Autonomous Rover Navigation on Unknown Terrains.

Article *in* The International Journal of Robotics Research · January 2002

Source: DBLP

CITATIONS
21

READS
424

7 authors, including:

Simon Lacroix
French National Centre for Scientific Research
165 PUBLICATIONS **4,280** CITATIONS
SEE PROFILE

Sara Fleury
French National Centre for Scientific Research
76 PUBLICATIONS **3,139** CITATIONS
SEE PROFILE

Matthieu Herrb
French National Centre for Scientific Research
48 PUBLICATIONS **1,530** CITATIONS
SEE PROFILE

Raja Chatila
Sorbonne Université
233 PUBLICATIONS **13,771** CITATIONS
SEE PROFILE

Autonomous Rover Navigation on Unknown Terrains Functions and Integration

Simon Lacroix, Anthony Mallet, David Bonnafous
Gérard Bauzil, Sara Fleury, Matthieu Herrb, and Raja Chatila
LAAS/CNRS
7, av. du Colonel Roche
F-31077 Toulouse Cedex 4 - France
Firstname.Lastname@laas.fr

Abstract

Autonomous long range navigation in partially known planetary-like terrains is yet an open challenge for robotics. Navigating hundreds of meters without any human intervention requires the robot to be able to build various representations of its environment, to plan and execute trajectories according to the kind of terrain traversed, to control its motions and to localize itself as it moves. All these activities have to be scheduled, triggered, controlled and interrupted according to the rover context. In this paper, we briefly review some functionalities that have been developed in our laboratory, and implemented on board the Marsokhod model robot Lama. We then present how the various concurrent instances of the perception, localization and motion generation functionalities are integrated. Experimental results illustrate the functionalities throughout the paper.

1 Introduction

Autonomous rovers are definitely involved in the various planetary exploration programs supported by the agencies (see *e.g.* various contributions in the i-SAIRAS and ASTRA conferences [26, 5]). Indeed, the constraints on the communications with earth calls for vehicles able to autonomously perform experiments and data collection tasks. Such a challenge has motivated roboticists for over 15 years, from pioneer work in the US in the 80's (*e.g.* the robots Robby [45] and Ambler [7]), to most recent developments throughout the world [10, 44, 52, 31, 54]. Moreover, the study of autonomous rovers in a planetary exploration context is also relevant for several applications on earth: scientific exploration in Antarctica [46], demining [14], military applications [30], surveillance...

But to foster ambitious exploration missions, future planetary rovers will have to fulfill tasks described at a high abstraction level, such as `'reach the top of that hill'` or `'explore this area'`. This calls for the ability to navigate for several hundreds of meters, dealing with various and complex situations, without any operator intervention. Such an ability is still quite an open challenge: it requires the *integration* and *control* of a wide variety of autonomous processes, ranging from the lowest level servoings to the highest level decisions, considering time and resource constraints.

We are convinced that no simple autonomy concept can lead to the development of robots able to tackle such complex tasks: we believe in the efficiency of *deliberative* approaches [11], that are able to plan and control a variety of processes. Following such a paradigm and according to a general economy of means principle, we want the robot to autonomously *adapt* its decisions and behavior to the environment and to the task it has to achieve [12]. This requires the development of:

- Various methods to implement each of the perception, decision and action functionalities, adapted to given contexts;

FIGURE 1: *The robot Lama on its experimentation site.*

- An architecture that allows for the integration of these methods, in which deliberative and reactive processes can coexist;
- Specific decision-making processes, that dynamically select the appropriate decision, perception and action processes among the ones the robot is endowed with.

In this paper, we present the current state of development of the robot Lama, an experimental platform within which our developments related to autonomous long range navigation are integrated and tested. We especially focus on the necessity to integrate various implementations of each of the main functionalities required by autonomous navigation. After a brief description of Lama and its equipment, the rest of the paper is split in two parts: the first part briefly presents the main functionalities required by long range navigation we currently consider (environment modeling, rover localization, path and trajectory planning), while the second part insists on the problems raised by the *integration* of these functionalities. The paper is intended to be an overview of our work, but still, various mentions to open problems are discussed throughout.

2 The Robot Lama

Lama is a 6-wheels Marsokhod model chassis [28] that has been totally equipped at LAAS[1]. The chassis is composed of three pairs of independently driven wheels, mounted on axles that can roll relatively to one another, thus giving the robot high obstacle traversability capacities (figure 2). The length of the chassis can be actively controlled: each of its axles can move one by one, independently from the others. This "crawling", or peristaltic locomotion mode is especially suited to climb over steep sandy slopes for instance [4], but has actually not yet been not used in our integrated experiments. Each motor is driven by a servo-control board, and the rover maximal speed is $0.17 m.s^{-1}$.

Lama is equipped with the following sensors:

- Each wheel is equipped with a high resolution optical encoder, allowing fine speed control and odometry;
- Five potentiometers provide the chassis configuration;

[1] Lama is currently lent to LAAS by Alcatel Space Industries.

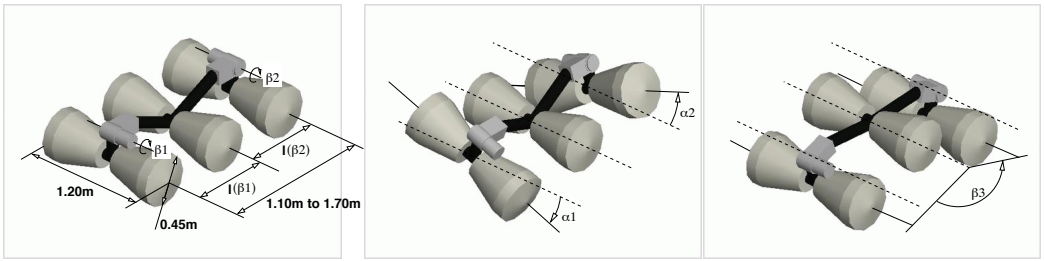

FIGURE 2: *Dimensions of the chassis of Lama and passive articulations of the axles.*

- A 2 axes inclinometer provides robot attitude, a magnetic fluxgate compass and an optical fiber gyrometer provide robot orientation and rotational speed;
- A first stereo bench on top of a pan and tilt unit, is mounted on a $1.80m$ mast rigidly tied to the middle axle. This bench has an azimuthal field of view of approximately 60°, and is mainly dedicated to goal and landmarks tracking;
- A second stereo bench, also supported by a PTU, is mounted upon the front axle, at an elevation of $0.80m$. It has a azimuthal field of view of approximately 90°, and is mainly dedicated to terrain modeling in front of the robot;
- A differential carrier-phase GPS receiver[2] is used as a reference to evaluate the localization algorithms.

All the computing equipment is in a VME rack mounted on the rear axle of the robot. The rack contains four CPUs, operated by the real-time OS VxWorks. Two 68040 are in charge of the data acquisitions (except the camera images) and of the locomotion and PTU control, whereas all the environment modeling and planning functionalities are run on two PowerPc's.

The terrain on which we test the navigation algorithms is approximately $100\times50m^2$. It contains a variety of terrain types, ranging from flat obstacle-free areas to rough areas, including gentle and steep slopes, rocks, gravel, trees and bushes (some pictures of this terrain can be seen in various places throughout the paper).

Part A: Navigation functionalities

3 Environment Modeling

Perceiving and modeling the environment is of course a key capacity for the development of autonomous navigation. Environment models are actually required for several different functionalities: to plan trajectories (section 5), paths and perception tasks (section 8), to localize the robot (section 4), and also to control the execution of trajectories. We are convinced that there is no "universal" terrain modeling algorithm, *i.e.* that can extract all the informations required by navigation from the various data. The rover must be therefore be able to manage various *heterogeneous* environment representations, adapted to specific functionalities and produced by dedicated algorithms. This section presents two environment mapping functions that use stereovision data as input, and the way to manage them during long missions. Pixel-based stereovision is indeed very efficient in natural terrains, as they are almost always textured (besides some very particular cases, such as on the earth' poles or in a salt lake). Moreover, progresses in the stereovision algorithms and in processor technologies nowadays allow the production of dense 3D frames at several Hertz [17, 48].

[2]Currently lent to LAAS by CNES.

FIGURE 3: *Discretization of a 3D stereo image. Left: regular Cartesian discretization in the sensor frame; right: its projection on the ground (the actual discretization is much finer).*

FIGURE 4: *An example of classification result. From left to right: image showing the pixels to which the stereovision algorithm could associate a depth (note that no depth is associated to the rover front stereo bench, visible in figure 3), description of the perceived area with the cells partial probabilities to be an obstacle (the clearer the more traversable), and re-projection of these probabilities in the sensor frame.*

3.1 Qualitative Model

To give the rover qualitative informations on the *traversability* of the perceived terrain, we developed a method that produces a description of the terrain in terms of *navigability classes*, on the basis of stereovision data [34]. The method is a classification procedure that produces a probabilistically labeled polygonal map, close to an occupancy grid representation. Most of the existing contributions to produce similar terrain models come to a data segmentation procedure (*e.g.* [43, 22]), that produce a *binary* description of the environment, in terms of traversable and non-traversable areas. But approaches that describe the terrain traversability in a continuous way, be it with probabilities, fuzzy numbers [24] or roughness estimate [57], are more robust, as they do not require any threshold determination – a tedious problem with segmentation algorithms.

The method relies on a specific discretization of the perceived area, that defines a *cell image*. This discretization "respects" the sensors characteristics, as the cell resolution decreases with the distance according to the decrease of the data resolution (figure 3).

Every cell is labeled with a non-parametric Bayesian classifier, using a learning base built off-line by an operator from sample images. On line, a 10-dimensions attribute vector is computed for each cell (cell density – number of points contained in a cell compared with a nominal density defined by the discretization rates, variance on the altitude, mean normal vector and corresponding variances). The usual Bayes formula then yields the determination of the partial probabilities for each cell to correspond to a pre-defined traversability class. Figure 4 shows a classification result, with two terrain classes considered (flat and obstacle). The discretization is dynamically controlled to allow a finer description: cell's whose probabilities distribution yield to a high statistical entropy are split, and the sub-cells are then classified. The classification results can also be combined with a terrain physical nature Bayesian classifier, on the basis of texture or color attributes. This latter classifier is however more complex (and not yet integrated in our experiments), and requires the knowledge of the types of terrain the robot is likely to encounter. Indeed, there are hundreds of "physical terrain nature" classes, and their definition is objective, whereas the definition of traversability classes is subjective, and fully defined by the considered rover characteristics.

