??, ??, 1?? (??)

Planetary Exploration by a Mobile Robot: Mission Teleprogramming and Autonomous Navigation

RAJA CHATILA, SIMON LACROIX, THIERRY SIMÉON, MATTHIEU HERRB

{raja,simon,nic,matthieu}@laas.fr

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

Received ??. Revised ??.

Abstract. Sending mobile robots to accomplish planet exploration missions is scientifically promising and technologically challenging. We present in this paper a complete approach that encompasses the major aspects involved in the design of a robotic system for planetary exploration. It includes mission teleprogramming and supervision at a ground station, and autonomous mission execution by the remote mobile robot. We have partially implemented and validated these concepts. Experimental results illustrate the approach and the results.

Keywords: Mobile robots, Mission Teleprogramming, Autonomous Navigation

1. Introduction and Overview

Several aspects make planet exploration a demanding and difficult problem for robotics:

- The robot has to operate in a natural, unstructured and a priori unknown environment.
- In the case of a remote planet (e.g., Mars), there is no possibility of continuous interaction with the robot because of important delays in communications and low bandwidth.
- The information on the robot and the environment is mostly acquired through the robot's own sensors. Little and rather poor a priori knowledge exist.

Because of time delays and low bandwidth, direct teleoperation is either impossible or very cumbersome. Telerobotics approaches [1], [2], [3] need a rather accurate model of the working space, and are therfore not applicable as such. The robot clearly needs important autonomous capacities.

It has been proposed [4], [5] to send one or more "simple" and *completely* autonomous robots, without any control from a ground station. Such robots, using a behavior-based control scheme [6], would achieve an imaging, measurement or sample collection mission. However, it is not possible in this scheme to interact with the robot in order to designate a precise site to which the robot has to navigate, or to send different missions. Even in this case, reaching a precise site needs capacities that such robots cannot be andowed with.

We present a different approach to meet the challenge of planet exploration by mobile robots [7]. It stems from the following considerations:

- The landing site may be remote from areas of interest to the scientists, mainly because it will be selected for its safety whereas the interesting areas are in general rather inadequate for landing [8]. Hence the robot has to travel some distance (tens of kilometers or more) from the lander to reach a *specific region* (not at random nor merely in a given direction).
- The mission is not defined once and for all. According to returned data, the scientists on Earth should be able to decide for the exploration of such or such site, the analysis of such sample, etc. It is necessary then to be able to control the robot, i.e., send it new missions. It is therefore important to know what it is doing. In order to provide it with new objectives, it is also important to know where it is.

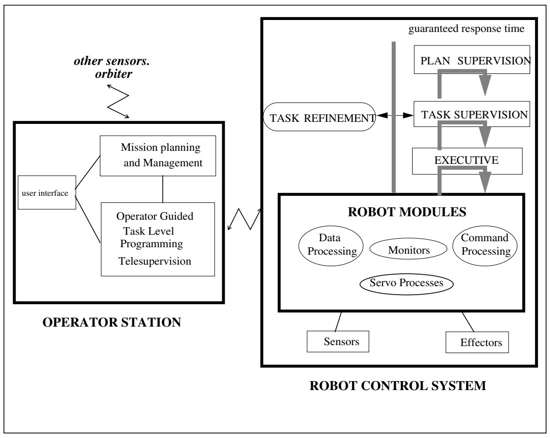

Fig. 1. Global Architecture

- Because the environment is poorly known, the mission can only be defined at a *task-level* in general, and not in its every detail (except in very special cases such as picking up a rock at reach). Hence the robot must be able to interpret the mission according to its actual context during its autonomous execution.
- The robot could fall into difficult situations wherein its perception, interpretation or decision making capacities are unsufficient. Human intervention would then be necessary for troubleshooting (which could be at a very low-level of command).

According to these arguments, we propose a global architecture for a robotic exploration system in two main parts (figure 1): a ground station for mission programming and supervision, and a remote robot able to interpret the mission and execute it autonomously.

2. The Ground Station

The Ground Station includes the necessary functions to allow a human to:

1. build a mission that can then be interpreted and executed by the robot. Such a mission is called an *executable mission*, as opposed to a higher level description of objectives as they may be expressed by planet scientists, and
2. supervise its execution, taking into account the delays and communication constraints.

The process of building an executable mission is decomposed into two phases which correspond to two different levels of abstractions and to different planning techniques:

1. a phase called "mission planning" which produces a "mission plan", i.e. a set of (partially) ordered steps with temporal constraints that will allow the robot to achieve a given goal.
2. a phase called "teleprogramming" that consists in refining a step in the mission in terms of tasks that can be interpreted and then executed by the robot. Depending on the nature of the mission and its difficulty, and on the amount of information available at planning time, an executable mission can be composed of a variable number of more or less detailed steps.

2.1. Mission Planning

Mission planning can be carried out with the help of a planning system able to take into account temporal and resource constraints as they can be foreseen at this stage. We have developed a temporal planning system called *IxTeT* [9], [10] which can reason on symbolic and numeric temporal relations between time instants. It produces a set of partially ordered tasks with temporal constraints. The explicit representation of time allows for a representation of planning operators that specify information concerning the duration of actions, the relative time when the consequences of an operator become true, the conditions which must remain true during action execution, joint effects with other operators executed in parallel, and the like. The descriptions of the world, the goals and the planning operators are given using symbolic and/or numeric temporal relations between time instants or elementary temporal relations between intervals which can be transformed into relations between instants.

