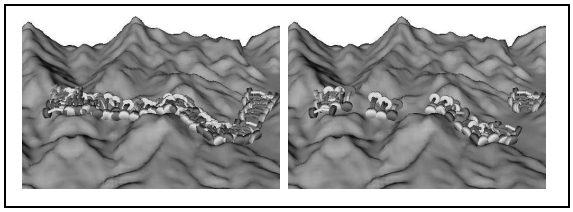

Figure 21: Trajectory smoothing

The final trajectory is a sequence of elementary motions, segment and arc of circle, and many oscillations are due to control change between them. We decide to smooth the trajectory by computing a Reeds and Shepp curve [20] between random closed configurations on the trajectory (the number of change control is limited). If the Reeds and Shepp curve is valid it replaces the old portion and decrease the number of controls change.

Figure 21 represents a trajectory before and after the smoothing. Each robot drawing is associated with a different control. The number of controls decreases from 40 to 13.

5. UNCERTAINTY CONSTRAINTS

5.1 Uncertainties on the terrain model

The uncertainties can be propagated through the different steps of the terrain modeling in order to produce an elevation map with an error interval associated with each elevation of the grid points. The terrain is therefore represented by two surfaces $(\mathcal{T}_{min}, \mathcal{T}_{max})$ corresponding to the envelopes obtained from the minimal/maximal elevations. The problem is to guarantee that the validity constraints imposed to the planner are satisfied for whatever terrain lying between both envelopes.

5.1.1 Robot placement

The placement of the robot is now defined by a vector $(q, \Delta r(q))$ where Δr represents, at configuration q, the possible intervals for the joints values. These intervals are computed by applying the algorithms described in §3.1.2 for the wheels placed either on the lower or upper envelopes:

- The minimal (resp. maximal) elevation of an axle is obtained when both wheels are placed on $\mathcal{T}_{min}$ (resp. $\mathcal{T}_{max}$).
- The minimal value of $\varphi_{<axle>}$ is obtained when the left wheel is placed on $\mathcal{T}_{min}$ and the right one on $\mathcal{T}_{max}$ (conversely for the maximal value).

- The minimal values of ψ_F and ψ_R are computed for the middle axle (both wheels) placed on $\mathcal{T}_{min}$ and the other axles on $\mathcal{T}_{max}$ (conversely for the maximal value).

The preprocessing step (*cf.* §) is performed four times, placing the right and the left wheel on $\mathcal{T}_{min}$ or $\mathcal{T}_{max}$.

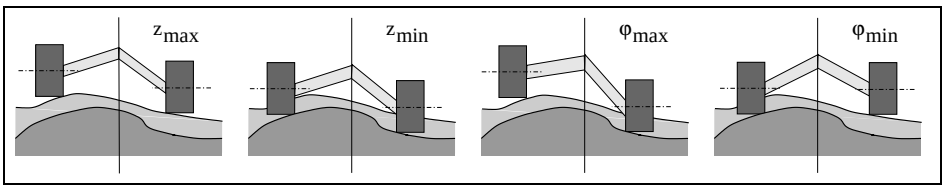

Figure 22: Placement intervals of an axle

For each slice θ, surfaces $\mathcal{S}_{zmin}, \mathcal{S}_{zmax}$, $\mathcal{S}_{\varphi min}$ and $\mathcal{S}_{\varphi max}$ represent the bounds on the parameters as functions of the (x, y) position.
During the online placement of the chassis, the middle axle parameters Δz_M and $\Delta\varphi_M$ are directly obtained from the precomputed surfaces. The function **Place_other_axle** is then applied to these surfaces to compute the parameter intervals $\Delta\varphi_F$, $\Delta\psi_F$ and $\Delta\varphi_R, \Delta\psi_R$ for the front and rear axles.

5.1.2 Validity constraints

The constraints are checked in the worst case situation obtained from the bounds of the placement intervals.

