Motion generation for a rover on rough terrains

David Bonnafous, Simon Lacroix and Thierry Siméon
LAAS/CNRS
7, Ave du Colonel Roche
F-31077 TOULOUSE Cedex 4
{firstname.name@laas.fr}

Abstract

This article presents an algorithm that determines safe motions for an articulated rover on rough terrains. It relies on the evaluation of a set of elementary trajectories on a digital elevation map built as the rover moves. The algorithm relies on the explicit computation of geometric constraints on the rover chassis. It has been integrated within a continuously running navigation loop on board the robot Lama, and tested under various terrain conditions.

1. Introduction

Among the various issues raised by autonomous rover navigation in unstructured environments, traversing rough areas is one of the most challenging. Indeed, such areas bring forth various difficulties to each of the perception, trajectory evaluation and trajectory execution control functions:

- To catch the surface geometry of the area to traverse, a fine model of the environment is required;
- To select feasible motions, geometric and kinematics constraints on the rover have to be explicitly taken into account;
- Finally, ensuring the proper execution of the selected motions is a much more difficult issue than on smooth terrains.

This paper focuses on the second point: it presents an algorithm that determines safe and efficient motions for a rover on rough terrains. The algorithm consists in evaluating a set of elementary trajectories (circle arcs) on a digital elevation map continuously updated as the rover moves, taking into account geometrical constraints on the robot chassis. The approach has been successfully integrated within a simple autonomous navigation loop on board the robot Lama: tens of experiments have been achieved, in which the robot is asked to reach a few tens meters distant goal in an initially unknown terrain (figure 1).

Related work: Pioneer work on planning motions on rough terrains dates back to the late 80's, and has essentially been considered within the context of planetary exploration robotics. In [1], an algorithm that plan minimal time paths on a surface represented by B-spline patches is presented: it insists on the consideration of the *vehicle dynamics*. Work presented in [2, 3, 4] extend this approach, by considering an articulated chassis mode and explicit vehicle/terrain interactions. In our lab, we also contributed to the problem [5, 6], and developed a planner that finds trajectories to reach a distant goal for an articulated chassis. Latest developments included the consideration of the terrain modeling uncertainties and the necessity for the rover to localize itself [7]. To our knowledge, all these work have only been tested in simulated environments.

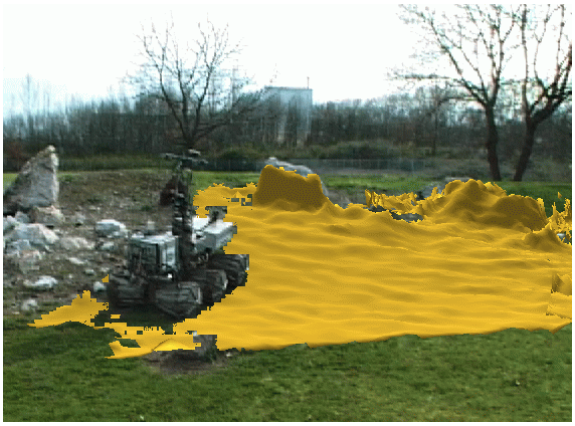


Figure 1: *The robot Lama during an autonomous rough terrain traverse. Here, the digital elevation map built during the traverse is surimposed on the image.*

More recently, various simpler approaches have been presented and effectively demonstrated on board rovers. Most of these approaches rely on the segmentation of the terrain model into two or more terrain navigation classes (flat, obstacle, slope) [8], or on the estimate of the terrain traversability by fitting planes over terrain patches [9], sometimes using fuzzy rules for this segmentation and for elementary motion generation [10]. In these cases, demonstrations on flat ground with sparse rocks are presented.

Outline: This paper is articulated as follows: the next section gives an overview of the three components involved in the approach, and presents the principle of the motion gen-

eration algorithm. Section 3 is dedicated to the determination of the chassis internal configuration on a given position on the terrain model. Section 4 depicts how the optimal circle arc is chosen, on the basis of a *risk* defined by a discrete set of configurations evaluated along the arc, a of an *interest* related to the goal to reach position. Some experimental results of the whole approach are presented in section 5.