At the mission planning level, the operator describes the mission in terms of results to be achieved, goals to be reached, temporal relations and numerical constraints, and so forth. The planner produces a set of tasks according to that description. This is the nominal plan.

As an example, let us consider the mission: go from `site1` to `site2`, there take a video panorama, send it to Earth, and finally proceed to

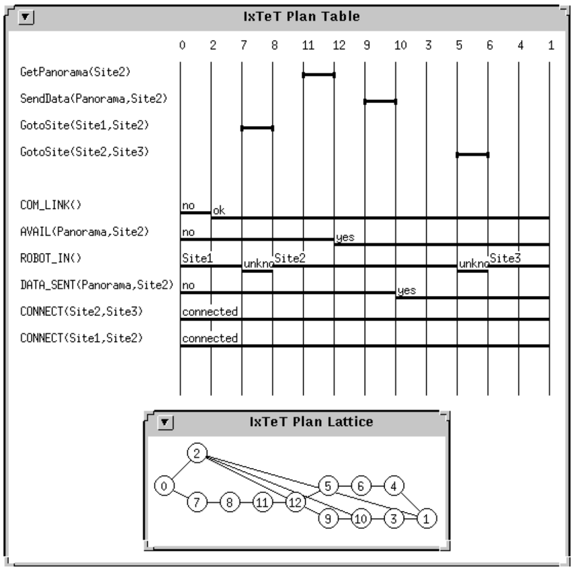

Fig. 2. Example of a mission plan

`site3`, in order to be there before nightfall. Given the appropriate task models, the IxTeT planner will produce the plan shown in figure 2, provided the overall time constraint ("before nightfall") is not too tight. The numerical constraints between time instants are not represented here.

In this example, we have chosen to use very simple task models (consisting in two instants: begin and end) for the sake of demonstrativity. According to these models, the two tasks `GotoSite` and `SendData` do not share the same resources, so the planner leaves them unordered. What this means is, they both occur after instant 12 (when `GetPanorama` is finished), and after instant 2 in the case of `SendData(Panorama,Site2)` (when the communication with the Earth becomes possible). In the case of the planetary rover, the physical machine cannot execute both tasks at the same time, because talking to the Earth means unfolding the antenna, pointing it, etc. But ordering these two tasks would be very constraining, considering that we do not know in advance which instant occurs first, 2 or 12.

2.2. Task-level Teleprogramming

Depending on the nature of the task and on its difficulty with respect to environment conditions, and depending on the robot decisional and operational capacities, a task selected by the planner can be sent as it is to the robot or must be further refined at the ground station. We call this process the "tele-programming phase". It uses all the information and expertise available at the ground station which may help the robot in performing its task. The result of this phase can be a more or less detailed program together with a set of execution "modalities" which provide a convenient representation for a class of conditional plans.

These execution modalities are expressed in terms of:

- constraints or directions to be used by the robot control system for executing the mission and each of its tasks ;
- a description of situations to monitor and the appropriate reactions to their occurrence; such reactions are immediate reflexes, "local" correcting actions (without questioning the mission), or requests for re-planning a task.

The plan produced by IxTeT corresponds to the nominal path of each task. We have to introduce the necessary indications to the robot to take into account failures and non-nominal events. A specific procedure is used to interactively build the final plan by appending other tasks.are signalled to the operator. The operator can modify and/or append tasks sing a graphic display and tools for the verification of preconditions and resource consumption.

Figure 3 shows an example. Here, the operator decided that the robot proceed to `Site3` should it fail to reach `Site2`; as well if it is short of time after completing the task `GetPanorama`. Of course, all events not made explicit should be handled by the robot itself.

