Motion planning on rough terrain for an articulated vehicle in presence of uncertainties

Alain Haït, Thierry Simeon

IEEE Int. Conf. on Intelligent Robots and Systems (IROS'96)
Osaka, Japan, November 4-8, 1996

Motion planning on rough terrain for an articulated vehicle in presence of uncertainties *

Alain Hait Thierry Siméon

LAAS-CNRS
7, Avenue du Colonel Roche
31077 Toulouse Cedex - France

Abstract

This paper addresses motion planning for a mobile robot moving on a rough terrain. We proposed in [14] a geometrical planner for a particular architecture of robot. The approach was based on a discrete search technique operating in the (x, y, θ) configuration space of the robot, and on the evaluation of elementary feasible paths between two configurations. The overall efficiency of the approach was made possible by the use of fast algorithms for solving the placement problem and checking the validity of such placements. The contribution of this paper is twofold. First, we study the extension of this approach to an articulated vehicle composed of three axles connected to the chassis by joints allowing the roll and pitch movements. We also propose new algorithms to improve the robustness of the planner by considering additionnal constraints: uncertainties in the terrain model and existence of a free channel around the computed trajectory. Some simulation results presented at the end of the paper show the effectiveness of the planner which turns out to be both fast and capable of solving difficult problems.

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.

Several systems for outdoor navigation have already been developped and demonstrated for specific tasks, e.g., road following [15], or motion in rather limited environment conditions [16, 17]. The UGV [7], as well as Ratler [8] achieve autonomous runs avoiding obstacles, but not coping to our knowledge with irregular terrain. The adaptive navigation approach currently developped at LAAS within the framework of the EDEN experiment [4] demonstrates fully autonomous navigation in a natural environment, gradually discovered by the robot. The approach combines various navigation modes in order to adapt the robot behaviour to the complexity of the environment.

In this paper, we concentrate on the motion planning algorithms required for the *3D navigation mode* [12] which is applied when the terrain to be crossed is uneven. On such terrain, irregularities are important enough and a binary partition into *crossable/forbidden* regions is not anymore sufficient: the notion of obstacle clearly depends on the capacity of the locomotion system to overcome terrain irregularities and also on specific constraints acting on the placement of the robot over the terrain. The trajectory planner therefore requires a 3D description of the terrain, based on the elevation map, and a precise model of the robot geometry in order to produce collision-free trajectories that also guarantee vehicle stability and take into account its kinematic constraints.

Although a large amount of work has been devoted to motion planning [10], only a few contributions have been recently reported for the case of a vehicle moving on a terrain. First works [13][11] addressed the problem of computing time optimal trajectories for a point robot subject to simple dynamic constraints. The planners proposed in [14][6][9] consider a geometric model of the robot by using discrete search techniques operating in the configuration space. Other interesting contributions studied the problem of simulating the motions of the vehicle along a precomputed path. The kinematic simulator described in [3] turns the problem into a minimization under constraints. The dynamic behavior of the robot can also be predicted by modeling the physics of the interactions between the wheels and the ground[2][5]. As proposed in [5], both geometric and physical models could be combined during the planning phase. However, because of the high computational cost, it is generally preferable to favour a two phases approach which simulates the behaviour of the vehicle along a path precomputed by a geometric planner.

The paper is structured as follows: after a brief overview in section 2 of the planning approach of [14], the contributions of the paper, ie. the extension to an articulated vehicle and the integration of uncertainty constraints, are respectively detailed in sections 3 and 4. Finally, some experimental results are presented in the last section.

*This work has been done in the framework of the Automatic Planetary Rover project conducted by the French Spatial Agency. It was partially supported by C.N.E.S.

2 Overview of the Planner

Given the models of the terrain surface and of the vehicle geometry, the problem is to find a feasible motion between two configurations, while respecting a set of geometric constraints related to the safeness of the motion. Typical *validity constraints* are:

- stability of the vehicle.
- collision avoidance of terrain irregularities.
- mechanical constraints expressing that the contact between the wheels and the terrain can be maintained without exceeding the limits of the articulations.

