A Practical Motion Planner for All-terrain Mobile Robots

Thierry Simeon, Benoit Dacre-Wright

IEEE Int. Conf. on Intelligent Robots and Systems (IROS'93)
July 1993, Yokohama (Japan)

A Practical Motion Planner for All-terrain Mobile Robots

T. Siméon B. Dacre-Wright

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

Abstract

This paper addresses the motion planning problem for wheeled vehicules moving on rough terrains [1]. First, we formalize the placement problem for the case of a rather complex locomotion system consisting of n wheels attached to the robot body by passive suspensions. We next analyze the geometric and kinematic constraints acting on the placements of the robot. Finally, we present a planning method that computes a safe and feasible path between two given placements of the robot. The approach basically consists in searching a path into a graph built incrementally during the exploration of a discrete 3D configuration space. The algorithms have been implemented and simulation results are reported at the end of the paper.

1 Introduction

There is an increasing interest in developping mobile robots for applications requiring autonomous navigation on natural terrains (planetary exploration [4][5], public safety robotics and other hazardous missions [8] [9]). This paper addresses the path planning problem for such all-terrain mobile robots.

Research in motion planing has been very active over the past decade. Most of the work has addressed the problem of finding a collision-free path for manipulators or for mobile robots moving in planar environments (see [6] for a detailed survey).

However, motion planning is almost inexistent in the litterature for mobile robots moving on non-planar terrains. There already exists all-terrain mobile robots which use simple path generation techniques for their autonomous navigation. These approches use the terrain model to characterize the traversability of small terrain patches (roughness of the patch, preferred crossing direction,...). In [4][10], paths are computed from this characterization by gradient propagation techniques. The AMR robot [8] uses a 2D path-planner similar to the one presented in [2] to plan a path that avoids the patches classified as obstacle.

These simple techniques are certainly sufficient for the case of benign terrains. However, they will fail to find a safe path for a robot moving on a rough terrain: in this case, the binary notion of obstacles and non obstacles regions does not hold anymore; the obstacles depend on the ability of the robot to cross over the irregularities of the terrain and their characterization requires to better formalize the constraints acting on the placement of the robot on the terrain.

We are aware of a very few contributions that consider this new path planning problem. A geometric 3D planner is described in [12] for the very simple case of a 3-wheels robot moving on a polygonal terrain. Related work is reported in [9] for the Intelligent Locomotion of a four tiltable track robot. We recently proposed [3] a free-space structuring algorithm based on the characterization of configuration space regions for which the locomotion architecture guarantees terrain irregularities absorption and stability of the vehicle. Finally, Shiller addressed in [11] the problem of finding a time optimal trajectory for a point robot moving on a terrain. The dynamics of the vehicle is used to take into account tip-over constraints.

In this paper, we present an extension of the planning approach proposed in [12]. This extension allows the planner to deal with more complex (and realistic) locomotion structures. We also describe several algorithms which significantly improve the efficiency of the planner.

2 General statements

2.1 Overview of the Approach

We consider an articulated robot moving on a terrain. The robot consists of a body and n wheels attached to it by passive suspensions. These suspensions allow the wheels to remain in contact with the terrain.

[1] 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. and by the ECC Esprit 3 Program within Project 6546 PROMotion.

However, they also complicate the placement of the robot which results from the interaction between the wheels and the terrain, and from the balance of the suspensions. We consider that a safe trajectory needs to satisfy the following constraints, called **placement constraints**:

(**C1**) the wheels remain in contact with the ground.
(**C2**) the suspensions cannot be stretched beyond some limit length.
(**C3**) 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).
(**C4**) the robot does not collide with the terrain.

Given an initial and a goal placement of the robot, our purpose is to plan a feasible trajectory (with respect to the non-holonomic constraint of the robot) that satisfies these placement constraints. The remaining of this section describes the models used to represent the vehicle and the terrain. In section 3, we give a formal statement of the problem and we define the three dimensional configuration space CS which will be used to search for a solution. Section 4 details several algorihms that are used by the path-planner presented in the last section.

2.2 The Vehicle Model

The body of the robot is modeled by a polyhedron R. We define a local frame $\mathcal{R}_{rob} = (G, \mathbf{u}, \mathbf{v}, \mathbf{w})$, where G is the gravity center of the robot, and $(\mathbf{u}, \mathbf{v}, \mathbf{w})$ are respectively its longitudinal, lateral and vertical axis (see Fig. 1).