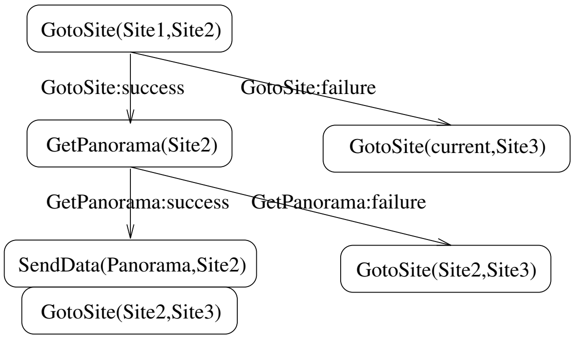

Fig. 3. The final plan

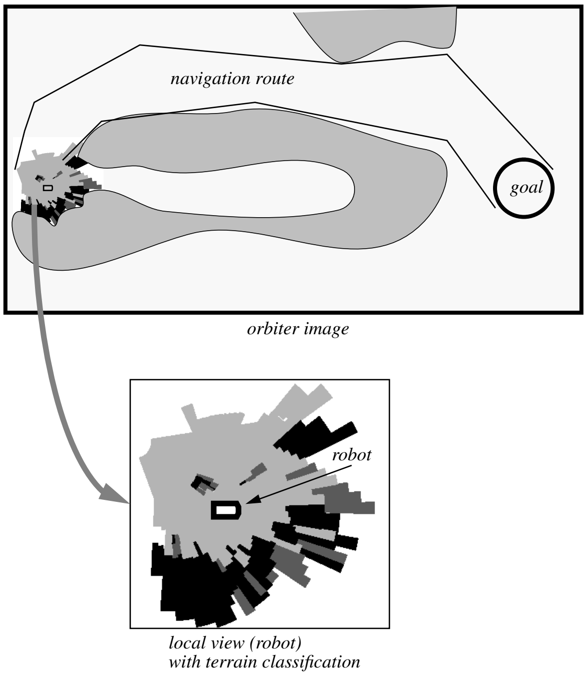

Fig. 4. Construction of a navigation route

At this point, the plan skeleton is complete, but the tasks need yet to be refined. For example, a motion task "GO-TO(site)" is replaced by a sequence of more robust global motions that guide the robot along a "good" route as it may be selected from data taken from orbit, and also relying on environment features or landmarks - if any [11]. The robot will be still executing autonomously its navigation, but instead of using only its own data, it also takes advantage of other observatons. In addition, execution modalities can be added to the task, to be used by the robot to take its own decisions. Such modalities include constraints and indications for selecting the adequate actions (e.g., decision to cross an uneven terrain, with respect to try to avoid it, at the price of a longer but easier trajectory). This teleprogramming phase ends up for the case of global navigation in constructing navigation routes as shown in figure 4.

Some task need also to be completely programmed if they are not already defined at the robot level. Then, the operator has to supply a program for that task. Here is a simplified example:

```
Task GetPanorama ( place ) {
  do {
    exec check_robot_at_place(place);
      report RESULT;
    if ( is_false(?RESULT) )
      raise END("context_failure");
    post point_camera(0.00)
      report STATUS;
    if ( is_failed(?STATUS) )
      raise END("hardware_failure");
    loop {
      post get_picture();
      post move_camera(30.00);
      exec check_camera_position(0.00)
        report RESULT;
      if ( is_true(?RESULT) )
        raise BREAK;
     } watching BREAK;
    raise END("success");
   } watching END;
 } export END;
```

The robot's version of the mission program includes the final mission plan, the modalities, the description of new tasks, and all necessary pieces of data. The plan is expressed as a data structure, consisting in a set of tasks, defined with their arguments, temporal constraints (relative to the start and end point, as well as duration) and modalities, connected by transitions labelled with internal and external events.

2.3. Telesupervision

Telesupervision in this context has both a mission monitoring role and a troubleshooting role. Because of communication constraints (communication requires pointing the antenna and hence cannot be done continuously) specific supervision commands such as status reports and data on mission execution must be included in the mission itself. In case of a problem encountered during execution, the robot must take the decision to call for help, or to continue the mission according to the given modalities.

3. The Rover

Because the robot is in a remote ill-known environment, and communications constraints prevent from a continuous exchange of data with it, it is not possible in general to plan its actions with all the details. Therefore, the robot control system should be able to interpret the tasks in terms of actions to be executed, taking into account the actual state of the system and of the environment [12]. Mission execution is completely autonomous and controlled on-board, without any direct interaction with the station (except if planned). Exchange of data with the Ground Station takes place as planned in the mission or when necessary because of execution status, e.g., the failure of some tasks.

The robot control architecture is derived from the architecture for complete autonomy presented in [13], in which the mission planning component is deported on the operator station, having more powerful computers as well as computer-aided facilities and human expertise at its disposal. It is organized into three levels.

The higher level is composed of a mission supervisor which interacts with the operator station and the next level (viewed as a set of processes which exchange signals with it).

The second level is composed of a task refinement planner and a task supervisor.

The activity of the supervisors consists in monitoring plan execution at their level by performing situation detection and assessment and by taking appropriate decisions in real time. In order to achieve this, the supervisor makes use of deliberation algorithms which are *guaranteed to be time-bounded* and compatible with the dynamics of the controlled system. Indeed, all deliberation algorithms which do not verify this property are actually performed by the planner (on-board or at the Operator's Station) upon request of the supervisor.

Note that in this architecture, on-board planning is necessary only at the second level. It is essentially a "refinement" using domain- or task-specific knowledge. For this, we use C-PRS [14] which provides a suitable framework for goal-driven as well as situation driven deliberation processes. Indeed, PRS implements script (called KA in PRS) selection and goal posting mechanisms. Planning can be performed through context-dependent goal decomposition; situation driven reaction can be performed by triggering procedures according to the environment model.

The lowest level includes the robot modules that perform perception and action execution. The response time of these modules that implement polynomial time algorithms is bounded. This level is managed and controlled by a central Executive in order to execute the actions requested by the task supervisor. The executive is a time-bounded system: its reactions to events are predetermined precompiled structure.

A module embeds primitive robot functions which share common data or resources. An internal control process called the "module manager" is responsible for receiving requests to perform these functions from the robot controller, and for otherwise managing the module. Each function being well defined, its activation or termination must respect certain conditions that the module manager verifies. Modules interact by message passing or by reading data exported by other modules, and by putting their own processing results into exported data structures (EDS). At a given time, a module can be executing several functions.

Such an architecture allows a level of robot autonomy which is essentially dependent upon the difficulty of the task and the state of the environment. The autonomy is determined by the procedures implemented on the refinement level, and the algorithms within the module functions.