The terrain is described by ruled surface patches defined from an elevation map in z associated with a regular grid in (x, y). The robot has an articulated chassis with passive joints allowing to adapt its shape to the terrain irregularities. Its placement $\mathbf{p} = (\mathbf{q}, \mathbf{r}(\mathbf{q}))$ is defined by a configuration vector $\mathbf{q} = (x, y, \theta)$ specifying the horizontal position/heading, and by a vector $\mathbf{r}(\mathbf{q})$ of the n parameters associated to the values of the joints. The dependancies between the parameters of $\mathbf{r}$ and the configuration $\mathbf{q}$ come from the contact relations between the wheels and the terrain. The search space is thus reduced to the configuration space $CS = (x, y, \theta)$ and we denote by C_{free} the subset of CS where the validity constraints are verified. The motion planning problem can be classicaly formulated as the problem of finding a path connecting two given configurations and lying in C_{free}. Determining the validity of a configuration $\mathbf{q}$ however requires to compute its associated placement $(\mathbf{q}, \mathbf{r}(\mathbf{q}))$. The next section details an efficient algorithm for solving this problem for a robot composed of several articulated axles.

The planning approach used in [14] builds incrementally a graph of discrete configurations that can be reached from the initial position by applying sequences of discrete controls during a short time interval. Typical controls consist in driving forward or backwards with a null or a maximal angular velocity. Each arc of the graph corresponds to a trajectory portion computed for a given control. Only the arcs verifying the placement constraints mentionned above are considered during the search. In order to limit the size of the graph, the configuration space is initially decomposed into an array of small cuboid cells. This array is used during the search to keep track of small CS-regions which have already been crossed by some trajectory. The configurations generated into a visited cell are discarded and therefore, one node is at most generated in each cell.

The A^* algorithm used to explore the graph returns a minimum cost trajectory which realizes a compromise between the distance crossed by the vehicle, the terrain difficulty along the path and a small number of maneuvers. The heuristic function used to efficiently guide the search requires a preprocessing step: a cost bitmap is first computed by evaluating for each patch of the terrain model, the slope and the roughness of a circular domain centered at this position, with a radius related to the size of the robot. A potential bitmap is then computed by using classical wave propagation techniques which integrate the cost across the bitmap, starting with a null potential at the goal position of the robot and propagating in priority the lowest cost pixels. During the planning step, the heuristic $H(\mathbf{q})$ is computed, as in [1], by a linear combination of the potential evaluated at two control points of the robot.

3 Application to an Articulated Robot

3.1 Model of the robot

Let us consider a three-axle articulated robot (Fig.1). 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 horizontal $\vec{x}$-axis.

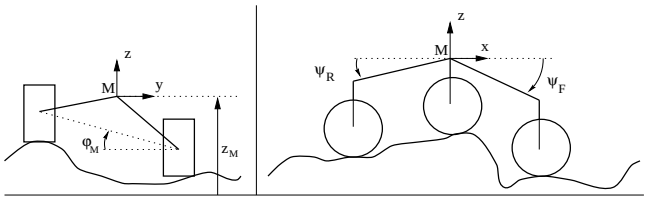

Figure 1: Model of the robot

For this locomotion system, $\mathbf{q} = (x_M, y_M, \theta)$ 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.

3.2 The Placement Algorithm

As mentionned earlier, the role of this 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. Finally, the last subsection describes a preprocessing step which allows to significantly improve the efficiency of the algorithm when used by the planner.