Stability: The global roll angle is checked for the extreme values of $\varphi_{<axle>}$:

$$MAX\left(\left|\frac{\Sigma\varphi_{<axle>min}}{3}\right|, \left|\frac{\Sigma\varphi_{<axle>max}}{3}\right|\right) < \varphi_{StabMax} \tag{4}$$

The minimal global pitch angle is obtained when the rear axle is placed on $\mathcal{T}_{max}$ and the front one on $\mathcal{T}_{min}$ (and conversely for the maximal value). This value may also vary as a function of the middle axle placement. However, we assume that this variation is small and the middle axle is placed on $\mathcal{T}_{min}$. Let $\psi_{Rtermin}$ (resp. $\psi_{Ftermin}$) denote the pitch angle computed for the middle and rear (resp. front) axles placed on $\mathcal{T}_{min}$. Then we have:

$$\psi_{StabMin} < \frac{\psi_{Rmin} - \psi_{Ftermin}}{2} \qquad and \qquad \frac{\psi_{Rtermin} - \psi_{Fmin}}{2} < \psi_{StabMax}$$

Mechanical constraints: The angular limit between two successive axles is checked when the roll angle of one is maximum and the other minimum:

$$MAX(\,|\varphi_{Mmin} - \varphi_{Rmax}|, |\varphi_{Mmax} - \varphi_{Rmin}|) < \varphi_{max} \tag{5}$$

$$MAX(|\varphi_{Mmin} - \varphi_{Fmax}|, |\varphi_{Mmax} - \varphi_{Fmin}|) < \varphi_{max} \tag{6}$$

and the minimal angle formed by the bodies is attained for the minimum value of both pitch angles (conversely for the maximum):

$$\psi_{Min} < \psi_{Fmin} + \psi_{Rmin} + \pi \qquad and \qquad \psi_{Fmax} + \psi_{Rmax} + \pi < \psi_{Max}$$

Collision Concerning the collision checker, the worst case corresponds to a placement of the robot on $\mathcal{T}_{min}$ and a test of intersection with $\mathcal{T}_{max}$. Consequently, only one hierarchical model corresponding to $\mathcal{T}_{max}$ is computed.

5.2 Uncertainty on the Robot Position

The robustness to the control errors is achieved by imposing the validity of a configuration domain around the planned trajectory. To guarantee the validity of the domain, we introduce the notion of neighborhood $\mathcal{N}(q)$ induced by two parameters $(\rho, \Delta\theta)$: at any configuration q, the robot may be translated by a distance ρ from its position (x, y), with an orientation $\theta \pm \Delta\theta$. The free configuration space is now defined as $C'_{free} = \{q \in CS \ / \ \mathcal{N}(q) \subset C_{free}\}$. A configuration q is declared valid, only if all the configurations of its neighborhood satisfy the validity constraints.
This approach guarantees a *corridor* of validity along the trajectory (see Fig. 23).

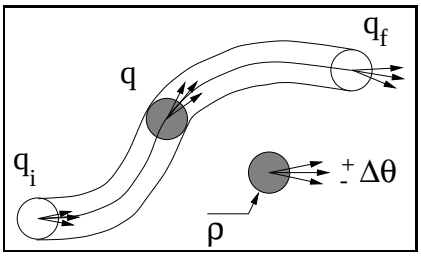

Figure 23: Validity corridor along the trajectory

5.2.1 Robot placement For the placement, we have to compute the vector interval $\Delta r(q)$ giving the limits of the parameter values for any configuration of $\mathcal{N}(q)$. The method proposed to solve this problem consists in transforming the position uncertainty into uncertainty on the placement of an axle, in order to use the precomputed surfaces of placement $\mathcal{S}_\varphi$ and $\mathcal{S}_z$.
The placement of an axle is defined by the position (x, y) of its center and its orientation θ. Since the frame $\mathcal{R}_{robot}$ is linked to the center of the middle axle, the placement of this axle depends on the neighborhood $\mathcal{N}(q)$ induced by $(\rho, \Delta\theta)$. Moreover, the placements of front and rear axles depend on different neighborhoods $\mathcal{N}_F(q)$ and $\mathcal{N}_R(q)$ because of the links between these axles and the middle one (see Fig. 24).

The placement parameter intervals Δz, $\Delta\varphi$ of each axle could be obtained by searching for the minimal and maximal values of these parameters in surfaces $\mathcal{S}_z$ and $\mathcal{S}_\varphi$, on the corresponding neighborhoods. However this method may induce two main problems:

- The extreme values of z and φ for each axle does not generally correspond to the same configuration of the robot (in $\mathcal{N}(q)$). The computation of $\Delta\psi_F$ and $\Delta\psi_R$ without considering the links between the axles may induce an overconstraint.
- During motion planning, numerous placements are performed along the portions of trajectory. As the distance between two successive placements is short, the corresponding neighborhoods $\mathcal{N}(q)$ overlap (see Fig. 25 (a)). This overlapping corresponds to configurations whose validity is tested several times while one test is sufficient.