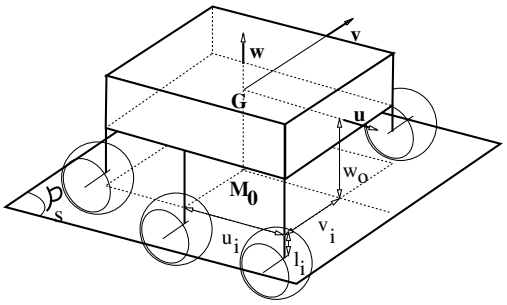

Figure 1: Model of the vehicle

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 $\mathcal{P}_S$ perpendicular to w axis, at height w_0 in the local frame. Then, the coordinates of wheel i are $(u_i, v_i, w_0 + l_i)$ where u_i and v_i are some fixed values related to the geometry of the robot, and l_i is the algebraic extension of the spring ($l_i = 0$ being its natural length). That means that the springs always keep a vertical orientation in the robot's frame. Fig. 1 shows an example of a 6-wheels robot, with the parameters detailed above.

Moreover we consider that two control parameters allow to drive and to steer the vehicle: the linear velocity v_{lin} (mesured along $\mathbf{u}$) and the angular velocity v_{rot} (mesured around $\mathbf{w}$).

2.3 The Terrain Model

The terrain is known through a discrete elevation map, that is the elevation values z on a discrete regular grid in (x, y). For each patch defined by this grid, the terrain is modeled by the non planar face $z = a.x + b.y + c.x.y + d$ which interpolates the four corresponding 3D points of the elevation map. The coefficients (a,b,c,d) associated to the patch (i, j) are directly obtained from the elevations $z_{i,j}, z_{i+1,j}, z_{i+1,j+1}, z_{i,j+1}$ and from the size Δter of the patch. Fig. 2 shows an example of a rough terrain.

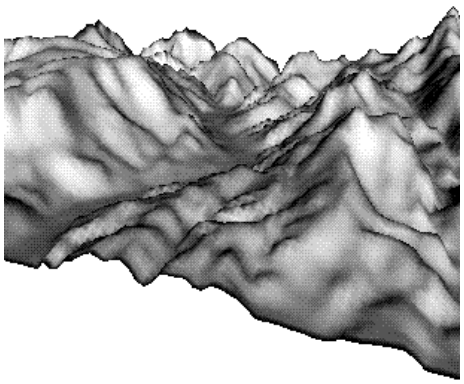

Figure 2: An example of terrain

3 Formulation of the Problem

3.1 The Placement Parameters

A position of the robot's body is defined by the 6-dimensional vector $\mathbf{p} = (x, y, z, \theta, \phi, \psi)$, where (x, y, z) are the coordinates of point G, and (θ, ϕ, ψ) are respectively the horizontal orientation of $\mathbf{u}$ axis, the roll angle and the pitch angle[2]. These parameters define the transform matrix from the robot frame $\mathcal{R}_{rob}$ to the

[2] $|\phi| < \frac{\pi}{2}$, $|\psi| < \frac{\pi}{2}$.

refrence frame $(O, \mathbf{x}, \mathbf{y}, \mathbf{z})$ in which the terrain model is expressed:

$$M(\mathbf{p}) = T(x, y, z)R(\mathbf{w}, \theta)R(\mathbf{v}, \psi)R(\mathbf{u}, \phi).$$

Therefore, a complete placement of the robot is given by $(6 + n)$ independant parameters :

- the 6-dimensional vector $\mathbf{p} = (x, y, z, \theta, \phi, \psi)$ for the position of the robot's body
- the n spring extensions l_i, for the wheel positions.

3.2 The Configuration Space

Nevertheless, the interactions with the terrain constrain these parameters and therefore reduce the dimension of the robot's configuration space. We consider the three dimensional Configuration Space $\mathbf{CS} = \mathbf{R}^2 \times S^1$ induced by the parameters x, y and θ of vector $\mathbf{p}$. To any configuration $\mathbf{q} = (x, y, \theta)$ corresponds a complete placement of the robot. Section 3.3 explains how its $n + 3$ remaining parameters $z(\mathbf{q})$,$\phi(\mathbf{q})$,$\psi(\mathbf{q})$ and the $l_i(\mathbf{q})$ can be obtained.

We define the **admissible configuration space** $CS_{free} \subset CS$ as the set of all configurations $\mathbf{q} \in CS$ for which the associated placement satisfies the placement contraints. The planning problem can be therefore formulated as the problem of computing a path connecting two given configurations and lying in CS_{free}.

3.3 The Placement Problem

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: $\mathcal{E} = \sum_{i=1}^{n} k{l_i}^2$ where k is the stiffness coefficient of the springs.

Moreover, we only need to consider the placements which keep all the wheels in contact with the terrain (**C1**). Let us denote by $L_i(\mathbf{p})$ the function which associates to a given value of the vector $\mathbf{p}$, the value of the spring extension l_i such that the wheel i is in contact (without intersecting) with the terrain (see Fig. 3).