4. Mission Execution

On-board plan supervision consists in sequencing the tasks according to expected events specified in the plan (begin and end events of the tasks, and time-synchronization events) as well as unspecified (for instance, task failure not addressed in the plan). In case of conflict between two tasks, the plan supervisor is responsible for deciding which task should be executed or interrupted and for enforcing that decision.

Each task in the plan corresponds to the execution of one or several procedures. According to the tasks and to the execution context, the procedures are either selected because they are explicitly designated in the task plan, and are then

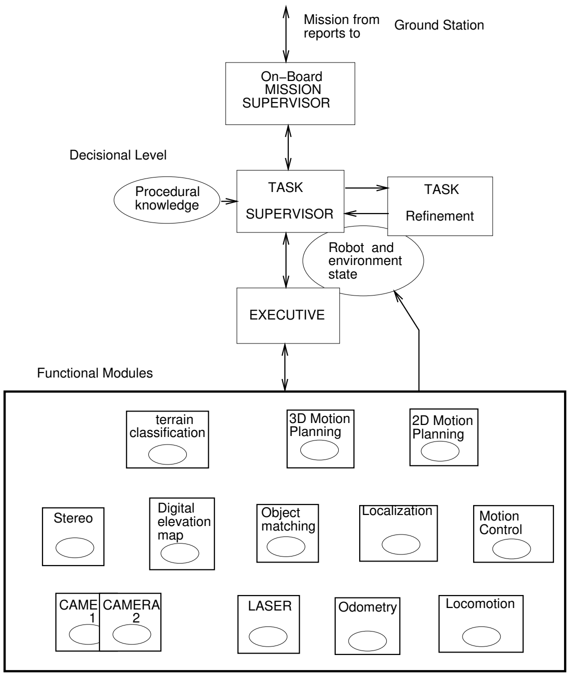

Fig. 5. On-Board Architecture

intanciated for execution, or are selected as a result of goal posting. In this case the selection of a procedure follows the general scheme of PRS and is based on some invocation conditions and on the context of execution as expressed in the data base. The choice of the best procedure, when several are possible candidates, is made by a meta-procedure that reasons on applicability criteria. Procedure selection is an iterative process.

The execution of a procedure may produce several outcomes. The plan explicitly provides the desired chaining between the tasks according to *some* of these outcomes. If this chaining is not explicit in the plan, default procedures are selected (or goals) and executed by the supervision system. Usually, such procedures will put the robot in a safe and stable situation, and try to communicate with the ground station.

As an example, we describe the procedure `GotoSite` which loops until the robot has reached the target site. Executing this procedure makes the system post new goals, and select new procedures that will eventually result in executing some actions (e.g., perception, trajectory planning, etc.) and so on:

```
task GotoSite (site) {
  loop {
    exec check_robot_at_site (site)
      report RESULT;
    if ( is_true (?RESULT) )
      raise END("success");
    post get_environment_model ()
      report MODEL;
    post choose_navigation_mode (?MODEL)
      report MODE;
    if ( equals(?MODE,#reactive) ) {
      fork watch_site_entry (site)
        report SITE_REACHED;
      exec move_until_obstacle ()
        watching SITE_REACHED
        report STATUS;
      if ( is_true(?SITE_REACHED) )
        raise END("success");
     }
    else {
      post find_sub_goal (?MODEL)
        report SUB_GOAL;
      post find_traj (?MODEL,?SUB_GOAL)
        report TRAJECTORY;
      if ( is_void(?TRAJECTORY) )
        raise END("failure");
      exec follow_traj (?TRAJECTORY);
     }
   } watching END;
 }
```

Here the robot starts by acquiring new data on the environment, and decides, on the basis of a first modelling of the terrain, which navigation mode should be selected. Two modes are possible: reactive and planned. The reactive mode is selected in case of a flat terrain almost free of obstacles. It makes the robot move toward the goal while trying to detect obstacles - without a full analysis of the terrain. In the planned mode, a navigation map is built, and a trajectory planner is selected to compute a collision free trajectory (either on flat or uneven terrain).

The selection of subgoals for navigation depends on the modalities associated with the plan, such as the navigation routes and landmarks.

The task supervisor then updates the execution modalities and posts the goal corresponding to the task. The task refinement level (see figure 5) selects the suitable procedures for achiev-

Fig. 6. ADAM in the Geroms test site

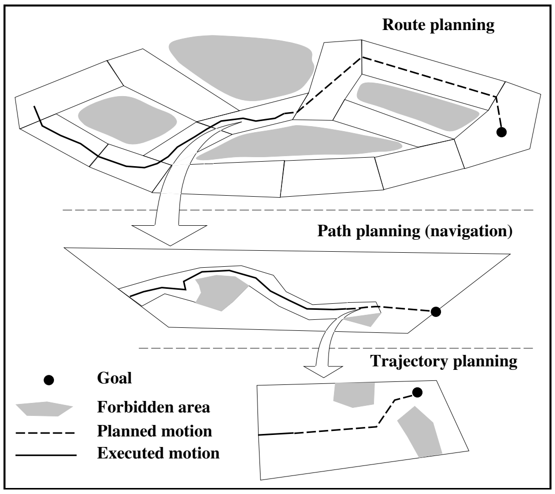

Fig. 7. Three levels of planning

ing the goal with respect to the execution context and the modalities. When the goal is fulfilled or recognized as unreachable, the task supervisor generates the task-termination event.