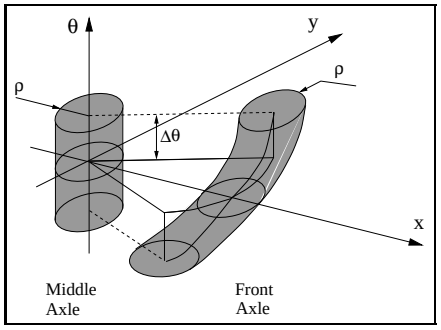

Figure 24: Neighborhoods on middle and front axle

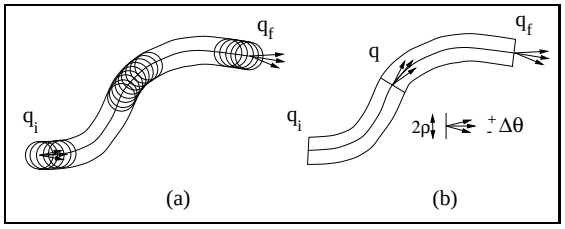

Figure 25: Uncertainty domains

A new domain of uncertainty is defined, taking into account these considerations. It corresponds to a segment normal to the trajectory and to an angular variation $\pm\Delta\theta$ (see Fig. 25 (b)). Thus overlapping is limited, and the validity tests of the configurations in the corridor are preserved in most of cases. Another advantage of this domain is that the effects of the overconstraint are reduced.
From this domain we deduce the neighborhoods $\mathcal{N}'$ on each axle. A single type of neighborhood is used for the three axles (see Fig. 26). The dimensions $2\Delta x, 2\Delta y, 2\Delta\theta$ correspond to the union of the neighborhoods of the axles.

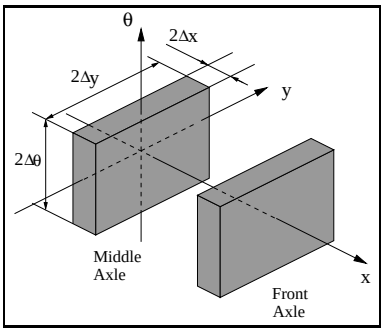

Figure 26: Modified neighborhoods on middle and front axle

The preprocessing step consists in computing the placement parameters z and φ of an axle all over the terrain for various orientations. As the neighborhood $\mathcal{N}'$ is the same for each axle, computation of the parameters intervals Δz and $\Delta\varphi$ is included in

preprocessing:

- for each slice θ_i, placement parameter surfaces $\mathcal{S}_\varphi$ and $\mathcal{S}_z$ are computed;
- for each slice θ_i, surfaces $\mathcal{S}_{\varphi\_min}$, $\mathcal{S}_{\varphi\_max}$ and $\mathcal{S}_{z\_min}$ and $\mathcal{S}_{z\_max}$ represent the min/max values of the parameters on the neighborhood $\mathcal{N}'$ around each point of the terrain.

Afterwards, the online processing of the robot placement is obtained from these surfaces as presented in §5.1.1 for uncertainty on the terrain model.

5.2.2 Validity constraints The validity of a placement interval $(q, \Delta r(q))$ is checked on the same way as for uncertainty on the terrain model, excepted for collision checking. Collision detection must take into account all the possible positions of the chassis according to the uncertainty domain. At this aim, new faces are defined, representing the inferior boundaries of the chassis elements when the robot configuration belongs to the uncertainty domain. Intersection of these faces with the terrain correspond to collisions (*cf.* §3.3.2).

5.2.3 Combining uncertainties Uncertainties on the terrain model and the robot position can be considered simultaneously with this method. This only affects the preprocessing step. First, terrain uncertainty is considered : axle placements on $\mathcal{T}_{min}$, $\mathcal{T}_{max}$ determine the surfaces $\mathcal{S}_{\varphi\_min}$, $\mathcal{S}_{\varphi\_max}$, $\mathcal{S}_{z\_min}$ and $\mathcal{S}_{z\_max}$. Then these surfaces are respectively used to determine $\mathcal{S}'_{\varphi\_min}$, $\mathcal{S}'_{\varphi\_max}$, $\mathcal{S}'_{z\_min}$ and $\mathcal{S}'_{z\_max}$, the new surfaces of min/max parameters that account for position uncertainty.

5.3 Path planning

Path planning is not different with or without uncertainty, because uncertainty models are integrated in the placement models and functions. It is however interesting to adapt the search guidance to the presence of uncertainty.

5.3.1 Search guidance

The cost bitmap is modified in order to penalize high uncertainty areas of the terrain: elevation variation Δz is now considered to compute the local slope and roughness of the terrain, increasing the cost of high-uncertainty areas. Wave expansion of the potential bitmap thus guide the robot away from these areas.