3.2.1 Case 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[1]. Let $dist = F(\varphi)$ be the distance from W_l to the

[1] afterwards, the elevation of its center is easily deduced

terrain when W_r is put in contact. Then we have:

$$F(\varphi) = 2l \sin \varphi - (z_l(\varphi) - z_r(\varphi)) \quad (1)$$

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. 2). The placement is obtained for the solution φ_{sol} of the equation $F(\varphi) = 0$. This equation cannot be solved analyt-

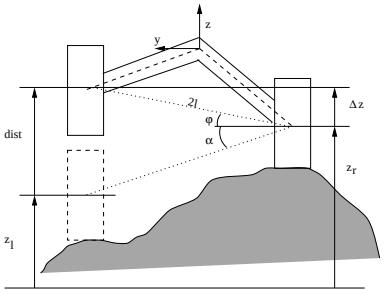

Figure 2: Placement of an axle

ically since $z_l(\varphi), z_r(\varphi)$ closely depend on the terrain geometry. However one can remark that for the case of a flat terrain having a slope α, we would have:

$$z_l(\varphi) - z_r(\varphi) = \cos \varphi \tan \alpha \quad (2)$$

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 step i, evaluating $z_l(\varphi) - z_r(\varphi)$ for a value φ_i and applying Eq. 2, allows to get the slope α which would be the solution for a planar approximation of the terrain. The next value φ_{i+1} is computed from φ_i and α.

begin
$i = 0$; $\varphi_0 = 0$
$dist_0 = F(\varphi_0)$
while $dist_i > \epsilon$
1. estimate of φ_{i+1} as if the terrain was flat: determination of α (cf fig.2) and $dist_\alpha = F(\alpha)$
2. computation of φ_{i+1}: the line passing through $(\alpha, F(\alpha))$ and $(\varphi_i, F(\varphi_i))$ cuts the horizontal axis $(F(\varphi) = 0)$ at φ_{i+1}. $dist_{i+1} = F(\varphi_{i+1})$
3. $i = i + 1$

$\varphi_{sol} = \varphi_i$
end

Tests performed 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.

3.2.2 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 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 additionnal constraint. Consider the case of the front axle[2]. 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.

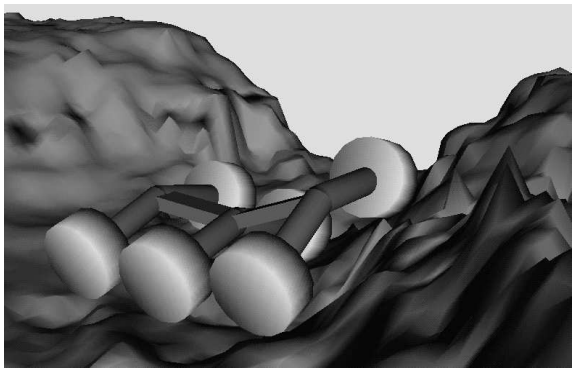

Figure 3: Example of placement

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 of the link with the middle axle: α is determined from z, z_M and the distance between both axles. Then we compute ψ_{i+1} such that the line passing through $(\alpha, G(\alpha))$ and $(\psi_i, G(\psi_i))$ cuts the horizontal axis $(G(\psi) = 0)$ at ψ_{i+1}. 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 initialisation and twice for each step of the algorithm. Again the solution is generally found in only one iteration. Figure 3 shows an example of placement computed by the algorithm.

3.2.3 Placement precomputation

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 computationnal cost of the placement algorithm when used by the planner. It consists in slicing the orientation parameter, and in computing two surfaces denoted $\mathcal{S}_\varphi(x, y)$

[2] the placement of the rear axle is exactly the same

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_{axle}) associated with a regular grid in (x,y). They are computed by applying the function **Place_axle** at every point of the grid [3].

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: $\varphi = (\varphi_{i+1} - \varphi_i)\frac{\theta-\theta_i}{\theta_{i+1}-\theta_i} + \varphi_i$

3.3 Validity constraints

These constraints are checked all along the elementary motions generated during the search for limiting the exploration to the C_{free} regions.

3.3.1 Stability

The constraint used to check that some static stability conditions (eg. 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. 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:

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

3.3.2 Mechanical constraints

These constraints 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] to minimize the number of steps, the angle φ is initialized with the value computed for the previous point of the grid.