Therefore, the energy can be expressed as a function of the placement $\mathcal{E}(\mathbf{p}) = \sum_{i=1}^{n} kL_i^2(\mathbf{p})$ and for a given $\mathbf{q}$, the remaining parameters of vector $\mathbf{p}$ result from the minimization of this function. Vector $(z(\mathbf{q}), \phi(\mathbf{q}), \psi(\mathbf{q}))$ is the solution of:

$$min_{z,\phi,\psi} \sum_{i=1}^{n} L_i^2(\mathbf{p}) \qquad (1)$$

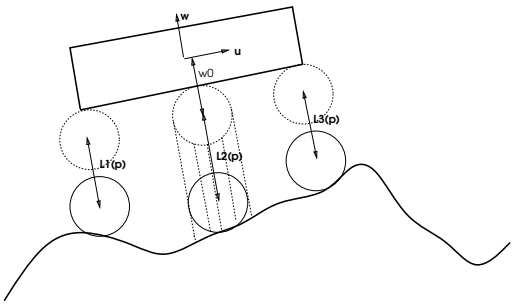

Figure 3: Definition of $L_i(\mathbf{p})$

The spring extensions $l_i(\mathbf{q})$ are obtained from the evaluation of the functions $L_i(\mathbf{p})$ for the computed placement. Section 4.1 details more precisely the algorithms which have been implemented to solve this problem.

3.4 The Nonholonomic Constraint

In order to establish the nonholonomic constraint applying to the motions of the vehicle, let us consider the velocity vector $\mathbf{v}_G$ of the origin G of $\mathcal{R}_{rob}$. Since the axis of the wheels always remain colinear to the lateral axis $\mathbf{v}$ of $\mathcal{R}_{rob}$, the velocity vector $\mathbf{v}_G$ is perpendicular to $\mathbf{v}$:

$$\mathbf{v}_G = v_{lin}.(\alpha\mathbf{u} + \beta\mathbf{w}) \qquad (2)$$

Expressing the coordinates $(\dot{x}, \dot{y}, \dot{z})$ of $\mathbf{v}_G$ in the reference frame yields the following constraint:

$$\dot{x}sin\theta - \dot{y}cos\theta = \frac{\beta}{\alpha}.sin\phi \qquad (3)$$

This expression is similar to the classical nonholonomic contraint $\dot{y} = \dot{x}tan\theta$ established for the case of a mobile robot moving in a two dimensional workspace (see for example [7] [1]). We recall that such a constraint restricts the set of achievable velocities at any configuration to a two dimensional subspace of the tangent space $(\dot{x}, \dot{y}, \dot{\theta})$. However the system remains fully controllable, that is: any two configurations lying in the same connected component of CS_{free} can be connected by a path respecting the kinematic constraint (3).

In our case, the equations of the feasible motions are nevertheless more involved than the one obtained for the planar case. We describe in section 4.3 a simple method to produce paths which satisfy approximatively this constraint.

4 Description of the Algorithms

We describe now the algorithms which have been developped to solve some of the problems mentionned above. These algorithms are used by the path-planner presented in Section 5.

4.1 Placement Computation

Two algorithms have been implemented to obtain a solution to the placement problem. The first one, denoted $PLACE(\mathbf{q})$, is based on standard minimization techniques which compute the minimum of a multivariable function (Eq. 1). This method simply requires to evaluate the functions $L_i(\mathbf{p})$. This evaluation is described in section 4.1.1. The second algorithm, denoted $PLACE_{APP}(\mathbf{p})$ allows to compute much more efficiently an approximate placement. Section 4.1.2 describes this algorithm and discusses the reliability of the solution.

4.1.1 Computing the $L_i(\mathbf{p})$

For a given value of $\mathbf{p}$, the n functions $L_i(\mathbf{p})$ are computed as follow: Let us first consider the case of a ponctual wheel (identified to its center). Its position is determined in $\mathcal{R}_{rob}$ by the vector $(u_i, v_i, w_0 + l_i)$. Thus, the parameter l_i allows to move this point along the $\mathbf{w}$ axis of $\mathcal{R}_{rob}$. This motion corresponds to a line parametrized in the reference frame by the point $(x_i, y_i, z_i, 1)^t = M(\mathbf{p})(u_i, v_i, w_0, 1)^t$ and the direction $\mathbf{n} = M(\mathbf{p})(0, 0, 1, 0)^t$. Let $z = f(x, y)$ denote the analytic expression of the terrain surface. A positive (resp. negative) value of $z_i - f(x_i, y_i)$ means that for a null value of l_i, the wheel lies over (resp. under) the terrain. Therefore the value of $L_i(\mathbf{p})$ can be simply determined from the computation of the first intersection point between the terrain model and the half-line issued from this point $(x_i, y_i, z_i)^t$ in direction $-\mathbf{n}$ (resp. in direction $\mathbf{n}$). The same algorithm can be extented to account for the shape of the wheels by considering a discrete number of "control points" placed on its surface. In this case, $L_i(\mathbf{p})$ corresponds to the minimum of the values obtained for each of the control points.