One of the main advantages of this method is that thanks to the probabilistic description, local

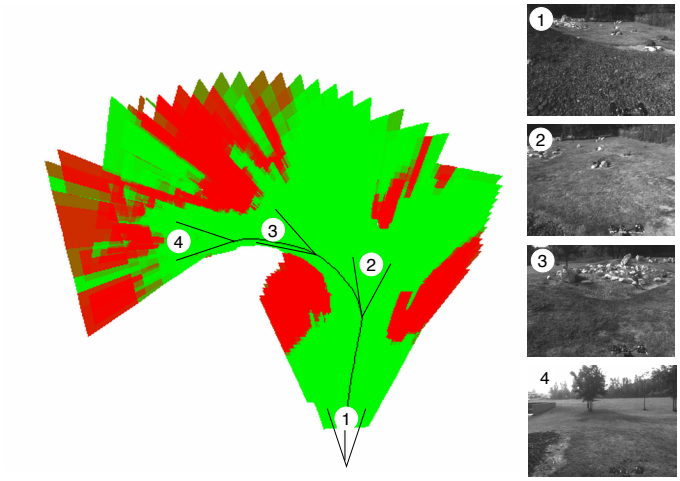

FIGURE 5: *A qualitative terrain model integrating 50 stereo pairs, acquired along a 20m trajectory.*

maps perceived from different viewpoints can be very easily merged into a global description, using the Bayes formula. The fusion procedure is performed on a Cartesian grid (a bitmap – figure 5), in the pixels of which are encoded cell partial probabilities and some additional informations (namely elevation and variance on the elevation). This terrain model can be either used to generate elementary motions on rather obstacle-clear terrains (section 5.1), or to reason at the path level (section 8).

3.2 Digital Elevation Map

The qualitative model is not suited to represent the terrain fine geometric features, required to navigate in rough terrains or to model landmarks. For that purpose, a digital elevation map (DEM), that represent the terrain as a set of elevations computed on a regular horizontal Cartesian grid, is computed as the rover navigates.

Although there has been several contributions to DEM building with rovers (see *e.g.* [32, 23]), we think that it has still not been addressed in sufficiently satisfactory way. The main difficulty comes from the uncertainties on the 3D input data, that can hardly be propagated throughout the computations and represented in the grid structure. Figure 6 illustrates this difficulty for various sensor/terrain configurations: ideally, a good way to rigorously manage the uncertainties would be to fill a 3D occupancy grid [59], from which a DEM would be extracted.

A 3D occupancy grid would however occupy a huge memory for large terrain models. But to have insights on a way to manage the uncertainties in a DEM, we studied occupancy grid building methods in the 2D case. It turned out that the best results were produced with Dempster-Shafer probabilistic fusion framework [47], using a sensor model that decouples the *imprecisions* on the points coordinates and the *confidence* on the fact that there is really a point in a particular place (figure 7).

On the basis of these results, we developed a heuristic fusion algorithm for the real 3D case. For each cell of the DEM, 3 values are computed: an elevation z, a *precision* σ on the elevation, and a *confidence* C on the elevation and precision. An interpolation is performed when integrating the 3D points, represented as 2D planar patches.

The fusion of a 3D point in the DEM is done as follows: first, the surface S of the projection of the corresponding planar patch is computed. Then, a bitmap painting algorithm computes the set of n cells Σ that intersect S, and a *confidence* C is computed as $1/n$ (this confidence value is therefore roughly inversely proportional to the square of the measured depth, thus conforming to the error behavior of stereo-vision). Finally, each cell of Σ is updated using the following rules:

$$z_{k+1} = \frac{C_k.z_k + C_c.z_c}{C_k + C_c} \quad ; \quad \sigma_{k+1} = \frac{C_k.\sigma_k + C_c.\sigma_c}{C_k + C_c} \quad ; \quad C_{k+1} = C_k + C_c$$

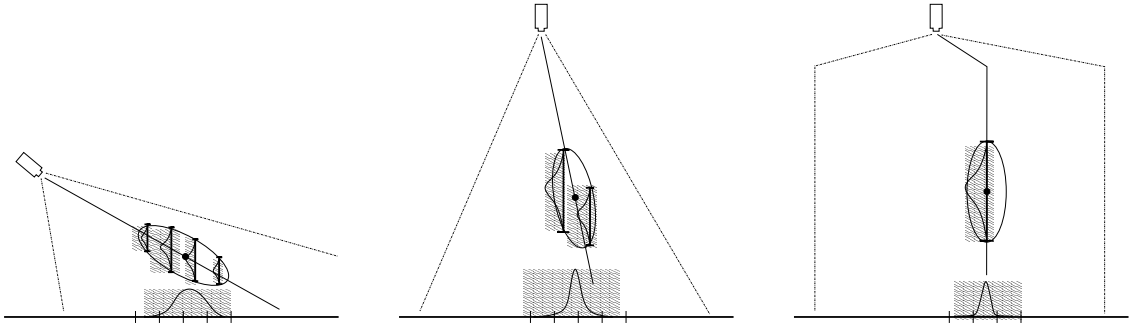

FIGURE 6: *2D illustration of the sensor errors with respect to an horizontal grid. In the leftmost case, the one encountered with range images acquired from a rover, the precision of the data, which is essentially in the range estimate, should be encoded as occupancy probabilities in a grid's cells – voxels in the 3D case. In the rightmost case, which correspond to data acquired from a high altitude device (traditional photogrammetry case), the problem is much better conditioned: the precision on the data can be directly represented by a precision on the elevation computed for the cells. One can also guess on these figures that the variable resolution of the data play an important role.*

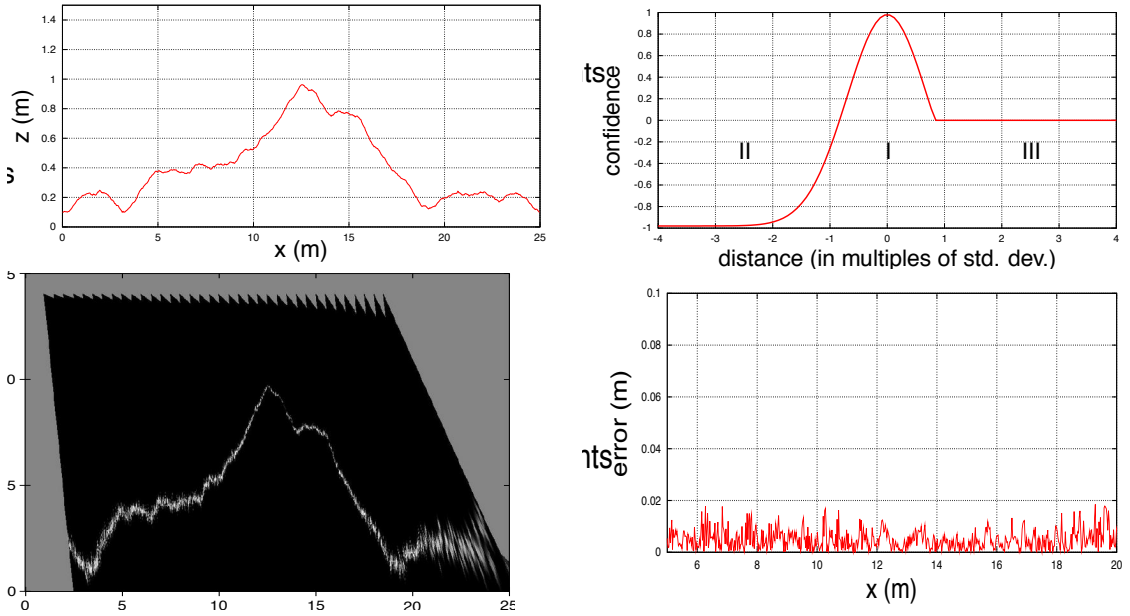

FIGURE 7: *Illustration of a 2D occupancy grid method using a Dempster-Shafer formalism. A simulated terrain profile (top left) is perceived by a range sensor, modeled as shown on the top left. Basically, this model is a Gaussian centered around the depth measurement (0 here), whose standard deviation is proportional to the square of the depth to take into account the behavior of stereovision [42]. The confidence ranges from −1 (empty, 100% probable) to +1 (occupied, 100% probable), and the area numbered* III *represents the occlusion due to the presence of the measured point (confidence of* 0*, maximum uncertainty). The bottom left figure represents the 2D occupancy grid built with this model. The differences between the real profile and the one extracted from this grid (by taking the most probable elevation in each column) is shown on the bottom right. Note that thanks to the fusion of numerous data points, the remaining errors (less than 2cm) are much smaller than the standard deviation on the coordinates of the points acquired by the camera (10cm times the square of the distance in this simulated case).*

FIGURE 8: *Analysis of our DEM building method in the simulated 2D case of figure 7. The left plot shows the error using a simple averaging of the raw 3D points in the cells. On the right, our method yields reconstruction errors almost as good as with the rigorous Dempster-Shafer approach of figure 7*

.

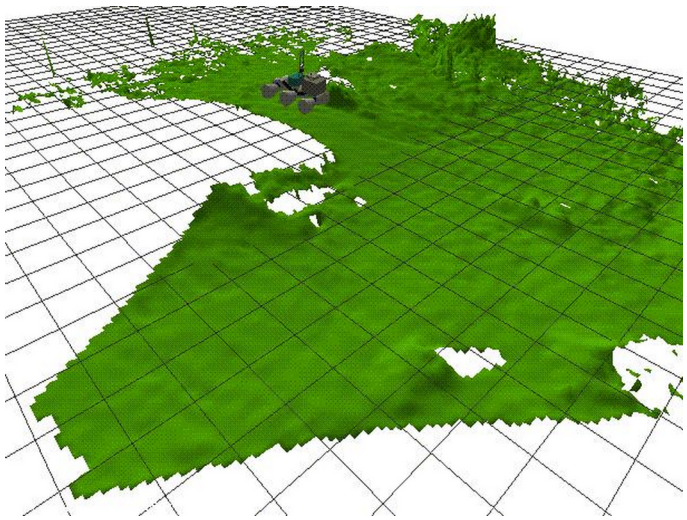

FIGURE 9: *A digital elevation map built from the same images that produced the qualitative model of figure 5. The grid size is* $1 \times 1m$*, and the map resolution is here* $0.1 \times 0.1m$*.*

the subscripts k, $k+1$ and c respectively denoting the values encoded in the DEM before fusion, after fusion, and the current point measure.