3.3.3 Collision

The last validity constraint considered in the current implementation guarantees that, except the wheels, the other parts of the locomotion system do not collide with the terrain irregularities. The algorithm described in [14] allows to efficiently check the collision between polygonal faces and the terrain surface. Applying this algorithm to eight rectangular faces is sufficient to detect the collisions of the robot's axles/bodies. Each axle is represented by two faces, each being placed under the semi-axle and tangent to it (see Fig.4). Only the lower faces of the two bodies are considered by the collision checker.

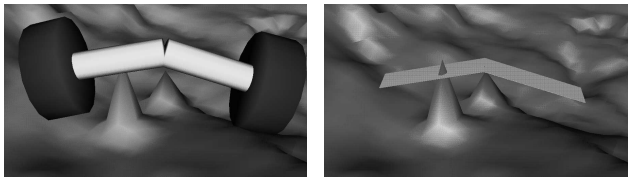

Figure 4: Collision of an axle with the terrain

4 Uncertainty Constraints

Dealing with uncertainties constitutes an important issue of motion planning [4] since robots operating in real world setting are faced to several sources of uncertainties arising from inaccurate models of the environment, control errors and limited sensing accuracy. Consequently, the use of planners which do not explicitly consider uncertainties is limited to some simple situations where the errors remain small with respect to the task tolerances. This is generally not the case in the context of outdoor navigation. First, the terrain model is incrementally built from sensors data and the errors in some areas can be rather important. The complex physical interactions between the wheels and the terrain also result into possibly large control errors, and the robot does not precisely execute the planned motions. This section shows how to improve the robustness of the planner by the introduction of uncertainty constraints in the algorithms described above.

4.1 Uncertainties of the Terrain Model

The uncertainties can be propagated through the different steps of the terrain modelling 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 enveloppes obtained from the minimal/maximal elevations. The problem is to guarantee that the validity constraints imposed to the

[4] See [10] for an overview of related work

planner are satisfied for whatever terrain lying between both enveloppes.

4.1.1 Robot placement

The placement of the robot is now defined by a vector $(\mathbf{q}, \Delta\mathbf{r}(q))$ where $\Delta\mathbf{r}$ represents, at configuration $\mathbf{q}$, the possible intervals for the joints values. These intervals are computed by applying the algorithms described in §3.2 for the wheels placed either on the lower or upper enveloppes:

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

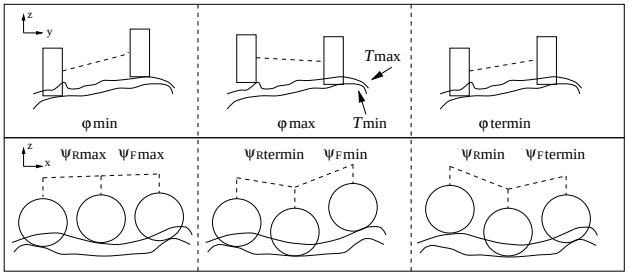

Figure 5: Placement intervals

For each slice θ, the preprocessing step is applied four times to produce the surfaces $\mathcal{S}_{zmin}$,$\mathcal{S}_{zmax}$ and $\mathcal{S}_{\varphi min}$,$\mathcal{S}_{\varphi max}$ which represent the bounds on the axle 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.

4.1.2 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 $\varphi_{<axle>}$:

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

The minimal 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 $\mathcal{T}_{min}$ (cf. Fig. 5). Then we have:

$$\psi_{StabMin} < \frac{\psi_{Rmin} - \psi_{Ftermin}}{2}$$

$$\frac{\psi_{Rtermin} - \psi_{Fmin}}{2} < \psi_{StabMax}$$

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

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

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

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

$$\psi_{Min} < \psi_{Fmin} + \psi_{Rmin} + \pi$$

$$\psi_{Fmax} + \psi_{Rmax} + \pi < \psi_{Max}$$

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 $\mathcal{T}_{max}$.