4.1.2 Approximate Placement

The method consists in applying iteratively a least square algorithm to improve an initial estimation [3] of the parameters z,ϕ and ψ. Each iteration is aimed to decrease the value of $\mathcal{E}(\mathbf{p})$. Let z_k,ϕ_k and ψ_k denote the value of the placement parameters at the beginning of iteration k. For the corresponding placement denoted $\mathbf{p}_k$, the algorithm $L_i(\mathbf{q})$ described in the previous section allows to compute the value $\mathcal{E}_k = \mathcal{E}(\mathbf{p}_k)$ and the n contact points P_i between the wheels and the terrain. The least square method is then used to obtain the equation of the plane which minimizes the quadratic mean of the distances to the points P_i.

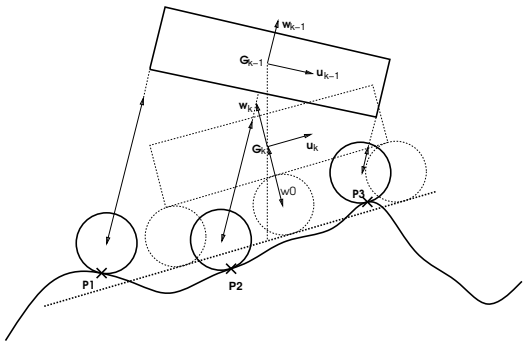

Figure 4: Iterative computation of the approximate placement

The new values of the placement parameters are deduced from this plane: ϕ_{k+1} and ψ_{k+1} are computed such that the vertical axis $\mathbf{w}$ of $\mathcal{R}_{rob}$ coincides with the plane normal. z_{k+1} is computed such that all the wheels contact this plane for a null deformation of the springs (see Fig. 4).

The procedure is iterated while $\mathcal{E}_k < \mathcal{E}_{k-1}$. When the algorithm stops at iteration k, it simply returns the vector $\mathbf{p}_{k-1}$ and the spring elongations $L_i(\mathbf{p}_{k-1})$.

Obviously, this iterative method is not garanteed to exactly converge to the solution of equation 1. However, the following remarks justify its interest: First, it is much more efficient than the algorithm $PLACE(\mathbf{q})$. Second, experimental tests performed with several terrain models, show that this simple method returns a placement very close to the one computed with $PLACE(\mathbf{q})$ when the portion of the terrain lying under the robot is relatively smooth. For the case of large terrain irregularities, $PLACE_{APP}(\mathbf{q})$ does not always succeed in decreasing $\mathcal{E}$ to its minimal value and therefore a larger error is possibly introduced by the algorithm. However, when this case occurs, some of the computed values $l_i(\mathbf{q})$ are generally too large to satisfy condition **(C2)** and consequently, the placement will not be considered as valid.

4.2 Validity of a Configuration

We consider now the problem of checking wether a given configuration $\mathbf{q}$ belongs to CS_{free} or not.

[3] the initial values are $\phi_0 = \psi_0 = 0$ and $z_0 = f(x_G, y_G)$

4.2.1 Deformation of the Suspensions

Let L_{max} be the maximal deformation allowed for the springs. A configuration $\mathbf{q}$ verifies (**C2**) iff:

$$\forall i \in [1,n], |l_i(\mathbf{q})| < L_{max} \quad (4)$$

4.2.2 Stability of the Placement

According to constraint (**C3**), the vertical projection of G has to belong to the support polygon. The shape of this polygon is clearly a function of ϕ, ψ and of the lengths $(l_i)_{i=1...n}$. 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 ψ, and the subset $S \subset]-\frac{\pi}{2},\frac{\pi}{2}[^2$ verifying this condition can be easily determined [4] . Thus, constraint **C2** is verified for a given $\mathbf{q}$ iff:

$$(\phi(\mathbf{q}),\psi(\mathbf{q})) \in S \quad (5)$$

4.2.3 Collision-free Placement

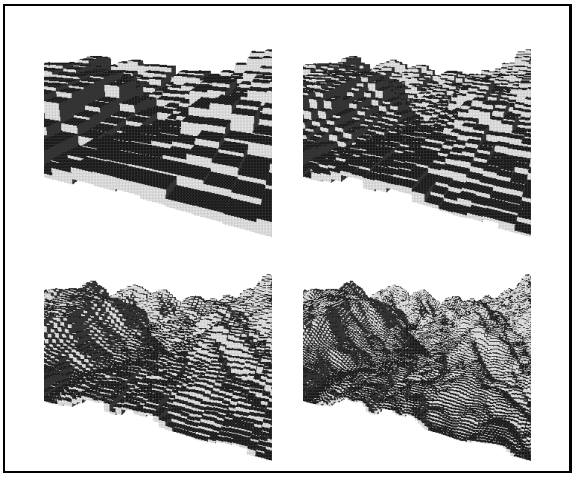

Figure 5: The hierarchical model (levels 5,6,7,8) for the terrain shown in Figure 2

Basically, the collision checker consists in verifying that the polygonal faces [5] of the polyhedron $R(\mathbf{q})$ do not intersect any of the underlying surface patches of the terrain model.

In order to improve the efficiency of the algorithm, we construct a hierarchical model (a quadtree) from the surfacic model of the terrain. The root of the quadtree (ie. the rectangloid cell corresponding to the definition domain of the terrain) is recursively subdivided into smaller cells. To each of these cells, we associate the extremal elevations z_{min} and z_{max} of the corresponding portion of the terrain. Thus, each level of the quadtree determines a set of cuboids that approximate the terrain geometry. The maximal level is obtained when the size of the cells is equal to the resolution Δter of the elevation map. Figure 5 shows the higher levels of the quadtree decomposition for the terrain model shown in Figure 2.

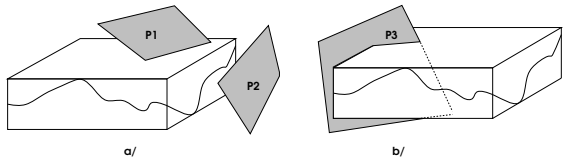

Figure 6: Polygon/quadtree collision checking

The following procedure is recursively applied, starting from the root of the quadtree, to efficiently check the non collision between a polygonal face P of the polyhedron $R(\mathbf{q})$ and the terrain. If the upper and lower horizontal faces of the cuboid associated to a given node do not intersect P (see Fig 6-a), then we are certain that P does not collide with the terrain. Similarly, when both faces intersect P (see Fig 6-b) we can directly conclude the existence of a collision. 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 [6], a more expensive test with the corresponding patch surface can be applied.

4.3 Computing Feasible Trajectories

Given a constant control (v_{lin}, v_{rot}) applied when the vehicle is at configuration $\mathbf{q}_0 = (x_0, y_0, \theta_0)$, we want to compute the configuration $\mathbf{q}_1 = (x_1, y_1, \theta_1)$ reached after the application of the control during a short time interval dt.

Let $\mathcal{R}_0 = (G_0, \mathbf{u}_0, \mathbf{v}_0, \mathbf{w}_0)$ be the frame which coincides with $\mathcal{R}_{rob}$ when the robot is at $\mathbf{q}_0$. We simplify the problem by assuming that, during the motion from $\mathbf{q}_0$ to $\mathbf{q}_1$, G remains in the plane defined by $(G_0, \mathbf{u}_0, \mathbf{v}_0)$ (see Fig 7).

Under this assumption, and for fixed values of (v_{lin}, v_{rot}), the trajectory of G relatively to $\mathcal{R}_0$ corresponds to a circular arc [7] in the plane $(G_0, \mathbf{u_0}, \mathbf{v}_0)$. This circular arc is centered at $(0, \rho, 0)$ and its radius is given by $\rho = v_{lin}/v_{rot}$. Therefore, after the time interval dt, the origne G of $\mathcal{R}_{rob}$ arrives at position

[4] In the current implementation S is simply approximated by an enclosed rectangular domain $[\phi_{max}, \phi_{max}] \times [-\psi_{max}, \psi_{max}]$

[5] In fact only the ones whose outer normal points toward $-\mathbf{z}$

[6] this case rarely occurs

[7] or a straight line segment when $v_{rot} = 0$

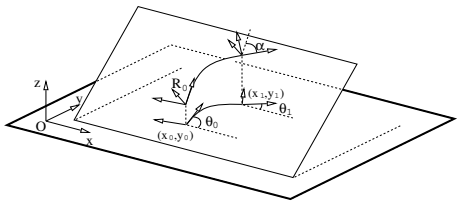

Figure 7: The kinematic constraint

$(\rho.sin\alpha, \rho.(1-cos\alpha), 0)$ where $\alpha = v_{rot}.dt$ corresponds to the angle between $\mathbf{u}_0$ and $\mathbf{u}$.