This technique gives results comparable to the rigorous Dempster-Shafer fusion (figure 8). The produced map can be either used to detect landmarks (section 4.3), or to generate elementary trajectories on rough terrains (section 5.2).

3.3 Maps Structure

To fuse several perceptions into a global map, be it the qualitative model or the DEM, the robot must be localized with a precision of the order of a cell size. This constraint is of course not always satisfied: we propose here a map structure that can tolerate localization errors, while preserving the possibility to build a consistent map.

Provided that the localization is precise enough between successive data acquisitions (*i.e.* perceptions that are close to each other), consistent local maps with a size of about the surface perceived by a 3D image can be built. A rigid local map is therefore initiated as the rovers starts. During navigation, the perception of some areas beyond the limits of this map triggers the creation of a new local map, the old one being memorized and not updated anymore. The process then goes on, as illustrated in figure 10: the resulting model is a chained list of small local maps, positioned with respect to a global frame.

The main advantage of this structure appears when the rover crosses a previously modeled area. In such cases, the global position is very likely to have drifted so much that the direct fusion of the overlapping maps would yield an inconsistent model. If a localization algorithm is able to

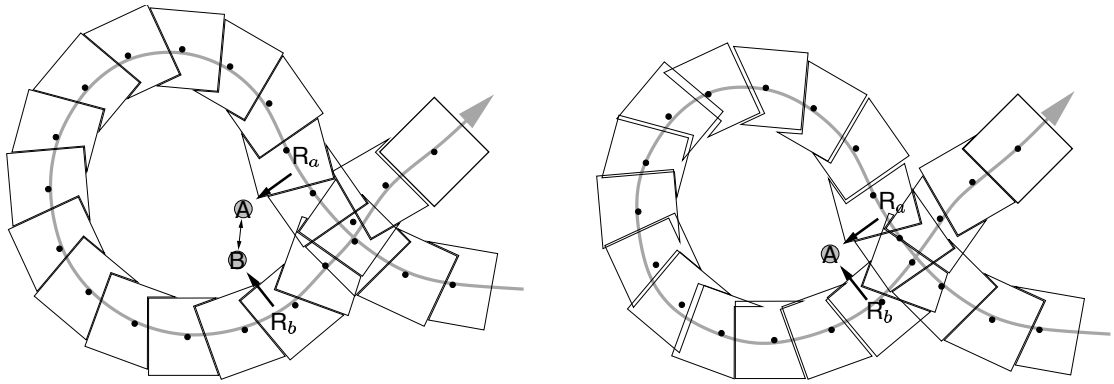

FIGURE 10: *Schematic representation of the map structure. The long arrow represents a fictitious trajectory, the small squares represent local maps that do not overlap, that produced by the fusion of only a few local perceptions. When the trajectory crosses itself, it is possible to apply a deformation algorithm to the local maps frames (right), that back-propagates the position correction estimated by a localization algorithm (that matched the A and B landmarks in this case).*

provide a correction on the current position, as well as on previous positions (typically using a SLAM-based localization algorithm – section 4.3), it is then easy to correct the position of each local map and generate a globally consistent model, with very few information lossage. This map structure is therefore able to bear with large localization corrections, and uses much less memory space than an approach in which all the acquired data would be memorized.

4 Localization

For long range autonomous navigation in planetary environments (and more generally for any mobile robot), the ability for the rover to localize itself, *i.e.* to estimate its position and the precision on this position, is *essential* for various reasons:

- The missions to be achieved by the rover are often expressed in localization terms, explicitly (*e.g.* "reach that position", "explore this area"...) or more implicitly (such as in "return near the lander" when it is out of sight).
- Long range navigation calls for the building of *global maps* of the environment, to find trajectories or paths and to enable mission supervision. The *spatial consistency* of such maps is required to allow an efficient and robust behavior of the rover: it is the knowledge of the rover position that guarantees this consistency.
- Finally, the proper execution of the geometric trajectories provided by the path planners calls for the precise knowledge of the rover motions.

Robot self-localization is actually one of the most important issue to tackle in autonomous navigation: this is confirmed by the vast amount of contributions to this problem in the literature, and the variety of approaches. In the absence of any external structure (*e.g.* radioed bearings), we are convinced that no single localization algorithm that uses on-board data can be robust enough to fulfill the various localization needs during long range navigation: a set of *concurrent and complementary* algorithms have to be developed and integrated for that purpose, which raises specific integration problems (section 7). We summarize here three localization methods, and conclude the section with a synthesis on the localization problem.

4.1 Odometry on Natural Terrain

Odometry is one of the most straightforward and "cheap" way to localize a robot. However, if it is a quite reliable way to localize a robot evolving indoor, it is much less precise on natural

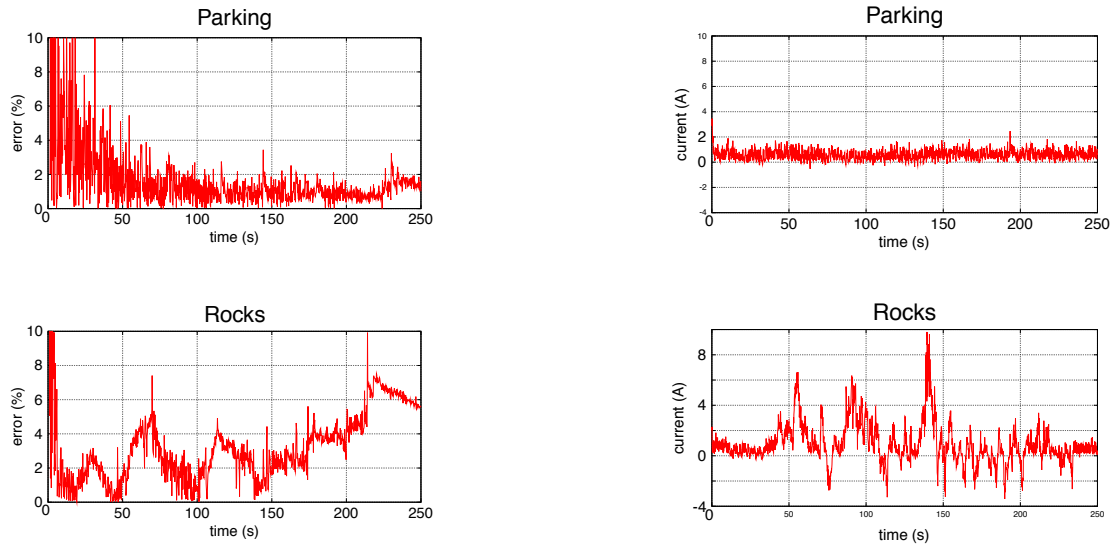

FIGURE 11: *Odometry behavior on two different kinds of terrain. The left images show the relative error, computed using an RTK centimeter precision differential GPS as a reference (error values are quite big during the first tens of seconds because of the GPS noise). The right images show the consumed current during the same trajectories. On a flat parking lot (top figures), both the error and the consumed current are stable, whereas on a rocky rough area (bottom), the figures are much more erratic (note the scale change in the consumed current curve). There are of course no strict correlation between the consumed current and the error growth, but monitoring these currents along with the individual wheel speeds would allow to detect slippages or other situations in which odometry yields gross errors.*

terrains. Not only the trajectories are three dimensional (which calls for at least inclinometers – or gyros if the rover dynamics voids their use), but also the slippages that often occur quickly lead to very erroneous position estimates: they especially affect the rotation estimates, which is all the more true when skid-steering is used to turn the rover. This is why in most of the contributions related to odometry in natural terrains, a rotation or orientation sensor is added (*e.g.* a gyrometer in [19, 53], or a sun tracker in [60]).

With the robot Lama, we use the attitude informations provided by a 2-axes inclinometer and a fiber optic gyrometer. Depending on the terrain, the position estimation results can be as good as 2% on flat lawn for instance, or much worse, when climbing rocky slopes for instance (figure 11).

An interesting issue here is to attempt to *qualify* on-line the motion estimates derived from odometry. For instance, since the 6 wheels of the robot Lama are equipped with very precise odometers, the study of the relative speeds of the wheels, as well as the current used to drive the motors[3], can help to detect slippages (figure 11). Such an analysis would however remain qualitative, and could to the best only detect gross errors. Although odometry will hardly ever be useful over long ranges, any progress in using such data is of course welcomed.

4.2 Visual motion estimation

There are few contributions that directly use the range data to produce a position estimate, without any segmentation or data structuration. The problem comes from the difficulty to establish matches between 3D points perceived from different rover positions. But when range data is produced by stereovision, data associations can be done in the video images, thanks to usual pixel tracking algorithms. We developed such a technique (figure 12), that is able to estimate the 6 parameters of the robot displacements in any kind of environments, provided it is textured enough so that pixel-based stereovision works well ([38] – a similar algorithm can be found in [49]).

We paid a lot of attention to the selection of the pixels to track: in order to avoid wrong correspondences, one must make sure that they can be faithfully tracked, and in order to have

[3]The chassis configuration parameters should also be taken into account.