4.2 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 neighbourhood $\mathcal{N}(\mathbf{q})$ induced by two parameters $(\Delta l, \Delta\theta)$: at any $\mathbf{q}$, the robot may be translated by Δl along the segment normal to the trajectory, with an orientation $\theta \pm \Delta\theta$ (see Fig. 6). The free configuration space is now defined as $CS'_{free} = \{\mathbf{q} \in CS \,/\, \mathcal{N}(\mathbf{q}) \subset CS_{free}\}$. A configuration $\mathbf{q}$ is declared valid, only if all the configurations of its neighbourhood satisfy the validity constraints.

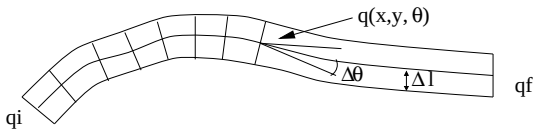

Figure 6: The neighbourhood $\mathcal{N}(\mathbf{q})$

4.2.1 Robot placement

For the placement, we have to compute the vector interval $\Delta\mathbf{r}(\mathbf{q})$ giving the limits of the parameter values for any configuration of $\mathcal{N}(\mathbf{q})$. The method proposed to solve this problem consists in transforming the position uncertainty into uncertainties on the elevations of the terrain model.

For a given neighbourhood $\mathcal{N}(\mathbf{q})$, the wheel elevations belong to an interval defined by the extreme elevations, for any horizontal position of the wheel induced by $\mathcal{N}(\mathbf{q})$. As illustrated by Figure 7, this position domain can be approximated by a rectangular region $\mathcal{R}$. Its size depends on $(\Delta l, \Delta\theta)$ and on the distance from the wheel center to the origin M of the robot's frame.

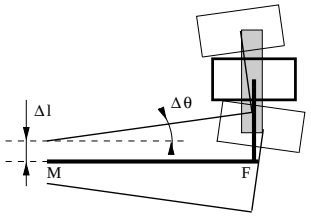

Figure 7: The region $\mathcal{R}$

Let us consider, for a given slice θ, the two surfaces denoted $\mathcal{T}'_{min}(x,y)$ and $\mathcal{T}'_{max}(x,y)$. These surfaces represent the extreme terrain elevations, for any position of the wheel inside the region $\mathcal{R}$ placed at position (x,y) with the orientation θ. Taking these surfaces [5] instead of the terrain enveloppes $\mathcal{T}_{min}$ and $\mathcal{T}_{max}$ (cf. section 4.1) for the preprocessing step of the placement algorithm, allows to compute the placement vector $\Delta\mathbf{r}$ by the method proposed in section 4.1.1 for the case of uncertainty in the terrain model.

Although it may overconstraint the problem when the terrain slope is important, ie. the produced placement intervals are larger than the expected ones, the advantage of this method is to handle position uncertainty with a low additionnal cost. Also, it allows to fuse both types of model/position errors in a very simple way, ie. by the union of the elevation intervals resulting from each source of uncertainty. The surfaces $\mathcal{T}'_{min}$ are computed from the lower terrain enveloppe $\mathcal{T}_{min}$ and conversely for maximal case.

4.2.2 Validity constraints

The stability and mechanical constraints are checked as for the case of the model uncertainty. The only adaptation concerns the faces considered by the collision checker. We first compute the volume swept by the lower faces for the placement domain $\Delta\mathbf{r}$ computed at configuration $\mathbf{q}$. The collisions are then tested between the lower faces of these volumes and the terrain surface (or with the upper envelope $\mathcal{T}_{max}$ if the model uncertainties are also considered).

[5] Only two surfaces are used for all the wheels. They are computed for the maximal $\mathcal{R}$ regions of the wheels

5 Experimental results

The algorithms have been implemented in C on a Silicon graphics indigo workstation with an R4000 processor, and the planner has been tested onto a number of different terrains. We describe below some of the experimental results obtained during these tests.