5. Autonomous navigation

This section presents an approach to autonomous robot navigation in an unknown planetary environment. It involves several levels of reasoning, several environment representations, and three different motion modes. We emphasise here especially on the "navigation level" of the whole system, which is in charge of reaching a distant goal by selecting sub-goals to reach, navigation modes to apply, and perception tasks to execute for this purpose.

We first present our general adaptative and hierarchical approach to autonomous navigation, pointing out the importance of the navigation level. Then we describe how terrain representations required by this level are incrementally built on the basis of 3D range and video sensory data. The algorithms that perform the selection of subgoals and perception tasks are described, and illustrated by experimental results with the robot ADAM[1] (see figure 6) that has performed a large number of runs using this system in two different test sites. at LAAS and at the Geroms lunar site at the French space Agency CNES.

The problem of long range navigation in unknown outdoors environments is not very frequently addressed. Important achievements are Robbie, [16] Ambler [18] and the navigation of the UGV [19].

An adaptative approach. According to a general "economy of means" principle due to limitations of on-board processing capacities, memory and energy put on the system, and to achieve a time-efficient behavior, we favour an *adaptive* approach in which the robot adapts its behaviour to the nature of the terrain [20], [21]. Hence, three motion modes are considered :

- A **reflex** navigation mode : on large flat and lightly cluttered zones, the robot locomotion commands are determined on the basis of a goal and informations provided by "obstacle detector" sensors ;
- A **2D planned** navigation mode : it relies on the execution of a planned 2D trajectory, using a binary description of the environment in terms of *Crossable/Non-Crossable* areas ;
- A **3D planned** navigation mode : this mode requires a precise model of the terrain, on which a fine 3D trajectory is planned and executed ;

A hierarchical approach. We assume that the terrain on which the robot must fulfill a navigation task is initially unknown, or mapped with a very low resolution. It is then only possible for an operator to specify a graph of *routes*, i.e., large corridors within which the robot has to move autonomously. To tackle this problem, we defined three layers of planning (figure 7) :

- *route planning* which chooses long-term paths to the goal on the basis of the initial informations. It selects a sub-goal for the path planning level ;

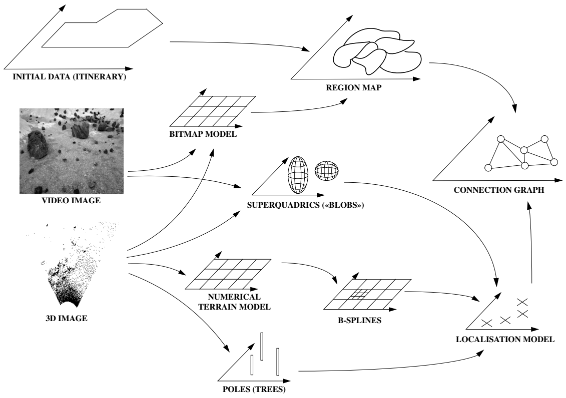

Fig. 8. The various terrain representations used in the system. Arrows represent the constructive dependancies between them

- *path planning* (or *navigation*) which reasons on a global qualitative representation of the terrain, built from the data acquired by the robot's sensors. It selects the next perception task to perform, the sub-goal to reach and the motion mode to apply ;
- Finally *trajectory planning* which determines the trajectory to execute (in one of the above mentioned three motion modes to reach the goal defined by the path planning level.

The main "Go-To" loop. Our approach to determine which navigation mode can be applied is based on a quick analysis of the raw 3D data produced either by a Laser Range Finder (LRF) or by stereovision. This quick analysis provides a description of the terrain in terms of navigation classes. This representation is incrementally built as the robot moves and new perceptions are fused with it to maintain a *global qualitative representation* of the environment. All "strategic" decisions are taken on the basis of this global representation. They concern the determination of the intermediate goal positions, the choice of the navigation mode to apply to reach them, as well as the definition of the next perception task to execute (which sensor to use ? with what operating modalities ? how should the data be processed ?). Such an approach involves the development of different perception and motion planning processes, and emphasizes the importance of the *navigation planner*, which is in charge of the strategic decisions.

5.1. Terrain representations

During navigation, a terrain representation is required for various processes: navigation planning, trajectory planning and robot localisation.
Aiming at building a "universal" terrain model that contains all the necessary informations is extremely difficult, inefficient, and moreover not really useful. It is more direct and easier to build different representations adapted to their use: for navigation decisions, for motion planning on flat terrain, on uneven terrain, and for localisation. Coherence relationships between these representations are to be maintained when necessary. The model is then *multi-layered and heterogeneous* (figure 8). Several perception processes coexist in the system, each dedicated to the extraction of specific representations. Perception is *multi-purpose.*

Incremental Terrain Modelling. For the purpose of navigation planning, a global representation that describes the terrain in terms of navigation classes is required. We focus in this section on the algorithms developed to build such a model from 3D data (produced either by a laser range finder or a correlation stereovision algorithm).