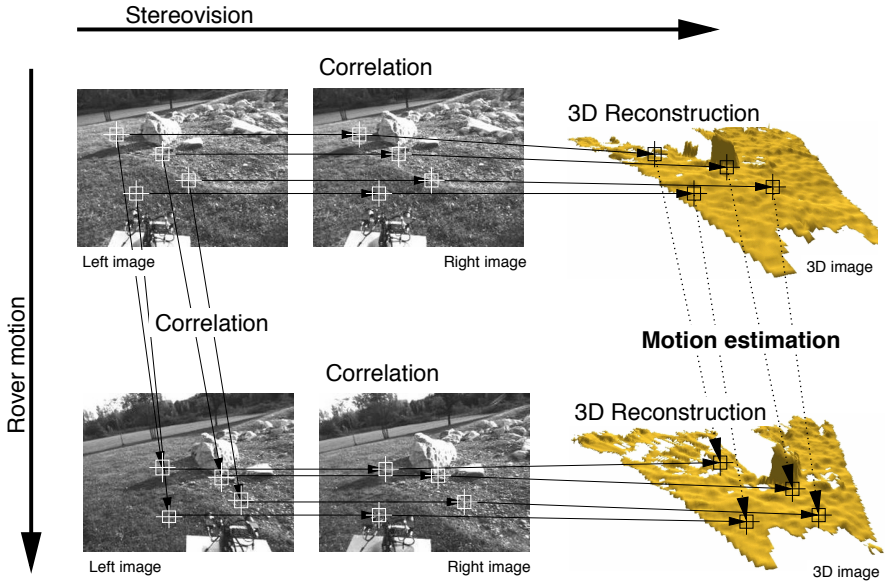

FIGURE 12: *Principle of the visual motion estimation technique based on stereovision and pixel tracking.*

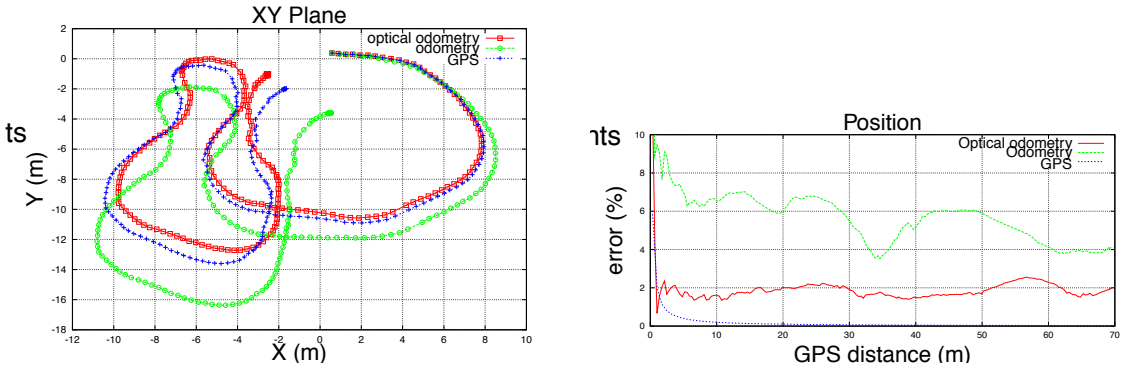

FIGURE 13: *Comparison of the error on the translations estimated by stereovision and pixel tracking (denoted "optical odometry") and odometry, during a 70m long run that includes sharp turns, hill climbing and rocky area traverses. The trajectory is shown on left, and the error on right.*

a precise estimation of the motion, one must choose pixels whose corresponding 3D points are known with a good accuracy. Pixel selection is done in three steps: an *a priori* selection is done on the basis of an error model of the stereo algorithm; an empirical model of the pixel tracking algorithm is used to discard the dubious pixels during the tracking phase; and finally an outlier rejection is performed when computing an estimate of displacement between two stereo frames (*a posteriori selection*).

We evaluated the algorithm on many runs (totalizing several hundreds of meters): it gives translation errors of about 2% of the distance (figure 13). Work related to this algorithm is still under way, to attempt to minimize the error growth by combining various elementary motions estimates computed over several frames. Also, a realistic error model is required to fuse the estimates with other localization processes: we are investigating a way to produce a covariance error matrix along with each elementary estimate, exploiting the error model of stereovision [38].

4.3 Long term localization

Odometry (or inertial navigation) and visual motion estimation cumulates errors over time or distance, and thus lead to an unbounded error growth as the rover navigates: localization methods

FIGURE 14: *Landmarks (black + signs) detected on a locally built digital elevation maps, and re-projected in the camera frame. Landmarks are here from 3 to 10 meters away.*

that memorize, match and exploit environment features are of course required to tackle long range navigation.

Landmark based localization One of the most widespread way to localize a robot is to detect and model landmarks in the environment, and refine both their position and the robot position during navigation using an extended Kalman filter (techniques often referred to as Simultaneous Localization And Mapping – SLAM, or Concurrent Mapping and Localization – CML). These techniques have now reached a mature level of achievement, and help to localize many robots in indoor or outdoor structured environments. However, their application in natural unstructured environments is not straightforward. Indeed, if the consideration of 3D rover motions (*i.e.* 6 state parameters) is not a problem, the landmark extraction and modeling from the acquired data is much more challenging: unstructured environments do not contain features that can easily be represented by geometric equations (*e.g.* lines or planes), with a covariance matrix associated to their parameters.

One of our first work in rover navigation has been devoted to this problem, using obstacle peaks as landmarks (a landmark is here a single 3D point), detected thanks to application of gradient operators in depth images, and an extended Kalman filter to refine the state vector [8, 9]. Results were satisfactory, but not applicable in every kind of terrains, as the peaks had to be clearly separated from the ground to be faithfully detected. More recently, we developed an algorithm to extract *local peaks*, that relies on the computation of similarity scores between a local DEM area and a pre-defined 3D peak-like pattern (a paraboloid for instance), at various scales (figure 14). Also, we investigated Set Theoretic approaches to tackle the state estimation problem [39]. Within such an approach, the only assumption made on the landmark positioning errors is that they are bounded – an assumption much easier to satisfy than in a Kalman filtering approach, where the errors distribution must be Gaussian. Estimates of the robot and landmarks positions are derived in terms of *feasible uncertainty sets*, defined as regions in which the robot and the landmarks are guaranteed to lie. These algorithms have however not been integrated on board Lama.

Using DEMs Most of the attempts to localize a rover in unstructured environments rely in the use of a DEM. Two approaches can be distinguished here: the DEM is either used to extract particular geometric features, as in figure 14 or in [33, 50, 25], or directly matched with the current perception (iconic approaches [51]). With our DEM building algorithm presented in section 3.2, we tried to localize the rover on the basis of a local/global DEM correlation, using a simplex algorithm to drive the search within the 6-dimensional state space. In term of precision, the results are however less precise than with the visual estimation technique in the case of successive perceptions, and when crossing a previously modeled area after a large loop, the method often falls in local minima, giving erroneous results. We think that more work has to be performed in this area, and that it certainly requires a more rigorous management of the uncertainties in the DEMs.

Using panoramic images More and more contributions related to robot localization using panoramic images are published in various robotics conferences (see *e.g.* [27, 40, 41]): panoramic sensors are actually making a real breakthrough in robotics. Their geometric properties are well understood [6], and a lot of catadioptric devices are now available on the market. Their main interest is that they can be used to tackle the *place recognition* problem using image indexing techniques, and thus bypassing the difficult landmark extraction step for that purpose. We recently developed such a method [18], using image histograms computed on local image characteristics to compute a distance between images acquired as the robot moves. It turns out that there is a relation between this distance and the Euclidean distance that separates the image acquisitions, and first results to match panoramic images are very promising. One must however keep in mind that such an approach is limited to a *qualitative* position estimate, that can therefore hardly be fused with other estimates. The identification of features in the environment is still required to refine this first estimate: interest points detected in the panoramic images are worth to look at, as they can fed localization algorithms based on bearing angles [37].

4.4 Synthesis: a classification of localization methods

We propose here a typology of approaches to localization in robotics, in order to have a better understanding of the problem, and to see its implications in the whole navigation system. This typology consists in three classes:

1. **Motion estimation**: we put in this class all the techniques that localize the rover *without memorizing any data*: they measure speeds or elementary motions, that are integrated over time to produce a position estimate. Odometry, inertial navigation and visual odometry belong to this class.

2. **Position refinement**: in this class are gathered all the algorithms that estimate the rover position (or correct an estimate) using environment models *built with the data acquired on board*. All the landmark, view and map based algorithms fit in this class.

3. **Absolute localization**: this last category contains the techniques that aim at localizing the robot with respect to an *initial global model* of the environment (such as images or DEMs derived from orbital or aerial imagery).

These three classes are not only according to the kind of data they use (with or without memory, with or without initial data), there are actually more criteria that lead us to establish such a classification: the evolution of the error on the position estimate (probably the most important one), the frequency of process activation, the level of abstraction of the data used, and the necessity to control the data acquisition and the rover motions. Table 15 summarizes these criteria.

	Motion estimation	Position refinement	Absolute localization
Error evolution	Unbounded growth	Unbounded growth but can decrease	Bounded
Process frequency	Very fast	Slow	Very slow
Data abstraction level	Raw data	Landmarks, envt. models or indexed images	Landmark or environment models
Control	No	Data acquisition control, influences motion generation	Drives the motion generation

FIGURE 15: Behavior of the four criteria in our typology of localization algorithms.

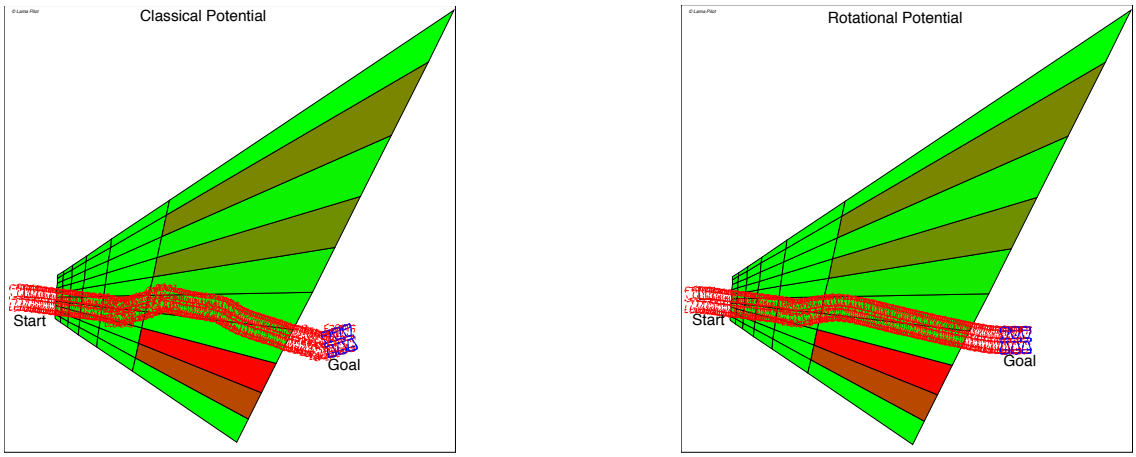

FIGURE 16: *Motions generated with a classical repulsive potential definition (left), and with the rotational potential definition (right) – for comparisons purposes, the motions are generated here across a sole obstacle map, until the goal is reached: in the real case, a new map is continuously generated as the robot moves.*

We are convinced that a complete navigation system should be endowed with at least one instance of each class (and actually several instances of concurrent algorithms for each class), that are complementary: motion estimation algorithms are directly used at the trajectory control level, and their result is used as initial estimates for position refinement algorithms. Position refinement algorithms are required to build consistent global terrain maps, which are the essential representation on which localization with respect to an initial terrain model can be performed.