2 Overview of the approach

The navigation approach to reach a given goal is sketched in figure 2. It is a very simple instance of a "perceive - decide" paradigm (often applied when dealing with rather easy terrains [11, 12]): from a pair of stereo images, a 3D image is computed using a pixel correlation-based algorithm. This 3D image is merged in a digital elevation map (DEM) of the terrain, and every time this map is updated, a set of elementary trajectories is evaluated: the best trajectory selected defines the motion commands to send to the rover, and the process is re-iterated until the goal is reached.

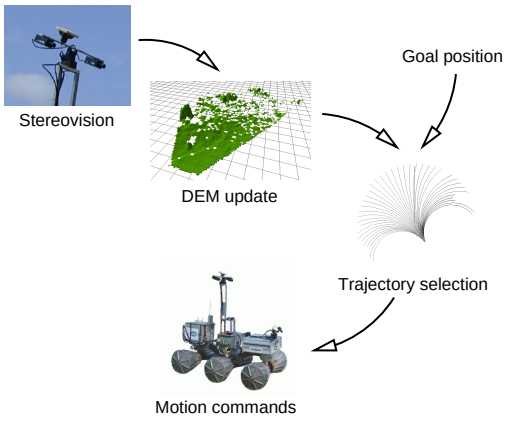


Figure 2: Principle of the navigation loop

Digital elevation map building: Digital elevation maps are a popular way to represent the environment of a rover: their structure is easy to manage, and they can represent quite faithfully the terrain surface geometry. The main difficulty to update such maps comes from the uncertainties on the 3D input data; however, a realistic model can be easily built by computing the mean elevation of the data points on the DEM grid cells.

One of the problems when dealing with rough terrains is the large number of occlusions caused by the terrain irregularities and the data resolution decrease on the ground (figure 3). To palliate this, the DEM is continuously updated as the robot navigates, which requires the knowledge of the robot 3D position, with a precision of the order of the grid size. Odometry being not reliable on rough terrains, in our system the 3D position estimate is provided by a visual motion estimation technique [13], which is running in parallel of the navigation loop.

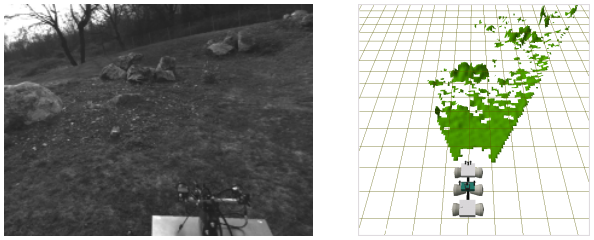
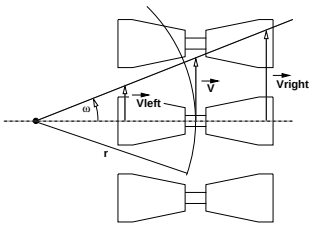


Figure 3: *A digital elevation map built with a single stereo image pair: several areas are unperceived because of the terrain irregularities and the drastic data resolution decrease with the distance.*

Motion control: The six wheels of Lama being non-orientable, rotations are produced according to a skid-steering scheme: each of the three wheel of one side of the robot are given the same rotation speed command, and the difference of the left and right speed command induces a rotation (figure 4). Thanks to a precise optical encoder and to a powerful motor/gear set, six simple PD controllers ensure a tight control of each wheel rotation speed, *whatever the terrain characteristics are*. Because of unpredictable ground/wheel frictions, the information returned by the wheel encoders do not allow a faithful estimation of the rotation speed: for that purpose, a fiber-optic gyrometer is used. The locomotion layer is therefore able to handle linear and angular rover velocities (v, ω) commands, by sending the six PD controllers commands at $40Hz$.