The terrain of the first example is represented by a 100×75 elevation map. For this example, Figure 8 illustrates some intermediate data computed during the preprocessing step: the cost and potential used to guide the search (cf. Fig. 8-b) and surfaces $\mathcal{S}_\varphi$ computed for θ-slices of 0 and 90 degrees (cf. Fig. 8-c). Finally, the trajectory computed by the planner is shown in Fig. 8-d. For this example, we also give some indications concerning the performances of the algorithms described in the paper.

Preprocessing step: The CPU times and memory requirements of this step are reported in the following Table. The left/right columns respectively correspond to the case without/with model/control uncertainties. The first line corresponds to the computation the parameters surfaces for ten θ-slices covering the interval $[0, \pi[$. The second line includes the cost/potential computation, and also the hierarchical terrain model used by the collision checker.

	without uncert.		*with uncert.*	
	time(sec)	*mem.(kb)*	*time*	*mem.*
slices	3.840	305	12.890	756
total	8.040	1383	17.200	1835

Planning step: During the search, most of the time is spent in computing the placements and checking the validity constraints. The following Table the respective CPU times of these algorithms. The computations were repeated 10.000 times in order to get significative times. One can notice the efficiency of the proposed algorithms, and also the additionnal cost when the uncertainties are considered.

	time (sec.)	
	without uncert.	*with uncert.*
placement	1.750	5.720
validity test	0.760	0.730
collision test	2.550	4.620
node expansion	5.570	11.660

After a preprocessing step of 8 sec., the trajectory of the first example (Fig. 8-d) was found by the planner after less than one second.

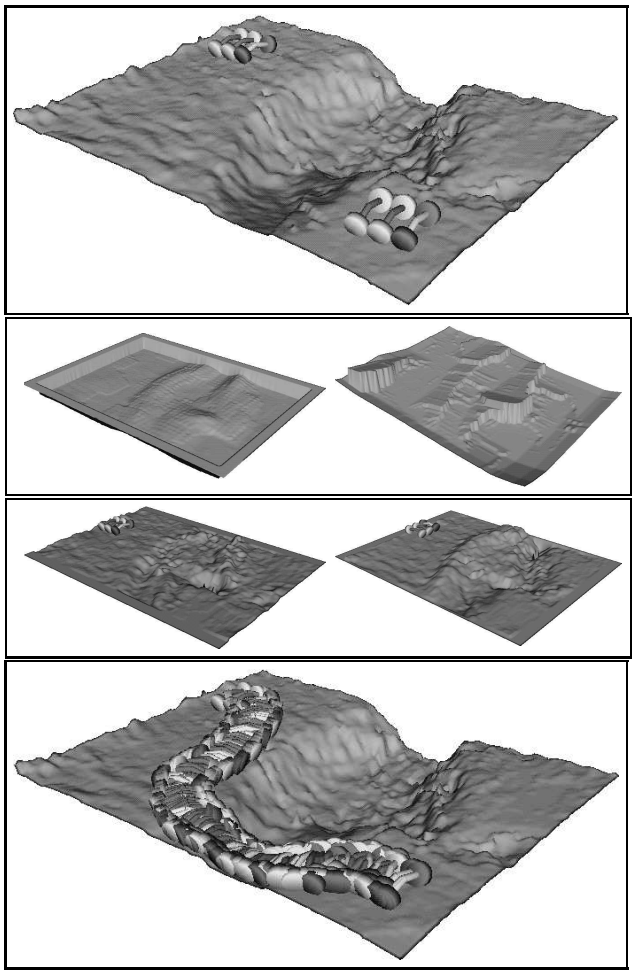

Figure 8: **a/** the terrain model and the initial/final configurations. **b/** the associated cost and potential functions. **c/** the $\mathcal{S}_\varphi$ surfaces for $\theta = 0$ and $90deg.$. **d/** the trajectory found by the planner

Uncertainties: The second example illustrates the influence of the uncertainties on the solutions produced by the planner. A first trajectory (Fig. 9-b) was generated in $2.5sec.$ without considering the uncertainties. Then we introduced the errors shown by Figure 9.c in the terrain model of Figure 9.a, and we imposed a $\pm 30cm$ width for the safety channel around the trajectory. The first trajectory traversing the canyon is not anymore safe, and the solution found by the planner in $14.5sec.$ is shown in the last Figure.