5 Trajectory generation

A generic trajectory planner able to deal with any situation should take into account all the constraints, such as rover stability, rover body collisions with the ground, kinematic and even dynamic constraints. The difficulty of the problem calls for high time-consuming algorithms, which would actually be quite inefficient in situations where much simpler techniques are applicable. We therefore think it is worth to endow the rover with various trajectory generation algorithms, dedicated to the kind of terrain to traverse. We briefly describe here three motion algorithms that have been integrated on board Lama. Section 8 describe how they are actively triggered and controlled.

5.1 On easy terrains

On easy terrains, *i.e.* rather flat and lightly cluttered, dead-ends are very unlikely to occur. Therefore, the robot can efficiently navigate and avoid the obstacles just on a basis of a goal to reach[4] and a terrain model that only exhibits non-traversable areas (the probabilistic qualitative model in our case).

Potential field approach When the environment only contains few obstacles, reactively generating motion commands on the sole basis of a local obstacle detector can be an efficient enough way to navigate. Our first experiments with Lama, back in 1997 [20], have been achieved with such a method: elementary motion commands are generated thanks to potential fields defined on *(i)* a local qualitative map and *(ii)* on the result of a visual goal tracker. No global localization is required, and no global environment model is updated.

Artificial potential field approaches are well known in the roboticists community, but only few attempts in the context of natural environment have reached effective achievements [19]. We adapted a method formerly developed in the context of indoor robots [29] to the probabilistic

[4]Not necessarily the distant global goal, it can be a formerly selected sub-goal - see section 8.

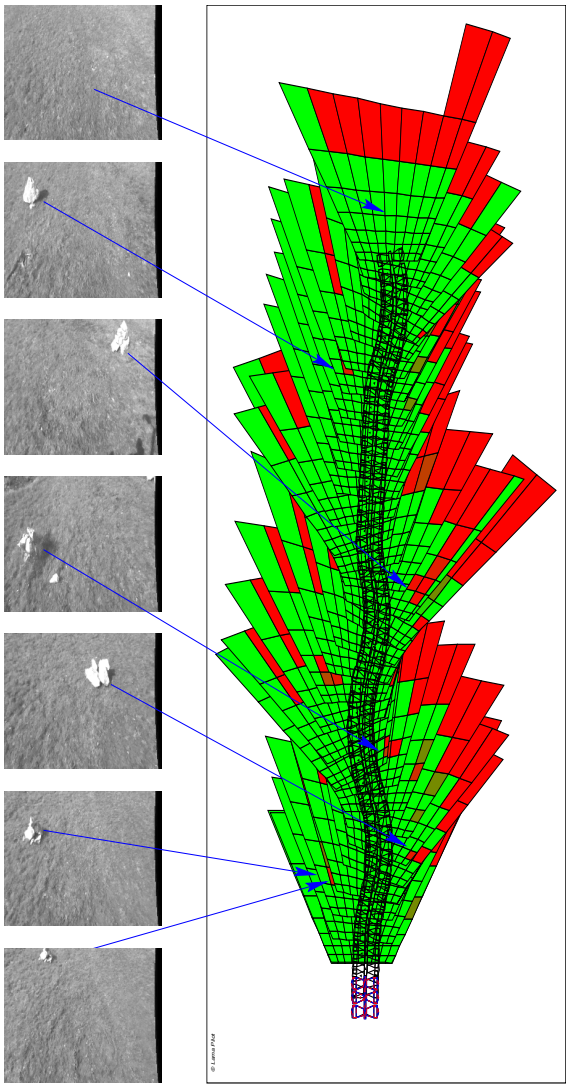

FIGURE 17: *A typical run in a lightly cluttered terrain with the potential field approach. The local qualitative maps are here surimposed, but only one map is known at a time by the rover.*

polygonal representation produced by the terrain classification algorithm. The method relies on repulsive *rotational potential functions*, which depends not only on the distance of the obstacles (the map cells), but also on the relative robot/obstacle cell configuration. In particular, the robot is not influenced by the presence of obstacle segments parallel to its moving direction. This leads to the generation of motions much smoother than with a classical "mass attraction" potential fully determined by the distances to the obstacles (figure 16). Also, the partial probabilities of each cell to contain an obstacle are taken into account in the definition of the potential field [20].

This approach gives satisfactory results in very clear terrains (figure 17), but the various parameter tuning is a bit tedious is one wants to cross more cluttered areas – the usual difficulties with potential field approaches. In particular, the method is very sensitive to a threshold that specifies the maximal influence distance of obstacles (*i.e.* the distance over which their potential is null), which is required to avoid potential wells.

Arcs approach We also developed an algorithm that evaluates circle arcs on the basis of the global qualitative probabilistic model, similarly to [36]. Every cycle, the algorithm is run on the updated terrain model: it evaluates the interest (in terms of reaching the goal) and the risk (in

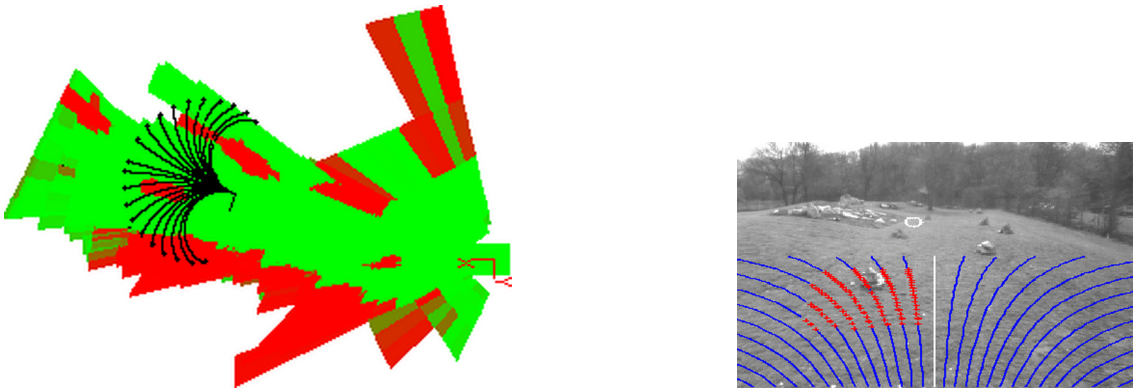

FIGURE 18: *A set of circle arcs evaluated on the global probabilistic model (left), and re-projected in the current camera view, with the chosen arc shown in white (right).*

terms of terrain traversability) of a set of circle arcs, and the arc that maximizes the interest/risk ratio is chosen (figure 18).

This approach yields more satisfactory behaviors (especially more stable) than the potential field approach. This is not surprising, not only the informations used are the result of several data acquisitions, but also the arc length considered makes the algorithm resemble more a trajectory planning algorithm, giving it a "look-ahead" possibility, whereas the potential field approach is a purely reactive one.

5.2 On rough terrains

On uneven areas, the notion of obstacle clearly depends on the capacity of the locomotion system to overcome terrain irregularities. This capacity can be expressed as a set of constraints to satisfy: stability constraints, ground/rover body collision constraints, and internal configuration constraints when the chassis is articulated. We developed an algorithm that evaluate elementary circle arcs (in a similar way to section 5.1), that takes into account the various constraints on the basis of the digital elevation map. This algorithm is a stripped down version of a formerly developed planner [21] (similar contributions to the problem can be found in [55, 16, 13]).

First, a set of nested arcs is produced, as shown on figure 19. The nesting depth is configurable, and depends on the desired ratio of ń reactivity ż vs. ń planning capabilities ż. The number and the length of the arcs are also important parameters. With Lama, we empirically observed that a nesting depth of 2 (as shown on the figure), with twenty $2m$ long arcs for the first depth level and ten $2m$ long arcs for the second depth gives good results.

The evaluation of the various arcs is done as follows: for each arc, a discrete set of regularly spaced positions are checked. At each position, the chassis attitude and internal configurations, in total five angles α_i, are predicted thanks to a geometric placement function whose principle is depicted on figure 20. The predicted configuration is then used to compute a "dangerousness" value for the considered position: to each configuration angle α_i is associated an upper limit m_{α_i}, which is related to a physical constrain the robot *must not* exceed. We choose a danger function d_{α_i} proportional to $m_{\alpha_i}/(\alpha_i - m_{\alpha_i})^2$, so that small configuration angles (close to a flat ground case) lead to small danger and bigger angles lead to a quadratically growing danger. The global danger of a position is defined as the maximum danger d_{α_i} computed for the five configuration angles.

Finally, to each considered arc is associated a *cost* that integrates the elementary costs of the successive configurations it contains, and an *interest*, which is the Dubbins distance to reach the goal from the arc end. The arc to execute is the one that maximizes the interest/cost ratio, and an A* algorithm is used to efficiently explore the tree formed by the nested arcs and the considered configurations.