Figure 4: *Motion control of Lama. A difference in the lateral wheel speeds induces an angular velocity, producing a circle trajectory with a radius r. Only the middle axle contributes to the rotation. It can be shown that the speeds to be applied to the front and rear axles wheels so that they do not oppose too much against this rotation are the same $\vec{V}_{left}$ and $\vec{V}_{right}$.*



Motion generation: The heart of the system is the motion generation algorithm: it consists in evaluating the *risk* and the *interest* of a set of elementary trajectories (figure 5), the one with the smallest *risk/interest* ratio being selected. The considered trajectories are circle arcs, as they correspond to a control easily handled by the locomotion layer.

To evaluate a circle arc, a discrete set of configuration along it are considered: a risk is defined for each configuration by checking constraints on the internal chassis configuration and on the robot attitude (section 3). The integration of the risks computed along an arc and the consideration of its interest, in terms of getting closer to the goal, are then used to select the optimal one (section 4).

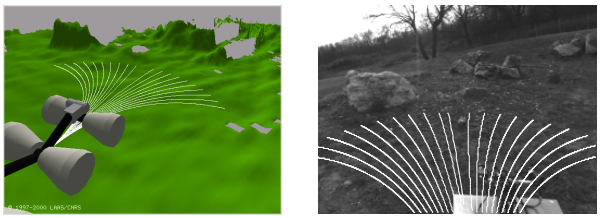


Figure 5: *The set of circle arcs evaluated at each step of the loop.*

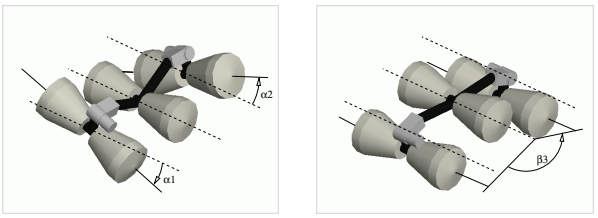


Figure 7: *The three passive articulations of the chassis.*

3 Chassis pose and configuration determination

The ability to predict the robot pose and its chassis internal configuration for a given position (x, y, θ) on the digital elevation map is a key point in our algorithm, as it produces the basic parameters that will be used to select feasible and safe motions. We describe here the `placeRobot` function, that fulfills this requirement.

3.1 Chassis geometry

Lama[1] is a Marsokhod articulated chassis [14]. Its length can be actively controlled by modifying the distance between the axles (figure 3.1), thus enabling the possibility to move in a *peristaltic* mode. However, for the algorithms and experiments presented in this paper, the inter-axle distance is maintained fixed.

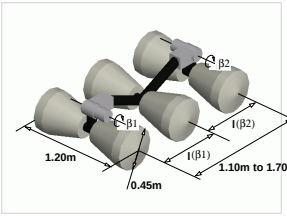


Figure 6: *Geometry of Lama locomotion structure. Depending on the angular values of β_1 and β_2, the length of the chassis can be adapted. In its "nominal" configuration, the chassis overall length is 1.85 meter.*

The chassis is also passively articulated, as shown in figure 7), which gives the robot high obstacle climbing skills. The mechanical variations of these three parameters, whose values is provided to the locomotion control layer by angular encoders, are bounded by the following values :

$$-35° < \alpha_{1,2} < 35° \quad (1)$$

$$110° < \beta_3 < 230° \quad (2)$$

The robot *attitude* is defined by the attitude of its middle axle with respect to the vertical: the two pitch and roll angle ψ_m and ϕ_m are measured thanks to a two-axis inclinometer.

3.2 Robot placement on the DEM

Given a position (x, y, θ) on the digital elevation map, the robot placement function `placeRobot` provides the five angular values $(\phi_m, \psi_m, \alpha_1, \alpha_2, \beta_3)$ that entirely define the chassis configuration parameters.

This function relies on the computation of a single axle placement (function `placeAxle`), which itself make iterative calls to the `placeWheel` function (figure 8). Similarly, once the middle axle is placed on the DEM, several iterative calls to the `placeAxle` function are required to place the front and rear axle, to finally obtain the five configurations angles.