The coordinates of the configuration $\mathbf{q}_1$ are obtained by expressing these values into the reference frame $(O, \mathbf{x}, \mathbf{y}, \mathbf{z})$ from the matrix $M(\mathbf{p}_0)$.

5 The Path Planner

5.1 Principle

Given two configurations $\mathbf{q}_i$ and $\mathbf{q}_g$, we want to compute a feasible path connecting $\mathbf{q}_i$ to $\mathbf{q}_g$ and lying in CS_{free}.

The planner is based on a slighly modified version of the planning approach proposed in [1] for the optimal maneuvering of non-holonomic mobile robots moving in a two dimensional workspace. We recall that this approach basically consists in generating a graph $\mathcal{G}$ of discrete configurations that can be reached from $\mathbf{q}_i$ by applying sequences of fixed controls during a short time interval. The fixed controls generally correspond to drive forward or backward with a null or a maximal angular velocity which steers the vehicle toward the left or the right:

$$(v_{lin}, v_{rot}) \in \{-V_{lin}, V_{lin}\} \times \{-V_{rot}, 0, V_{rot}\} \qquad (6)$$

In order to limit the size of the graph, the configuration space is initially decomposed into an array of m small cuboid cells. This array is used during the search to keep track of small CS- regions (the cuboids) which have already been crossed by some trajectory. The successors generated into a marked cell are discarded and therefore, one node is at most generated in each cell.

We use a classical A^* algorithm to search the graph $\mathcal{G}$. Starting with $\mathbf{q}_i$ as current node, this node is expanded ie. its successors are computed and are stored into a list. The next iteration selects the best element of the list, which becomes in turn the current node to be expanded. The procedure is repeated until the goal is reached.

5.2 Neighbors of the Current Node

Figure 8 details how the current node $\mathbf{q}_{cur}$ (belonging to the cell C_{cur}) is expanded. For each of the six controls defined in (6), the algorithm tries to compute a neighbor $\mathbf{q}_{next}$ as follows: For a given control (v_{lin}, v_{rot}) a successor $\mathbf{q}'$ of $\mathbf{q}_{cur}$ is first computed as indicated in section 4.3. The algorithms described in section 4.1 and 4.2 allow to obtain the corresponding placement and to check the validity of $\mathbf{q}'$. If $\mathbf{q}'$ does not satisfy the placement constraints, the current control does not allow any node creation and the next control is considered. Otherwise, the same process is repeated from $\mathbf{q}'$ until a cell C_{next} adjacent to C_{cur} is reached. A graph node is associated to the reached configuration $\mathbf{q}_{next}$ only if the cell C_{next} has not been yet visited. The corresponding control and its duration are stored into the arc linking $\mathbf{q}_{cur}$ to $\mathbf{q}_{next}$.

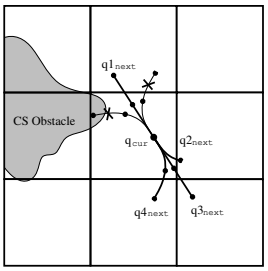

Figure 8: Computing the neighbors of $\mathbf{q}_{cur}$

5.3 Cost of the Arcs

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 ponderated in order to penalize the changes of control and the trajectories for which the angles $\phi(\mathbf{q})$, $\psi(\mathbf{q})$ or some of the lengths $l_i(\mathbf{q})$ are close to the limits imposed by the placement constraints (**C2**) and (**C3**). Therefore, the minimum-cost trajectory returned by the planner realizes a compromise between the distance crossed by the vehicle, the security along the path and a small number of maneuvers.

5.4 Choice of the Parameters and Complexity Issues

Note that during the node expansion, the placement constraints are checked in a discrete way after each incremental motion $(v_{lin}.dt, v_{rot}.dt)$. In order to be garanteed that the placement constraints remain satisfied during the incremental motions, the choice of these parameters needs to be related to the resolution Δter of the elevation map. They are chosen in such a way that the horizontal motion of any point of the robot is less than Δter. An upper-bound dep_{max} of this displacement can be estimated as follows: Let D_{max} be the maximal distance from the origin G of

$\mathcal{R}_{rob}$ to any point of the robot and let us consider that a linear velocity v_{lin} is applied during the time interval dt with a fixed radius of curvature $\rho = v_{lin}/v_{rot}$. The horizontal motion of any point of the robot can be upper-bounded by $dep_{max} = v_{lin}.dt + v_{rot}.D_{max}$. Therefore $dl = v_{lin}.dt$ is chosen such that:

$$dl \approx \frac{\rho}{\rho + D_{max}}.\Delta ter$$

The richness of the search space (ie. the size of graph G) is directly related to the cell decomposition of CS. Note that several incremental motions are generally needed to compute, for a given control, the neighbor of the current configuration (ie. to move from C_{cur} to C_{next}). A decomposition into large cells allows the planner to search efficiently for a solution into a reduced search space. For the case of constrained environments, requiring important maneuvering capacities of the vehicle, this reduced search space is unlikely to contain a solution, and the planner needs therefore to be called with a finer decomposition.