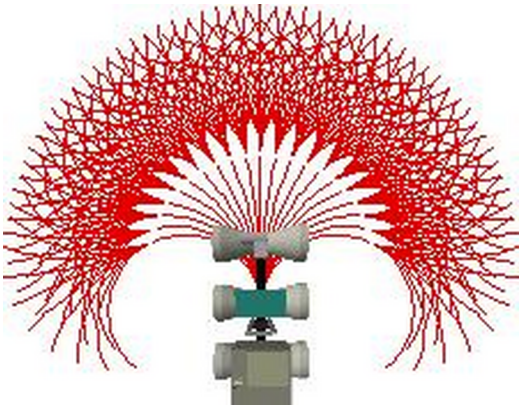

FIGURE 19: *A sample set of arcs used by the rough terrain motion generation method (20 first depth arcs with 10 second depth arcs are shown here).*

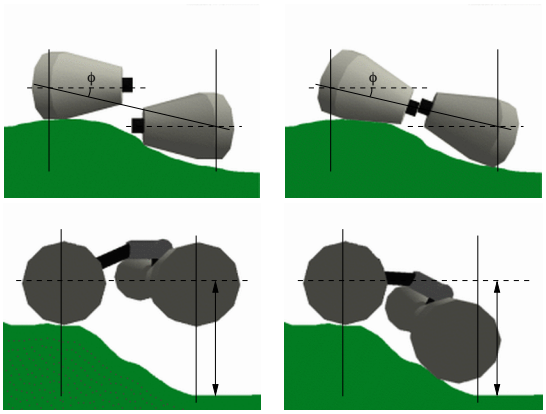

FIGURE 20: *Illustration of the geometric placement procedure. First, given an horizontal position x, y, θ on the DEM the roll and height of each axle is iteratively computed (top). The constrains between the three axles of Lama are then taken into account and used to compute the global placement of the rover (bottom).*

FIGURE 21: *A predicted placement of the robot on the DEM (left), and the corresponding real placement on the terrain (right).*

5.3 Motion control

Whatever the method is used, the execution of the generated trajectory relies on the same procedure: each elementary geometric movement is translated into a linear and angular speed couple, according to a specified speed. This reference is periodically updated and transmitted to a PID based servo-controller, that determines a reference rotation speed for each of the 6 wheels (the wheels being themselves servoed on the optical encoders by a PID). This simple motion control algorithm is satisfactory in most of the cases, when the rover is not evolving in rough terrains. But in the latter case, more sophisticated motion control algorithms should be used, to avoid side-slippings, skids, grip loss... Detecting faulty locomotion behaviors is of course a very important problem to tackle to traverse rough terrains. This is actually related to the monitoring of odometry mentioned in section 4.1, and also requires the estimation of motions with exteroceptive means as in section 4.2. Several innovative locomotion concepts are regularly proposed in the literature (*e.g.* [56, 31]: most of the times, they raise very challenging control issues.

Part B: Integration

6 Integration and Software Architecture

As emphasized in the first part of the paper, endowing a robot with autonomy requires the integration of a large variety of heterogeneous and sometimes concurrent functionalities: sensors and actuators control, servo-controls, monitoring, image processing, terrain modeling and localization algorithms, trajectory generation algorithms... In this section, we briefly present how we tackle this functional integration in our lab, and in particular with Lama.[5]

6.1 Problem specifications

Co-integration One of the main problem raised by the integration of these various functionalities comes from their variety. They have been developed by experts from various domains (control theory, vision, motion planning...), not necessarily aware of the robot software architecture, or even of its operating system. Moreover, these functionalities have various temporal and logical characteristics, and must share common CPU resources. Some of them have strong real time constraints and require secure execution contexts (*e.g.* servo-controllers or monitoring processes), whereas others can require an unpredictable amount of CPU time and/or memory. Finally, the system must be very flexible and open, allowing to easily add, remove or modify functionalities, according to the application needs, the demonstration that has to be illustrated or any additional development or modification one want to integrate.

Controllability and data flow The functionalities must not only coexist: they may also have to *cooperate*, *i.e.* exchange data or even synchronize their execution. Once integrated on board, they also have to be controlled according to the mission and the current execution context. They must therefore offer mechanisms that allow to parameterize, start, interrupt and start them again upon requests (control flow), and that defines the way they are fed them with adequate input data and the way they export processed data (data flow).

6.2 Approach: a network of modules

In order to master all these integration problems, we have specified the concept of a *module* as follows: a module is an active software entity that can encapsulate almost every kind of algorithm (periodic, aperiodic, synchronous or asynchronous, interruptible or uninterruptible). It is a server that manages all the communications with the others modules, that runs the functional algorithms

[5]The system presented here is a part of the definition of a generic software and decisional architecture for autonomous machines [1], which has been successfully instantiated with indoor mobile robotics experiments, autonomous satellite simulations and multi-robot cooperation experiments.

when required, and that paces the algorithms using its own threads or processes. From an architectural point of view, a module gathers a set of services, *i.e.* processes related to given resources. A resource may either be physical or logical: it can be a sensor, an effector or a data structure (position, image, map, trajectory...). In this way, the robot functional level is organized as a network of modules. Figure 23 presents the modules of Lama corresponding to the functionalities described in the first part of the paper, that are summarized in the table 22.

	module	description	paragraph
hardware controllers	PTU	Pan-tilt unit controller	§ 2
	CAMERA	Cameras controller (settings + acquisition)	§ 2
	GPS	Global Position System (for validation)	§ 2
	LOCO	Odometry and servo-control (speed or position)	§ 4.1 and 5.3
localization	Steo	Stereo-based "odometry"	§ 4.2
	PoM	Position manager	§ 7
	Stereo	Stereo vision (correlation procedures)	§ 3
mapping	DEM	Digital Elevation Map	§ 3.2
	Classif	Terrain classification	§ 3.1
motion generation	Potential fields	Potential fields' based motion	§ 5.1
	Arc	Arcs' based motion	§ 5.1
	P3D	3D local planner	§ 5.2
	Goal tracker	Camera tracking on visual target	*not presented*

FIGURE 22: *The main modules on-board Lama*

It is important to note that the presence of a module in the robot functional level does not presuppose its use. It offers a set of services that can or can not be used in a given experiment, according to current needs. A module does not decide by itself to start a service: its services are dynamically started, interrupted or (re)parameterized upon asynchronous client requests, based on a nonblocking client/server protocol. For each request, it returns to the client a final reply that qualifies how the service has been executed.

6.3 Automatic modules generation

Every module of the functional level is a specific instance of the generic module model, so that the behavior and the interfaces of the module operational functions obey standard specifications, which helps the systematization of the integration of several modules. A programmer could write by himself a module on the basis of this generic model, it would be a fastidious operation, not to mention the risk to not comply with the underlying operating system.

So we have developed a tool, GenoM (Generator of Modules), that automatically produces a new module according the generic module specifications from a synthetic description. The description does not depend on the operating system and does not require any knowledge on distributed systems. It is elaborated using a very simple language (in fact it is a form to fill) that allows to declare the services and to specify their parameters, the possible qualitative results, the exported data, the temporal and logical characteristics, etc.

On the sole basis of this formal description, GenoM produces:

- a module that can run under various operating systems (currently VxWorks, Solaris and linux);
- a set of interface libraries to communicate with the module using various programming languages (currently C, Tcl/Tk and Propice – a procedural reasoning system);
- an interactive program for preliminary test purposes.

This approach allows the programmer to focus on the developments of its algorithms, without caring about the robot operational system and architecture design. The automatic production of

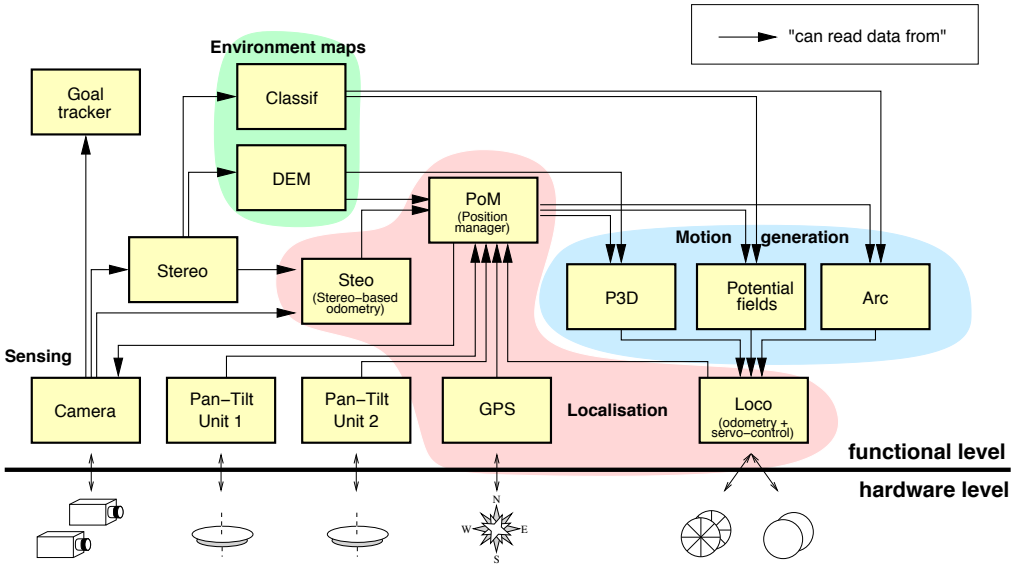

FIGURE 23: *The network of modules on board Lama. Near to the hardware devices, at the bottom, are the modules that manage the sensors and the actuators like the cameras (CAMERA) or the pan-tilt units (PTU) or servo-control the motor-wheels (LLOCO), etc. On the basis of data or services offered by these basic modules, other modules can estimate the robot position (STEO), compute maps (DEM), or produce references for collision free motions (ARC, P3D). The arrows represent all the possible data flow that can be set up from a module to another.*

the modules guarantees a standard behavior and avoids the execution of tedious logical tests. The standardization associated to the formal description allows to easily develop a complex functional level, in which several tens of modules can be integrated. This system has been systematically used for all our indoor and outdoor robots for over 5 years, and has been validated in large scale projects.

6.4 Modules control and interface

All the modules integrated on the robot form the functional level: they offers a set of services that can be activated upon requests sent either by an operator or by a software supervisor. This is enabled by the interface libraries automatically generated by GenoM, that include all the functions required to send a request, receive a reply or read a data produced by the module.