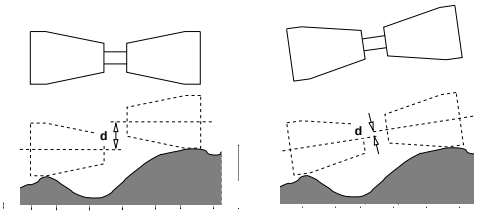


Figure 8: *Principle of the* `placeAxle` *function: the axle orientation being set to 0, two calls to the* `placeWheel` *function define the elevation of the wheels for this orientation (left). The distance* `d` *between the two wheels rotation axes define a new axle orientation (right), and the process is reiterated until* `d` *stabilizes itself around* `0`.

The implementation of the function `placeWheel` is of crucial importance, at it is called a lot of times to compute the robot five configurations angles. For that purpose, we have tabulated the elevations of the conic wheel profile in a rectangular array, with a resolution twice as fine as the DEM resolution: the minimum distance between this array elevations and the DEM elevations on the projected rectangle gives the elevation of the wheel on the DEM.

However, the `placeAxle` function is quite time consuming: not only it makes successive calls to the `placeWheel` function, but also at each iteration the tabulated wheel profile values have to be rotated according to the current axle orientation. Table 1 gives the number of iterations that have been required for more than 300,000 calls to the `placeAxle` function: as one can see, in more than 90% of the cases, only one iteration is required. We therefore choose to only execute a single iteration in the `placeAxle` function for each of the three axles: on a Sun Ultra 10 SparcStation, a call to the `placeRobot` function, which determines the 5 robot configuration parameters, takes less than $3ms$.

[1] The chassis of Lama is owned by Alcatel and is lent to LAAS. All the equipments has been conceived and developed at LAAS.

Nb iterations	Nb placements
1	293780
2	15694
3	4024
4	1521
5	867
6	547
7	382
8	277
9	197

Table 1: *Statistics on the number of iterations required by the function* `placeAxle`.

3.3 Experimental validation

To check the validity of the robot placement function, we made some experimental trials, in which the robot was manually driven along a straight line, roving over rocks of various sizes (figure 9).

Figure 9: *A predicted placement of the robot on the DEM (left), and the corresponding placement in reality (right).*

Figure 10 shows a comparison between the robot configuration angles predicted by the placement function on the computed DEM and the angles measured on board the robot: no significant differences can be noticed.

4 Evaluation of the arcs

Every time the DEM is updated, our algorithm evaluates a set of circle arcs to select the "optimal" one. In this section, we explain how this evaluation is made on the basis of the placement function, and the search strategy that selects the arc to execute.

4.1 The "danger" of a placement

Besides the internal chassis configuration constraints 1 and 2 mentioned above, we consider two stability constraints on each of the three axles :

$$-\phi_{max} < \phi_{r,m,f} < \phi_{max} \quad (3)$$
$$-\psi_{max} < \psi_{r,m,f} < \psi_{max} \quad (4)$$

where r, m and f respectively stands for the rear, middle and front axles. The pitch and roll angles ψ and ϕ for the