3D data classification. Applied each time 3D data are acquired, the classification process produces a description of the perceived areas in term in *terrain classes.* It relies on a specific discretization of the perceived area that respects the sensor resolution. The discretization defines "cells" on which different characteristics are determined : density (number of points contained in a cell compared with a nominal density defined by the discretization rates), mean altitude, variance on the altitude, mean normal vector and corresponding variances... A non-parametric bayesian classification procedure is used to label each cell : a learning phase based on prototypes classified by a human lead to the determination of probability density functions, and the classical bayesian approach is applied, which provides an estimate of the probability for each possible label. A decision function that privileges false alarms (i.e., labeling a flat area as obstacle or uneven) instead of the non-

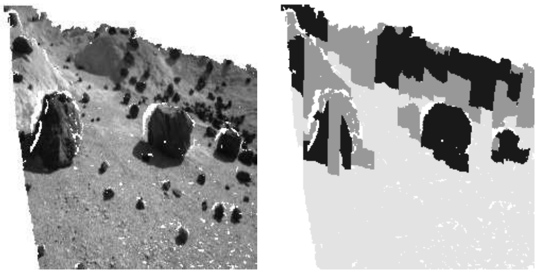

Fig. 9. Classification of a correlated stereo image : correlated pixels (left) and reprojection of the result in the camera frame (right - from clear to dark : unknown, flat, uneven and obtacle)

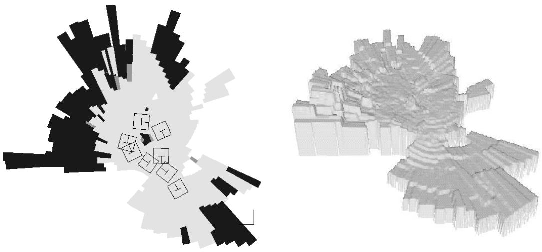

Fig. 10. Fusion of 8 different classified laser images : terrain classes (left) and altitude (right)

detections (i.e., the opposite: labeling an obstacle as a flat area) is used (figure 9).

Incremental fusion. The partial probabilities of a cell to belong to a terrain class and the variance on their altitude allow to perform a fusion procedure of several classifed images. The fusion procedure is performed on a bitmap, in the pixels of which are encoded all the cell attributes determined by the classification procedure (figure 10).

Model structure and management. For the purpose of navigation planning, the global bitmap model is structured into a region map, that defines a connection graph. Planning a *path* (as opposed to planning a *trajectory*) does not require a precise evaluation of the static and kinematic constraints on the robot : we simply consider a robot point model, and therefore perform an obstacle growing in the bitmap before segmenting it into regions (figure 11). The regions define a connection graph, whose nodes are on their borders, and whose arcs correspond to a region crossing.

In order to satisfy memory constraints, the global model is explicited as a bitmap only in the surroundings of the robot's current position, and

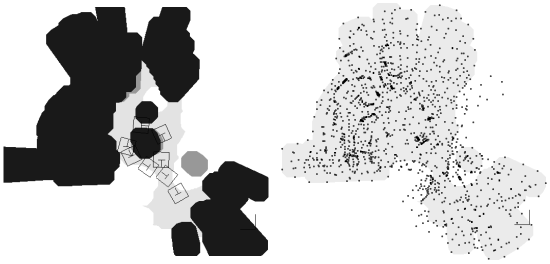

Fig. 11. The model of figure 10 after obstacle growing (left) and the nodes defined by the region segmentation (right)

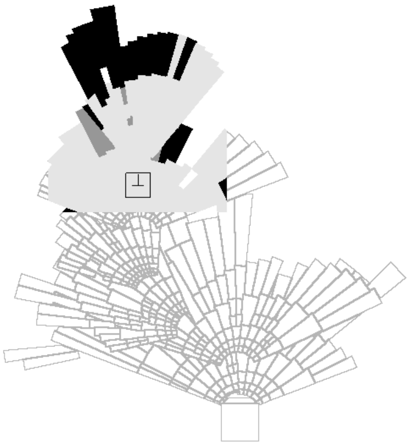

Fig. 12. Only the area surrounding the robot is explicited as a bitmap

the region model (much more compact) is kept in memory during the whole mission (figure 12).

5.2. Navigation planning

Each time 3D data are acquired, classified and fused in the global model, the robot has to answer autonomously the following questions :

- Where to go ? (sub-goal selection)
- How to go there ? (motion mode selection)
- Where to perceive ? (data acquisition control)
- What to do with the acquired data ? (perception task selection)

For that purpose, the navigation planner reasons on the robot capabilities (action models for perception and motion tasks) and the global terrain representation.

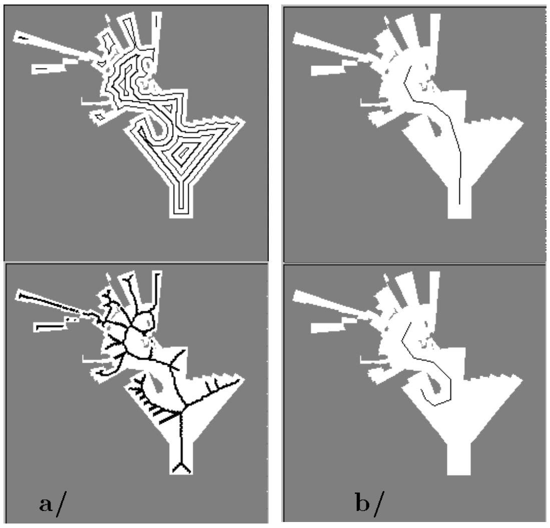

Fig. 13. The 2D planner: a/ distance to the obstacles and skeleton of the free space, b/ trajectories produced by the planner

A straightforward fact is that motion and perception tasks are strongly interdependant : executing a motion requires to have formerly modeled the environment, and to acquire some specific data, a motion is often necessary to go the adequate observation position.