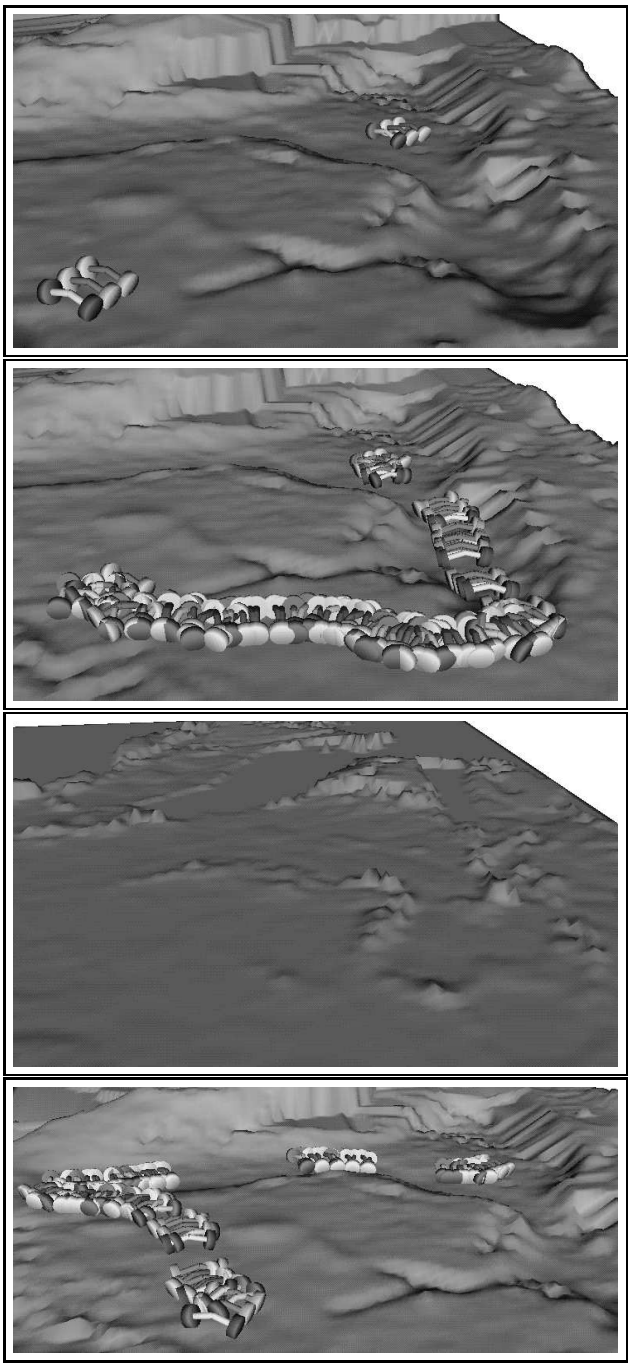

Figure 9: **a/** the terrain model and the initial/final configurations. **b/** a first trajectory computed without considering uncertainties. **c/** the uncertainties of the terrain model. **d/** the trajectory computed in presence of model/position uncertainties

6 Conclusion

In this paper, we considered the difficult problem of planning safe motions for an articulated vehicle moving on rough terrains. The overall approach essentially consists in using discrete search techniques operating in the configuration space of the robot, and in evaluating the validity of elementary motions. This approach is made possible by the use of efficient algorithms to compute the robot's placements and to check their validity. We also proposed techniques which improve the robustness of the solution in presence of model and control errors. Experiments conducted with this planner on various terrains have demonstrated its capacity to solve rather difficult problems in a very reasonable amount of time.