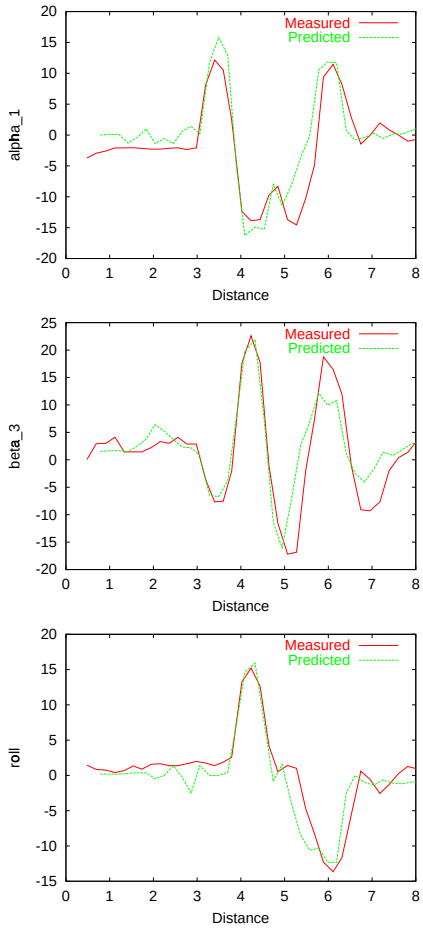


Figure 10: *Comparison between the predicted and measured angles (in degrees), as a function of the curvilinear abscissa on a straight trajectory. From top to bottom: $\alpha_1, \beta_3, \phi_m - \pi$. The images of figure 9 were taken during this trajectory, when the corresponding abscissa was approximately $5m$.*

front and rear axles are easily derived from the five angular values that define the chassis configuration[2]. So in total, a set of 9 bounding constraints are considered: any placement that violates one of these constraints is considered as not valid. We consider that these constraints are all "equivalent", in the sense that none of them can be violated, and that the closer is a value to a bound, the more dangerous is the robot configuration. These constraints are therefore normalized, and we end up with the following constraint set:

$$\mathcal{C} = \{-1.0 < c_i < 1.0\}_{i \in [0,\ldots,9]}$$

An other constraint has also to be considered: indeed, there are always some unperceived areas in the DEM, especially behind sharp obstacles (as one can see on figure 9, just below the left rear wheel). To consider this,

[2]The consideration of a constraint on the pitch angle for an axle might be surprising, but one must not forget that some equipments are mounted on each of the axles.

the `placeWheel` function returns a value comprised in $[0, 1]$, which indicates the proportion of unknown pixels in the DEM in the area defined by the projection of the wheel. We decided that this value should not be greater than 0.5, and the biggest of the 6 values returned by a call to the `placeRobot` function defines a tenth constraint.

Now given a robot position (x, y, θ) on a circle arc, the robot placement function provides the five corresponding configuration angles, to which correspond a point in the 9-dimensional constraint space after normalization. The danger d of a given position (x, y, θ) is expressed by:

$$d_{(x,y,\theta)} = max_{i \in [0,\ldots,10]}(|c_i|)$$

4.2 The cost of a motion

The cost of an elementary motion from the robot position (x_1, y_1, θ_1) to (x_2, y_2, θ_2) integrates the notion of "danger" (or risk) presented above in the following way:

$$C_{1\to 2} = (1.0 + c_1 + \delta c_{1\to 2})\Delta s_{1\to 2}$$

where:

• $\Delta s_{1\to 2}$ is the distance between the two considered positions.

• c_1 is the cost associated to the danger d_1, computed as follows:

$$c_1 = \begin{cases} 0 & \text{if } d_1 < d_{min} \\ 1.0/(1.0 - d_1) & \text{otherwise} \end{cases}$$

The introduction of the threshold d_{min} on d is to reduce the influence of small irregularities of the terrain: in the case of a flat terrain for instance, the robot is said to be in a risk-less flat configuration, and only the goal orientation is considered to select the arc to execute.

• and $\delta c_{1\to 2}$ is a cost associated to the internal configuration change between positions 1 and 2:

$$\delta c_{1\to 2} = \begin{cases} 0 & \text{if } |d_2 - d_1| < \delta_{min} \\ |d_2 - d_1| & \text{otherwise} \end{cases}$$

This parameter favors the trajectories in which the robot internal configuration does not change too much. Again, the introduction of a threshold δ_{min} under which it has no influence is to get rid of the small terrain irregularities.

4.3 Search algorithm

Once the DEM is updated, a set of circle arcs is generated, and each arc is discretized into a discrete set of robot configurations (x, y, θ), which defines a tree structure. Configurations are regularly spaced, and are defined up to a maximal curvilinear abscissa on the arcs, which correspond approximately to twice the distance that the robot is supposed to travel until the DEM is updated again: this gives the robot a kind of "look ahead" behavior, and avoids the selection of trajectories that go too close to obstacles.