Let $O(n_{cs})$ be the number of discretization points along each coordinate axis of CS and let $O(n_{ter}^2)$ be the number of patches describing the terrain model. The graph G contains in the worst case $O(n_{cs}^3)$ nodes (and arcs). Each node expansion requires [8] $O(n_{ter}/n_{cs})$ calls to the algorithms described in section 4. Therefore these algorithms are called at most $O(n_{ter}.n_{cs}^2)$ times during the search. n_{ter} is given by the precision of the terrain model. n_{cs} can be seen as a "tuning" parameter which allows to realize a compromize between the completeness of the planner and its efficiency.

5.5 Simulation results

The algorithms presented above have been implemented in C and the planner runs on a Silicon Graphics Indigo Workstation. We experimented the planner with several simulated robot (with up to 8 wheels) and different terrain models. The size of the terrains was approximatively ten times superior to the size of the vehicle. For a discretization of CS into 64^3 cells [9] the computation times ranged from a 10 seconds up to a few minutes to find a solution (or to report failure). Figures 9 and 10 show two examples of trajectories computed by the planner for a six-wheels robot.

[8] the number of incremental motions linearly depends on the ratio between size of the $CS-$cells and the length dl which is proportional to the size Δter of the terrain patches.

[9] for a unit size vehicle, this correponds to cells whose size is $1/6$ along the x and y axis and 5 degrees along the θ axis

6 Conclusion

In this paper, we have addressed the path planning problem for the case of a mobile robot moving on rough terrains. From a formalization of the placement problem and of the constraints acting on the robot placements, we have proposed a discrete configuration space approach to solve the problem. We have described efficient algorithms which have been implemented in a path planner. The experiments show the ability of the planner to solve problems of practical interest in a reasonable amount of time.

References

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

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

[3] 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.

[4] E. Gat, M. Slack, D.P. Miller and R.J. Firby. Path planning and execution monitoring for a Planetary rover. In *IEEE International Conference on Robotics and Automation, Cincinnati (USA)*, 1990.

[5] G. Giralt and L. Boissier. The French Planetary Rover VAP: Concept and Current Developments. In *IEEE International Conference on Intelligent Robots and Systems, Raleigh (USA)*, July 92.

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

[7] J.P. Laumond. Feasible Trajectories for Mobile Robots with Kinematic and Environment Constraints. In *International Conference on Intelligent Autonomous Systems, Amsterdam (Netherland)*, 1986.

[8] R. Laurette, A. de St. Vincent, R. Alami, R. Chatila and V. Perebaskine. Supervision and Control of the AMR intervention robot. In *Fifth International Conference on Advanced Robotics, Pisa (Italy)*, 1991

[9] M. Iagolnitzer, F. Richard, J.F. Samson and P. Tournassoud Locomotion of an all terrain mobile robot. In *IEEE International Conference on Robotics and Automation, Nice (France)*, 1992.

[10] E. Schalit. ARCANE: Towards utonomous Navigation on rough terrains In *IEEE International Conference on Robotics and Automation, Nice (France).*

[11] 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.

[12] T. Siméon. Motion planning for a non holonomic mobile robot on three dimensional terrains. In *IEEE International Workshop on Intelligent Robots and Systems, Osaka (Japan)*, November 91.