Using Tcl scripts A very simple and straightforward way to control the robot consists in loading the Tcl library produced by GenoM in a Tcl shell interpretor, allowing the operator to interactively activate the services. Of course, more or less elaborate functions or macros can be written to sequence the various services. For instance, the 3 following Tcl procedures allow to reach a goal using the arc-based approach on the qualitative model:

1. Qualitative model updating with `updateClassifMap`. This procedure consists in sequentially acquiring a stereo image pair, run the stereovision algorithm, classify the produced 3D image and fuse it in the global model.

```
# procedure to update the classification map
proc updateClassifMap { } {

    # get a stereo image from bank A: request "OneShot" to module CAMERA
    camera::OneShot CAMERA_BANK_A CAMERA_STEREO

    # stereo correlation : request "SCorrel" to module STEREO
    stereo::SCorrel

    # classification and fusion : requests Classif and FuseLCLMap to module CLASSIF
    classif::Classif
    classif::FuseMap
}
```

2. Generate motions with `arcStep`. This second procedure computes speed reference for the robot using the ARC module. The result of the `arc::GetArc` request is evaluated to know whether the goal has been reached or an error has occurred. If not, the speed references chosen are sent to the LOCO servo-controller module (`loco::GotoSpeed` request).

```
# procedure to compute one reference using ARC module
proc arcStep { } {

    # Choose best arc
    set retCode [catch {
        set arc [arc::GetArc -raw $::lamaSpeed]
        set v [lindex $arc 2]
        set w [lindex $arc 3]
    } message]

    # test end
    if { $retCode } {
        switch -regexp $message {
            "GOAL_REACHED" {
                set ::goalReached 1
            }
            default {
                set ::arcFailed 1
            }
        }
    # servo the robot
    } else {
        loco::GotoSpeed $v $w
    }
}
```

3. Main procedure `arcRun`. Finally, the main procedure loops overs the previous `updateClassifMap` and `arcStep` procedures.

```
# Main function to run
proc arcRun {xGoal yGoal} {

    # Some initializations
    set ::goalReached 0
    set ::arcFailed 0
    set ::lamaSpeed 0.08
    arc::SetGoal $xGoal $yGoal

    # Main loop
    while {$::goalReached == 0 && $::arcFailed == 0} {
        updateClassifMap
        arcStep
    }

    # goal reached ?
    if {$::goalReached} {
        loco::Stop
        puts "Goal reached. Alleluyah !"
    }

    # motion mode not adapted ?
    if {$::arcFailed} {
        loco::Stop
        puts "Arc failed. Call P3D ! !"
    }
}
```

The data produced by the modules (images, robot positions, maps, trajectories, etc) can also be directly read via Tcl using GenoM functions and displayed in the Tk graphical environment. This allows to conceive relatively easily graphical control interface (see *e.g.* figure 28).

Control using Propice The development of complex integrated experiments may require an elaborated on-board software supervisor (possibly coupled with a task planner). For instance, our service robot experiments are scheduled and supervised by Propice, a procedural reasoning system [3, 2]. Some demonstrations with Lama have been integrated with this tool: figure 24 presents a graphical Propice procedure that implements a function similar to the Tcl one presented just above. The graphical representation allows to easily express parallelism and synchronization.

More generally, Propice is composed of a set of tools and methods to represent and execute plans and procedures. Each plan/procedure is self-contained: it describes in which condition it is applicable and the goals it achieves. This is particularly well adapted to context based task refinement and to a large class of incremental robot tasks, such as adaptive rover navigation in a partially known environment.

7 Integration of Localization Algorithms

Some particular problems are raised by the integration of several concurrent localization algorithms on board the robot. To tackle this in a generic and reconfigurable way, we developed a particular module named PoM (POsition Manager). This module receives as inputs all the position estimates produced by the localization algorithms (each of them being itself integrated within a specific *module*), and produces a single consistent position estimate as an output. PoM addresses the following three issues: the *sensor geometrical distribution*, the *localization module asynchronism* and the *fusion of the various position estimates*.

7.1 Sensor geometrical distribution

Sensors are usually physically distributed over the robot, and their relative positions can dynamically change if they are mounted on orientable devices, or if the chassis is articulated. These

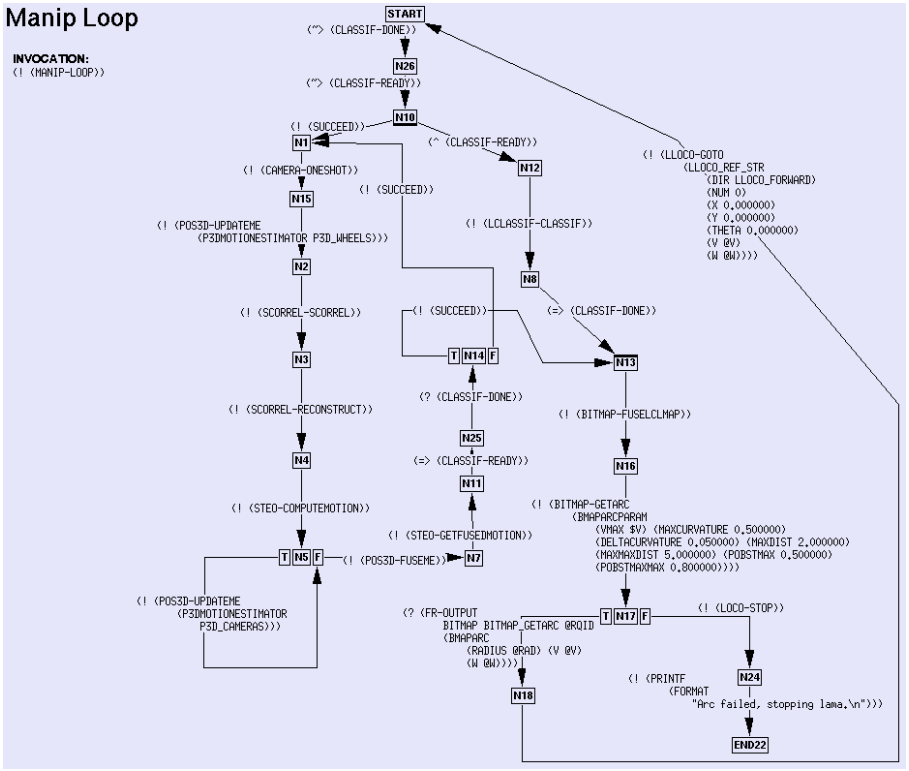

FIGURE 24: *The Propice procedure to reach a goal using the arc-based approach on the qualitative model.*

position informations are of first importance for the algorithms that process the data relative to the environment (raw sensor data as well as processed data). For instance, the vision-based localization module must know the orientation of the cameras for each image it processes, whereas the DEM module needs the relative and/or absolute position of the 3D images it fuses.

Distributing the geometrical information within the modules is not satisfying, as it can lead to consistency problems (synchronism), not to mention the portability issues when linking the module to an other sensor or an other robot. It is indeed much preferable to keep the modules as generic as possible. For that purpose, we use a centralized geometrical description of a robot, which provides the necessary frame coordinates to any module when the robot navigates. All the data acquisition modules use this information to "tag" the data they produce.

Geometric relations among sensors Upon startup, PoM reads a configuration file which is the textual description of a geometrical graph (figure 25). The nodes of the graph are frames coordinates that needs to be exported: they usually correspond to sensors locations but can also be a convenient way to split otherwise complex links. Frames are connected one to another with either *static* (rigid) or *dynamic* (mobile) links. A static link cannot change during execution and is usually related to some mechanical part of the robot. On the other hand, dynamic links depend on the chassis configuration or on a particular actuator configuration (for instance a PTU), and are updated continuously at a predefined frequency. Finally, the configuration file defines a *main* frame, in which the data acquired by modules are always expressed, which facilitate and homogenize inter-module data transfers.

Dispatching the geometric relations Every data acquisition component can then read the frame which it relates to, and associate to its data a "tag" structure which contains at least the time of acquisition, the configuration of the sensor relative to the "main" frame and the current global robot position estimate. Thanks to this tagging, data acquisition modules only have to know the name of the frame they are connected to, which can be dynamically specified. Once

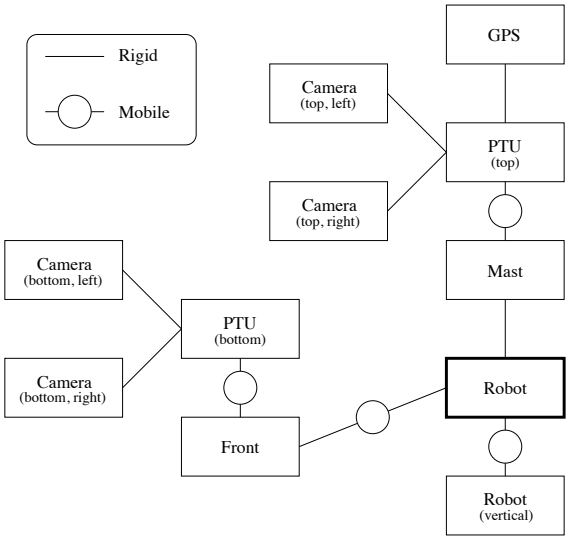

FIGURE 25: *The graph that represents the geometry of Lama. Rectangular boxes are frames configurations that are exported in the system. The thick box (*Robot *in the figure) is the* main *frame. Solid lines are links (either rigid or mobile) that are declared in the configuration file.*

tagging is done, clients using data do not have to care from where it comes, since all the necessary geometrical and time information is associated to the data itself. The tag is propagated along with the data between modules, thus making inter-module data communication very flexible.

7.2 Localization modules asynchronism

In addition to internal frame configurations, PoM collects the position produced by the various position estimators present in the system (see figure 23). It is the place where positional information is centralized and made available for any module that requires it.

The localization algorithms have individual time properties: some produce a position estimate at a regular high frequency, while others require some non-constant (and possibly long) computation time. Thus the robot has often to deal with outdated positions. To maintain the time consistency between the various position estimates, PoM internally maintains a time chart for each position estimator (figure 26), plus one particular chart for the *fused* position, which is the result of the fusion of every position estimator (see following section). PoM periodically polls every position estimator and look if a position estimate has been updated. When a new position is produced, it is inserted in the chart at the corresponding date (found in the position "tag"), and the *fused* position is then updated.