The optimal circle arc is determined thanks to an $A^\star$ algorithm, in which the heuristic is the distance of the Dubbins path [15] to reach the goal. The cost to go from the last configuration on a circle arc to the goal is also equal to the Dubbins distance. Thanks to the efficiency of the $A^\star$ algorithm, only a small number of configurations along the arcs are effectively evaluated (practically, a set of 40 circles arcs is considered, and about 20 configurations are defined on each arcs: most of the times only a third or a fourth of this configuration set is evaluated).

5 Results

The whole navigation loop has been integrated on board the robot Lama, within the LAAS Architecture for Autonomy [16]. It proved its efficiency on various kinds of terrains: figures 11 and 12 present two trajectories executed by Lama using this navigation loop. The mean time required to select an optimal arc at each step is approximately $300ms$: therefore, we also use this algorithm to find motions on easy terrains.

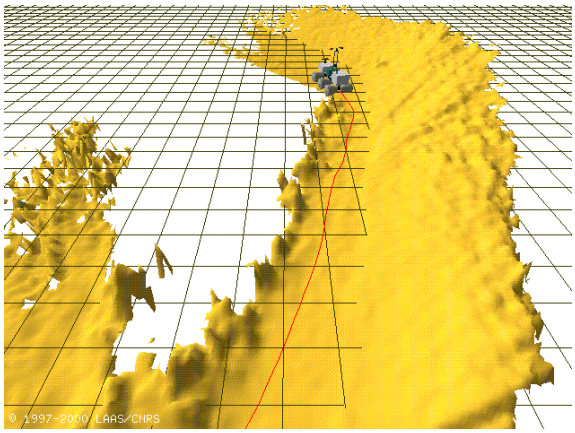


Figure 11: *Lama found its way trough a narrow corridor: 85 iterations of the navigation loop have been executed, the trajectory is about $40m$ long. (Live demonstration in the "Cité de l'espace" museum in Toulouse, Sept. 2000).*

6 Summary and discussion

We described an algorithm that generates elementary motions on rough terrains, which explicitly takes into account the geometric constraints on an articulated chassis. The approach has been integrated on board the robot Lama, and is very often demonstrated in continuously running experiments, where the rover is able to autonomously reach a few tens meters distant goal in initially unknown terrains. This functionality is integrated within a more global autonomous long range navigation system, which manages

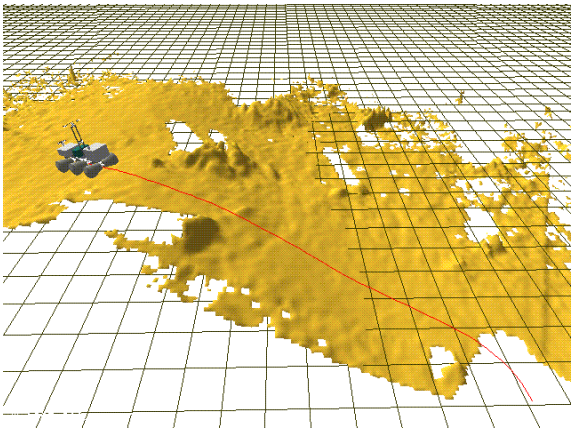


Figure 12: *Lama found a pass to climb a 2 meters high hill: 50 iterations of the navigation loop have been executed, the trajectory is about* $25m$ *long. (Trial in our experimentation site).*

several terrain models, motion modes and localization algorithms [17].

Among the open problems that still have to be tackled, the consideration of realistic dynamic issues is one of the most important. These issues have been modeled and considered in pioneer work related to rough terrain motion planning, but their consideration on board real rovers requires additional work and validation, as shown by recent practical contributions [18, 19].

References

[1] Z. Shiller and Y-R. Gwo. Dynamic motion planning of autonomous vehicles. *IEEE Transactions on Robotics and Automation*, 7(2):241–249, April 1991.