Finding paths in the connection graph that minimizes some criteria (time and energy) is easily solved by classical search techniques, using cost functions that express these criteria ; but estimating the result (and the utility) of a perception task is much more difficult. We developed an error model of the classification procedure that allows to estimate the amount of information it can bring, and a model of the localisation task that predicts the precision on the robot position it can provide.

A direct and bruteforce approach to answer the former questions would be to perform a search in the connection graph, in which *all* the possible perception tasks would be predicted and evaluated at *each* node encountered during the search. Besides its drastic algorithmic complexity, this approach appeared unrealistic because the model of the classification task cannot not predict *what* will be effectively perceived : it is then difficult to estimate the interest of these tasks.

We therefore choose a different approach to tackle the problem : the perception task selection is *subordinated* to the motion task. A search algorithm provides an *optimal* path, that is analyzed afterwards to deduce the perceptions tasks to perform. The "optimality" criterion takes here a crucial importance : it is a linear combination of time and energy consumed, weighted by the terrain class to cross and the confidence of the terrain labeling. The introduction of the robot position uncertainty in the cost function allows to plan localisation tasks along the path.

5.3. Trajectory planning

Depending on the label of the regions produced by the navigation planner, the adequate trajectory planner (2D or 3D) is selected to compute the actual trajectory within these regions.

Flat Terrain The trajectory is searched with a simplified and fast method, based on bitmap and potential fields techniques. In a natural environment, and given the uncertainties of motion, perception and modelling, we consider it sufficient to approximate the robot by a circle and its configuration space is hence two dimensional, corresponding to the robot's position in the horizontal plane. Path planning is done according the following procedure :

- a binary bitmap *free/obstacle* is first extracted from the global bitmap model over the region to be crossed;
- a classical wavefront expansion algorithm then produces a distance map from which the skeleton of the free-space is computed (figure 13-a);
- the path reaching the sub-goal is obtained by propagating a potential through this skeleton. This path is finally transformed into a sequence of line segments and rotations (figure 13-b).

Search time only depends on the bitmap discretization, and not on the complexity of the environment. The final trajectory is obtained within less than 2 seconds (on a Sparc 10) for a 256×256 bitmap.

Uneven Terrain On uneven terrain, irregularities are important enough and the binary partition into *free/obstacle* areas 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

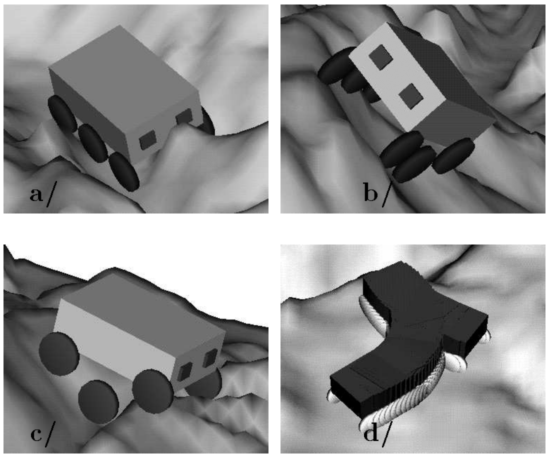

Fig. 14. The constraints considered by the 3D planner: a/ collision, b/ stability, c/ terrain irregularities and d/ kinematic constraint

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.

This planner, described in [23], computes a motion verifying such constraints by exploring a three dimensional configuration space $CS = (x, y, \theta)$ (the x-y position of the robot frame and its heading θ). The obstacles are defined in CS as the set of configurations which do not verify some of the constraints imposed to the placement of the robot (figure 14). The ADAM robot is modelled by a rigid body and six wheels linked to the chassis by passive suspensions. For a given configuration, its placement results from the interaction between the wheels and the terrain, and from the balance of the suspensions. The remaining parameters of the placement vector (the z coordinate, the roll and pitch angles ϕ, ψ), are obtained by minimizing an energy function.

The planner 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.

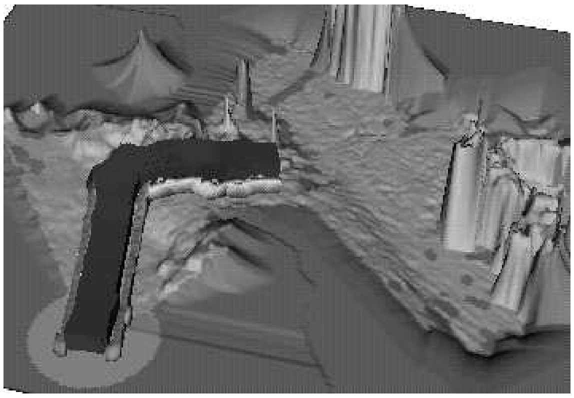

Fig. 15. A 3D trajectory planned on a real elevation map

In the case of incremental exploration of the environment, an additional constraint must be considered: the existence of unknown areas on the terrain elevation map. Indeed, any terrain irregularity may hide part of the ground. When it is possible (this caution constraint can be more or less relaxed), the path must avoid such unknown areas. If not, it must search the best way through unknown areas, and provide the best perception point of view on the way to the goal. The avoidance of such areas is obtained by an adapted weight of the arc cost and also by computing for the heuristic guidance of the search, a potential bitmap which includes the difficulty of the terrain and the proportion of unknown areas around the terrain patches [22].