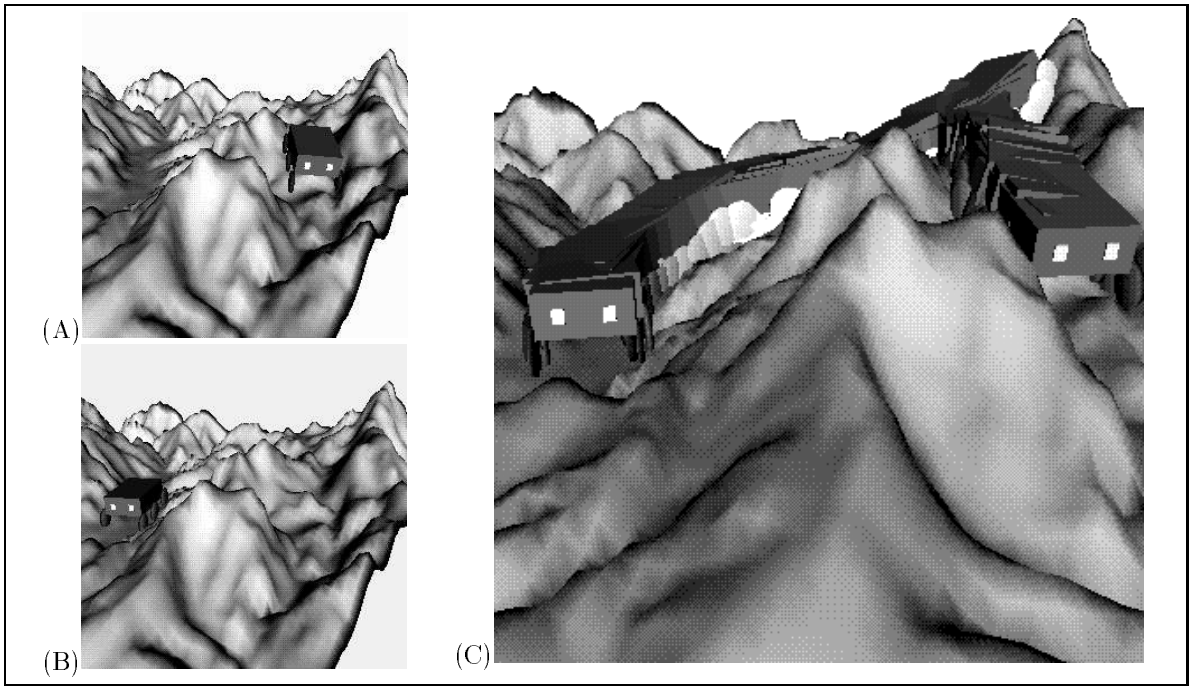

Figure 9: (A) The initial configuration. (B) The goal configuration. (C) The trajectory computed by the planner (computation time 87sec.)

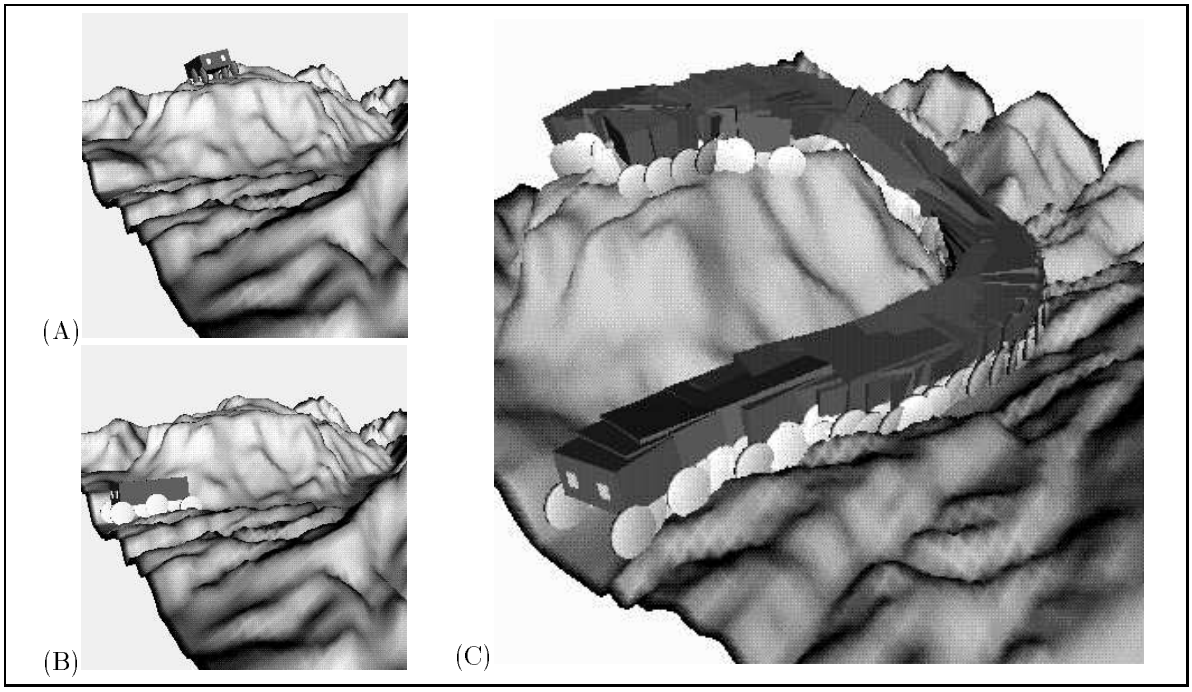

Figure 10: (A) The initial configuration. (B) The goal configuration. (C) The trajectory computed by the planner (computation time 31 sec.)