As information of the cost/potential bitmap is a 2D one, we can only consider *position* uncertainty of the robot (*i.e.* translation ρ around the position (x, y) of the robot). To account for that in the search guidance, a *growing* of the cost bitmap is performed: each point of the bitmap takes the maximal value of the cost on the circular domain (radius ρ) around it. Then the potential bitmap is computed as usual.

Both types of uncertainty can easily be combined for search guidance, applying the growing action to the cost bitmap that accounts for terrain uncertainty.

5.4 Results

Figure 27 shows the planned trajectories between two configurations on a terrain (left). Without uncertainty, the robot passes through the gorge to reach the goal (center) . The last picture (right) shows the trajectory found with uncertainty. Uncertainty on the robot position ($\rho = 30cm, \Delta\theta = 40°$) and uncertainty on the terrain (obstacles appear on the terrain) lead to a different trajectory, avoiding the gorge.

Preprocessing time increases when uncertainties are taken into account (12 sec. without

uncertainty, 72 sec. with), but then the trajectory is found as rapidly in both cases (a few seconds).

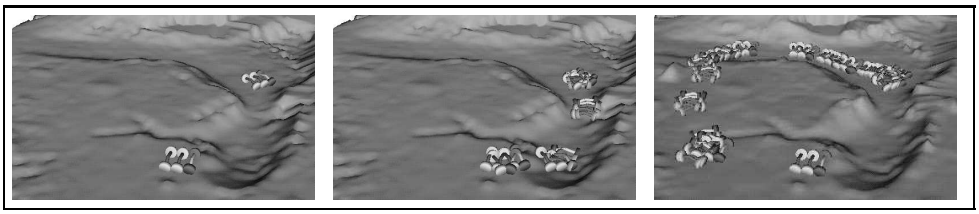

Figure 27: Path planning with uncertainty

6. CONCLUSION

In this paper, we considered the problem of planning safe motions for a robot moving on rough terrains. The main problem in these environments is the interaction between the robot and the terrain. Motion planning require numerous placements and validity checking, and these actions become difficult when the robot geometry and the terrain characteristics are considered accurately.

Two directions have been studied to solve this problem. First we developped efficient algorithms of placement and collision checking, relying on precomputed specific geometric models. Then we proposed heuristics relying on the terrain geometry to limit the exploration of search space, and consequently limit the number of placement/validity computations.

With this approach, trajectory planning is performed in a few seconds despite the accuracy of the models. Moreover, the approach can be suited to various types of robots, modifying the specific parts of the geometric algorithms.

To improve the robustness of the trajectory, uncertainties on the terrain model and the position of the robot have been considered. For a given configuration of the robot, the placement parameters become intervals of values, integrating uncertainties. Validity is then checked in the worse situation. Heuristic also accounts for uncertainties, in order to keep an efficient search. Therefore the adaptation of the algorithms allow to find robust trajectories, without excessive time increase.

Another extension of this approach has been presented in [13] to improve the robustness of the motion in long-range displacements. It consists in exploiting additional information from environment sensors (*e.g.* cameras). We consider regions of the terrain where natural landmarks are visible, solving the problem of cumulative errors. A first step based on simplified models determines a 2D trajectory called *path*. Then a 3D trajectory is planned along the path. It is a sequence of trajectories, alternately verifying or not the visibility of landmarks.

Recent work [5] use geometric constraints on articulated robot to compute elementary motion on rough terrains.

References

[1] D. Bapna, E. Rollins, J. Murphy, M. Maimone, W. Whittaker, and D. Wettergreen. The atacama desert trek: outcomes. In *Proc. IEEE International Conference on Robotics and Automation, Leuven (Belgium)*, pages 597–604, 1998.

[2] J. Barraquand and J.C. Latombe. On non-holonomic mobile robots and optimal maneuvering. *Revue d'Intelligence Artificielle*, 3(2):77–103, 1989.

[3] J. Barraquand and J.C. Latombe. Robot motion planning : a distributed representation approach. *The International Journal of Robotics Research*, 10(6):628–649, 1991.

[4] F. Ben Amar and Ph. Bidaud. Dynamic analysis of off-road vehicles. In *Proc. International Symposium on Experimental Robotics*, pages 363–371, 1995.

[5] D. Bonnafous, T. Siméon and S. Lacroix. Motion generation for a rover on rough terrains. In *IEEE International Conference On Intelligent Robots and Systems*, Hawaii (USA), October 2001.

[6] R. Chatila, S. Lacroix, T. Siméon, and M. Herrb. Planetary exploration by a mobile robot: mission teleprogramming and autonomous navigation. *Autonomous Robots*, 2(4):333–344, 1995.