Current work concerns the introduction into the planner of constraints related to the visibility and to the selection of landmarks which could be tracked at execution in order to decrease errors in localization. We also plan to integrate this planner into the outdoor navigation approach [4] currently developped at LAAS within the framework of the "EDEN" experiment, and to validate its efficacy in real world settings.

References

[1] J. Barraquand and J.C. Latombe. Robot Motion Planning: A distributed Representation Approach. *The International Journal of Robotics Research*, 10(5), 1991.

[2] F. Ben Amar, P. Bidaud and F. Ben Ouesdou. Modelling and Motion Planning of Planetary Vehicles. In *Revue d'Intelligence Artificielle, Vol. 7, n° 4, p. 451-464* 1993.

[3] C. Bellier, C. 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)*, July 1993.

[4] R. Chatila, S. Lacroix, T. Siméon, M. Herrb. Planetary exploration by a mobile robot: mission teleprogramming and autonomous navigation. In *Autonomous Robots Journal, Vol. 2, n° 4, p 333-344*, 1995.

[5] M. Cherif, C. Laugier. Dealing with Vehicle/Terrain Interactions when Planning the Motions of a Rover. In *IEEE International Conference on Intelligent Robots and Systems, Munich (Germany)*, September 1994.

[6] B. Dacre-wright and T. Siméon. Free Space Representation for a Mobile Robot moving on a Rough Terrain. In *IEEE International Conference on Robotics and Automation, Atlanta (USA)*, 1993.

[7] M. Hebert. Pixel-based range processing for autonomous driving. In *IEEE International Conference on Robotics and Automation, San Diego (USA)*, 1994.

[8] E. Krotkov, M. Hebert and R. Simmons. Stereo Perception and Dead Reckoning for a Prototype Lunar Rover. *Autonomous Robots Journal, Vol. 2, n° 4, p 313-331*, 1995.

[9] T. Kubota, I. Nakatani and T. Yoshimitsu. Path Planning for Planetary Rover based on Traversability Probability. *Seventh International Conference on Advanced Robotics, Sant Feliu (Spain)* September 1995.

[10] J.C Latombe. *Robot Motion Planning.* Kluwer Pub., 1990.

[11] A. Liegeois, C. Moignard. Minimum-time Motion Planning for Mobile Robot on Uneven Terrains. In *Robotic Systems, p. 271-278*, Kluwer 1992.

[12] F. Nashashibi, P. Fillatreau, B. Dacre-Wright, and T. Simeon. 3d autonomous navigation in a natural environment. In *IEEE International Conference on Robotics and Automation, San Diego (USA)*, May 1994.

[13] Z. Shiller and J.C. Chen. Optimal motion planning of autonomous vehicles in three dimensional terrains. In *IEEE International Conference on Robotics and Automation, Cincinnati (USA)*, 1990.

[14] T. Simeon and B. Dacre-Wright. A practical motion planner for all-terrain mobile robots. In *IEEE International Conference on Intelligent Robots and Systems, Yokohama (Japan)*, July 1993.

[15] C. Thorpe, M. Hebert, T. Kanade, and S. Shafer. Toward autonomous driving : the cmu navlab. part i : Perception. *IEEE Expert*, 6(4), August 1991.

[16] C.R. Weisbin, M. Montenerlo, and W. Whittaker. Evolving directions in nasa's planetary rover requirements end technology. In *Missions, Technologies and Design of Planetary Mobile Vehicules. Centre National d'Etudes Spatiales, France*, Sept 1992.

[17] B. Wilcox and D. Gennery. A Mars Rover for the 1990's. *Journal of the British Interplanetary Society*, 40:484–488, 1987.