[2] S. Jimenez, A.Luciani, and C. Laugier. Predicting the dynamic behaviour of a planetary rover vehicle using physical modeling. In *International Workshop on Intelligent Robots and Systems, Yokohama (Japan)*. INRIA-LIFIA, 1993.

[3] M. Cherif and C. Laugier. Motion planning of autonomous off-road vehicles under physical interaction constraints. In *IEEE Int. Conf. on Robotics and Automation, Nagoya (Japan)*, pages 1687–1693, May 1995.

[4] 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*, 15(2):202–218, 1999.

[5] T. Simeon and B. Dacre-Wright. A practical motion planner for all-terrain mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, Yokohama (Japon)*, pages 26–30, July 1993.

[6] A. Hait and T. Simeon. Motion planning on rough terrain for an articulated vehicle in presence of uncertainties. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, Osaka (Japan)*, pages 1126–1133, Nov. 1996.

[7] A. Hait, T. Simeon, and M. Taix. Robust motion planning for rough terrain navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, Kyongju (Korea)*, pages 11–16, Oct. 1999.

[8] S. Laubach and J. Burdik. An autonomous sensor-based path-planner for planetary microrovers. In *IEEE International Conference On Robotics and Automation. Detroit, MI (USA)*, May 1999.

[9] S. Singh, R. Simmons, M.F. Smith, A. Stentz, V. Verma, A. Yahja, and K. Schwehr. Recent progress in local and global traversability for planetary rovers. In *IEEE International Conference on Robotics and Automation, San Francisco, Ca (USA)*, 2000.

[10] A. Howard and H. Seraji. Real-time assessment of terrain traversability for autonomous rover navigation. In *IEEE/RSJ International Conference on Intelligent Robots and systems*, pages 58–63. JPL, Nov. 2000.

[11] D. Langer, J. Rosenblatt, and M. Hebert. A behavior-based system for off-road navigation. *IEEE Transactions on Robotics and Automation*, 10(6):776–782, Dec. 1994.

[12] H. Haddad, M. Khatib, S. Lacroix, and R. Chatila. Reactive navigation in outdoor environments using potential fields. In *International Conference on Robotics and Automation, Leuven (Belgium)*, pages 1232–1237, May 1998.

[13] A. Mallet, S. Lacroix, and L. Gallo. Position estimation in outdoor environments using pixel tracking and stereovision. In *IEEE International Conference on Robotics and Automation, San Francisco, Ca (USA)*, pages 3519–3524, April 2000.

[14] A. Kemurdjian, V. Gromov, V. Mishkinyuk, V. Kucherenko, and P. Sologub. Small marsokhod configuration. In *IEEE International Conference on Robotics and Automation, Nice (France)*, pages 165–168, May 1992.

[15] L.E. Dubbins. On curves of minimal length with a constraint on overage curvature and with prescribed initial and terminal positions and tagents. *American Journal of Mathematics*, 79:497–516, 1957.

[16] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. An architecture for autonomy. *Special Issue of the International Journal of Robotics Research on Integrated Architectures for Robot Control and Programming*, 17(4):315–337, April 1998. Rapport LAAS N. 97352, Septembre 1997, 46p.

[17] S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury, M. Herrb, and R. Chatila˙ Autonomous rover navigation on unknown terrains: Functions and integration. In *7th International Symposium on Experimental Robotics, Honolulu, HI (USA)*, Dec. 2000.

[18] G. Andrade-Barosso, F. Ben-Amar, P. Bidaud, and R. Chatila. Modeling robot-soil interaction for planetary rover motion control. In *1998 IEEE/RSJ International Conference on Intelligent Robots aned Systems, Victoria (Canada)*, pages 576–581, Oct. 1998.

[19] S. Farritor, H. Hacot, and S. Dubowsky. Physics-based planning for planetary exploration. In *IEEE International Conference On Robotics and Automation. Leuwen (Belgium)*, pages 278–283, May 1998.