[7] M. Cherif. Motion planning for all-terrain vehicles:a physical modeling approach for coping with dynamic and contact interaction constraints. *IEEE transactions on Robotics and Automation*, 2(15):202–218, 1999.

[8] B. Dacre-Wright. Planification de trajectoires pour un robot mobile sur un terrain accidenté (in french). *Thèse de l'École Nationale Supérieure des Télécommunications*, octobre 1993.

[9] B. Dacre-Wright and T. Simeon. Free space representation for a mobile robot moving on a rough terrain. In *Proc. IEEE International Conference on Robotics and Automation, Atlanta (USA)*, pages 37–43, 1993.

[10] S. Farritor, H. Hacot, and S. Dubowski. Physics-based planning for planetary exploration. In *Proc. IEEE International Conference on Robotics and Automation, Leuven (Belgium)*, pages 278–283, 1998.

[11] E. Gat, R. Desai, R. Ivlev, J. Loch, and D.P. Miller. Behavior control for robotic exploration of planetary surfaces. *IEEE Journal of Robotics and Automation*, 4(10):490–503, 1994.

[12] A. Hait. *Algorithmes pour la planification de trajectoires robustes d'un robot mobile autonome sur un terrain accidenté (in french)*. PhD thesis, Université Paul Sabatier, Toulouse, France, 1998.

[13] A. Haït, T. Siméon, and M. Taïx. Robust motion planning for rough terrain navigation. In *IEEE International Conference On Intelligent Robots and Systems, Kyongju (Corée)*, pages 11–16, 1999.

[14] S. Jimenez, A. Luciani, and C. Laugier. Predicting the dynamic behaviour of a planetary vehicle using physical modeling. In *IEEE International Conference On Intelligent Robots and Systems, Yokohama (Japan)*, pages 345–351, 1993.

[15] S. Lacroix, R. Chatila, S. Fleury, M. Herrb, and N. Simeon. Autonomous navigation in outdoors environments: Adaptative approach and experiments. In *Proc. IEEE International Conference on Robotics and Automation, San Diego (USA)*, 1994.

[16] S. Laubach, J. Burdick, and L. Matthies. An autonomous sensor-based path planner for planetary microrovers. In *Proc. IEEE International Conference on Robotics and Automation, Detroit (USA)*, 1999.

[17] L. Matthies, E. Gatt, R. Harrison, B. Wilcox, R. Volpe, and T. Litwin. Mars microrover navigation : performance evaluation and enhancement. *Autonomous robots*, 2(4):291–311, 1995.

[18] Ch. Milési-Bellier, Ch. Laugier, and B. Faverjon. A kinematic simulator for motion planning of a mobile robot on a terrain. In *IEEE International Conference On Intelligent Robots and Systems, Yokohama (Japan)*, pages 339–344, 1993.

[19] F. Nashashibi. *Perception et modélisation de l'environnement tridimensionnel pour la navigation autonome d'un robot mobile.* PhD thesis, Université Paul Sabatier, Toulouse, France, 1993.

[20] J.A. Reeds and L.A. Shepp. *Optimal path for a car that goes both forwards and backwards.* In Pacific Journal of Mathematics, vo.l 145, 1990

[21] H. Seraji. Traversability index: a new concept for planetary rovers. In *International Symposium on Artificial Intelligence, Robotics ans Automation in Space, Noordwijk (Netherlands)*, pages 159–166, 1999.

[22] D. Shiller. Motion planning for mars rover. In *First workshop on Robot Motion and Control, Poland*, June 1999.

[23] D. Shirley and J. Matijevic. Mars pathfinder microrover. *Autonomous Robots*, 2(4):283–289, 1995.

[24] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proc. IEEE International Conference on Robotics and Automation, San Diego (USA)*, pages 3310–3317, 1994.

[25] A. Stentz and M. Hebert. A complete navigation system for goal acquisition in unknown environments. *Autonomous Robots*, 2(2), 1995.

[26] M. Tarokh, Z. Shiller, and S. Hayati. A comparaison of two traversability based path planners for planetary rovers. In *International Symposium on Artificial Intelligence, Robotics ans Automation in Space, Noordwijk (Netherlands)*, pages 151–157, 1999.

[27] R. Volpe, T. Estlin, S. Laubach, C. Olson, and J. Balaram. Enhanced mars rover navigation techniques. In *Proc. IEEE International Conference on Robotics and Automation, San Francisco (USA)*, pages 926–931, 2000.