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. Search time strongly depends on the difficulty of the terrain. The whole procedure takes between 40 seconds to a few minutes, on an Indigo R4000 Silicon Graphics workstation. Figure 15 shows a trajectory computed on a real terrain, where darker areas correspond to interpolated unknown terrain.

6. Conclusion

The presented approach is based on a generic architecture for intervention robots we are developing for several highly demanding applications, including planet exploration. It has been partially instanciated in the "EDEN" experiment carried out at LAAS with the mobile robot ADAM, especially the functions necessary for autonomous navigation in a natural environment, including perception, environment modeling, localization, path and trajectory planning and execution on flat or uneven terrain. The integration of the full system including the mission planning and teleprogramming phases is being currently achieved.

Notes

1. ADAM is property of Framatome and Matra Marconi Space, currently lent to LAAS.

References

1. T. Sheridan. Telerobotics. In *IEEE International Conference on Robotics and Automation, Scottsdale, (USA)*, 1989.
2. G. Hirzinger, J. Heindl, and K. Landzettel. Predictive and Knowledge-based Telerobotic Control Concepts. In *IEEE International Conference on Robotics and Automation, Scottsdale, (USA)*, pages 1768–1777, 1989.
3. S. Hirai and T. Sato. Motion Understanding for World Model Management of Telerobot. In *Robotics Research : The Fifth International Symposium, Tokyo (Japan)*, pages 5–12, 1989.
4. C. M. Angle and R. A. Brooks. Small Planetary Rovers. In *IEEE International Workshop on Intelligent Robots and Systems (IROS '90), Tsuchiura (Japan)*, 1990.
5. D. P. Miller, R. S. Desai, E. Gat, R. Ivlev, and J. Loch. Experiments with a small behaviour controlled planetary rover. In *Missions, Technologies and Design of Planetary Mobile Vehicles, Toulouse, France*, Sept. 1992.
6. R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, April 1986.
7. G. Giralt and L. Boissier. THE FRENCH PLANETARY ROVER VAP: Concept and Current Developments. In *IEEE International Workshop on Intelligent Robots and Systems (IROS '92), Raleigh (North Carolina, USA)*, pages 1391–1398, July 1992.
8. F. Costard *et al.* A reference martian mission for a long range rover. In *Missions, Technologies and Design of Planetary Mobile Vehicles, Toulouse, France*, Sept. 1992.
9. M. Ghallab and A. Mounir Alaoui. Managing Efficiently Temporal Relations Through Indexed Spanning Trees. In *11th International Joint Conference on Artificial Intelligence (IJCAI), Detroit, Michigan (USA)*, pages 1297–1303, 1989.
10. C. Dousson, P. Gaborit, and M. Ghallab. Situation recognition: Representation and algorithms. In *13th International Joint Conference on Artificial Intelligence (IJCAI), Chambery (France)*, 1993.
11. R. Chatila, R. Alami, S. Lacroix, J. Perret, and C. Proust. Planet exploration by robots : from mission planning to autonomous navigation. In *International Conference on Advance Robotics*. ICAR'93, Tokyo (Japon), November 1993.
12. R. Chatila, R. Alami, B. Degallaix, and H. Laruelle. Integrated planning and execution control of autonomous robot actions. In *IEEE International Conference on Robotics and Automation, Nice, (France)*, 1992.
13. R. Alami, R. Chatila, and B. Espiau. Designing an intellingent control architecture for autonomous robots. In *International Conference on Advance Robotics*. ICAR'93, Tokyo (Japon), November 1993.
14. F. Ingrand, M.P. Georgeff, and A.S. Rao. An architecture for real-time reasoning and system control. *IEEE Expert. Intelligent Systems and Their Applications*, 7:pp.34–44, 1992.
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.
18. E. Krotkov, M. Hebert, M. Buffa, F. Cozman, and L. Robert. Stereo friving and position estimation for autonomous planetary rovers. In *IARP 2nd Workshop on Robotics in Space, Montreal, Canada*, 1994.
19. M. Hebert. Pixel-based range processing for autonomous driving. In *IEEE International Conference on Robotics and Automation, San Diego, California*, 1994.
20. R. Chatila, S. Fleury, M. Herrb, S. Lacroix, and C. Proust. Autonomous navigation in natural environment. In *Third International Symposium on Experimental Robotics, Kyoto, Japan, Oct. 28-30*, 1993.
21. S. Lacroix, R. Chatila, S. Fleury, M. Herrb, and T. Simeon. Autonomous navigation in outdoor environment : Adaptative approach and experiment. In *IEEE International Conference on Robotics and Automation, San Diego, California*, 1994.
22. F. Nashashibi, P. Fillatreau, B. Dacre-Wright,T. Siméon. 3D autonomous navigation in a natural terrain. *IEEE International Conference on Robotics and Automation, San Diego (USA)*, May 1994.
23. T. Siméon, B. Dacre Wright. A Practical Motion Planner for All-Terrain Mobile Robots. *IEEE International Conference on Robots and Systems, Yokohama (Japan)*, July 1993.