FIGURE 26: *Time management and internal representation of motion estimators in PoM (the position estimators shown here are only examples). Black dots are previously stored positions, and grey dots are new upcoming positions, produced at the the time labeled 'now', but corresponding to older dates.*

One important thing to note is that we cannot simply update the current fused position "as is" because it might be very different from the previous position. Since some module may use the robot position continuously (such as the locomotion module that servos the motions along a given trajectory for instance), such virtual jumps are not permitted. To handle this, we define a *reference position estimator* (usually, odometry is a good choice). The fused position computed by PoM contains two parts: the current position of the *reference* estimator, plus a *correction* of this reference. One can then use the *reference* position locally, and include the correction if an absolute position is needed.

7.3 Fusion of the position estimates

The two sections above describe a *structural* scheme to integrate various localization algorithms. But the most important thing related to this integration is the ability to *fuse* the various estimates, thus leading to a more precise position estimate. Among the various formalisms that can be applied to this problem, Kalman filtering is the most popular: this requires the knowledge of an error model for each localization algorithm, but also the ability to discard erroneous estimates (fault detection).

Up to now, our various algorithms are not "qualified", in the sense that their error model is not precisely known: the estimates are selected on the basis of a *confidence* (real value between 0.0 and 1.0). These confidence values are currently hard-coded, and computed from our experience of the estimators, but will eventually be dynamically set by each position estimator, on the basis of the qualification of the algorithms (*e.g* behavior of the currents consumed and the chassis configuration for odometry, or number of pixels effectively tracked and residual of the mean square estimator for the visual motion estimation technique).

8 Integration of several motion modes

The methods to generate motions presented in section 5 are all local: they evaluate elementary motions, and are therefore not efficient to deal with long range navigation, avoiding dead-ends and optimizing the whole trajectory length. In other words, they can successfully achieve way-point navigation, provided the way-points are given by an entity that is aware of the whole mission context (global environment models, possible a priori knowledge, mission definition with the associated constraints...). Also, each motion generator is suited for a given type of terrain: according to a general "economy of means" principle and for efficiency purpose, the right mode has to be triggered at the right time. The integration of these methods therefore calls for a specific decisional ability, which we refer to as the "navigation planner", that chooses, triggers and control them by providing the appropriate way-points (or sub-goals). Several contributions to rover navigation also tackle the navigation problem with a 2-level hierarchy, a global planner being in charge of generating way-points for a local trajectory finder [58, 57, 15].

A first solution Given the robot current position, the goal to reach and the global qualitative model that describe the terrain in terms of navigability areas, the navigation planner must find a sub-goal to reach, the motion mode to apply, and the perception tasks to execute. Indeed, the decisions related to motion and perception must be taken jointly, as they are strongly interdependent: executing a motion requires to have formerly modeled the environment, and to acquire some specific data, a motion is often necessary to reach the adequate observation position.

We proposed a first approach to this problem, using an optimal path search algorithm ($A^\star$) in the graph defined by the traversability regions of the global qualitative model [34]. Each region of the terrain model have been processed beforehand: a terrain class label has been decided, on the basis of the partial probabilities. The "optimality" criterion takes here a crucial importance: it is a linear combination of time and consumed energy, weighted by the class of the regions to cross *and the confidence of the terrain labeling* (figure 27). Introducing the labeling confidence in the crossing cost of an arc comes to consider *implicitly* the modeling capabilities of the robot: tolerating to cross obstacle areas labeled with a low confidence means that the robot is able to acquire easily informations on this area. The returned path is not executed directly, it is analyzed to define the sub-goal to reach, which is the last node of the path that lies in a surely labeled traversable area, and the area to perceive is the one that maximizes the coverage of the remaining path.

Current work The work briefly summarized above dates back to the middle of the 90's, and actually needs to be revisited. In particular, the $D^\star$ would be more performant in terms of optimal path computations [58]. But more fundamentally, we are currently investigating an approach within a decision theoretic framework, that would integrate the various informations in a more

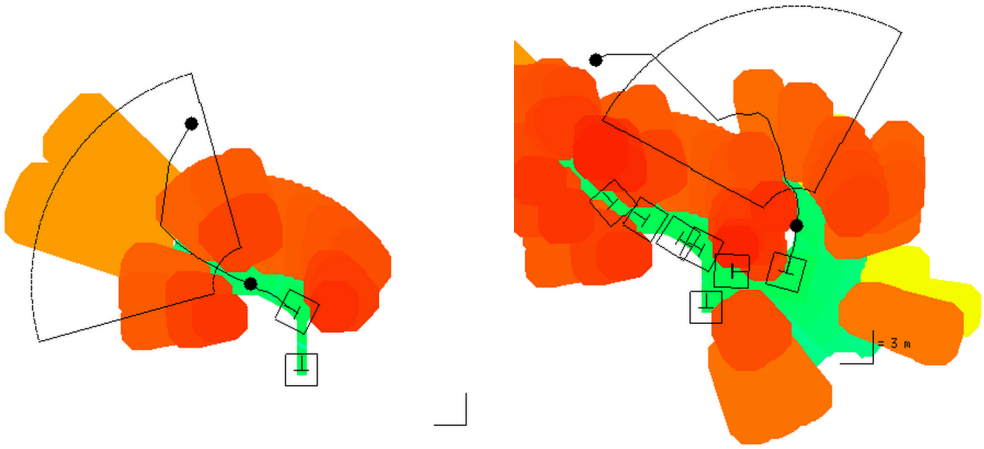

FIGURE 27: *Two results of the navigation planner on the qualitative model. The result of the analysis of the shortest path found can be interpreted as the answer to the question "what area should be perceived to reach the goal ?"*

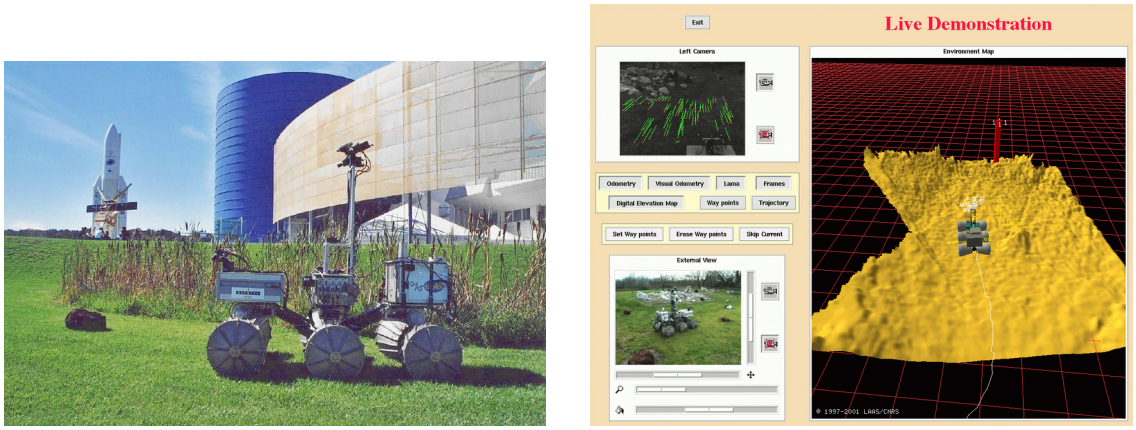

FIGURE 28: *Lama during a live demonstration at the "Cité de l'Espace" in Toulouse, Sept. 2000, and the graphical user interface used.*

rigorous way. The integration on the detected landmarks should also be taken into account, to generate path along which a good localization can be guaranteed.

9 Experiments

The experimental evaluation of the developments is a key point in robotics, to which we devote a lot of energy. The various functionalities described throughout the first part of this paper have all been tested on-board Lama, in various scenarios. Most of experiments have been conducted in our experimentation site, and live public demonstrations have been presented in September 2000 at the "Cité de l'Espace", a museum dedicated to space technologies in Toulouse (figure 28).

Up to now, no experiments integrating *all* the presented functionalities have been successfully performed (it requires in particular the mentioned developments at the navigation planner level). Each motion generation algorithm has been individually tested[6]: they are all able to reach sub-goals distant from ten to several tens of meters, when the terrain do not exhibits any dead-end.

[6]The potential field approach is not used anymore, as it gives trajectories less satisfying than the arc method.

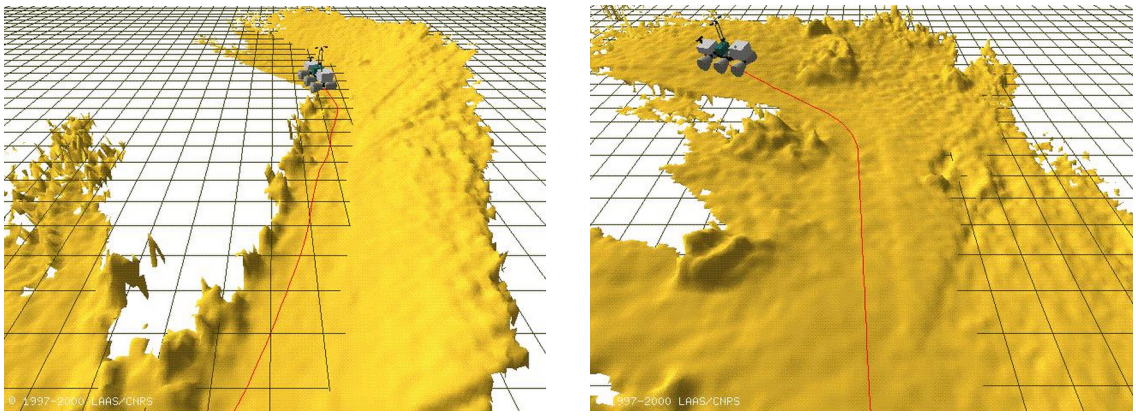

FIGURE 29: *Two autonomous runs of Lama. Left: live demonstration in the "Cité de l'espace" museum in Toulouse, Sept. 2000. Lama found its way trough a narrow corridor, 85 iterations of the navigation loop have been executed, the trajectory is about 40m long. Right: in our experiment site, Lama found a pass to climb a 2 meters high hill.*

The most complex integrated experiment we performed is the following : the two environment models (qualitative map and digital elevation map) are *continuously* updated every time new data are gathered, and two integrated localization algorithms (odometry and visual motion estimate) are also continuously running. The selection of the trajectory generation algorithm is the following: given a global goal to reach, the easy terrain algorithm is applied until no feasible arcs can be found. In such cases, the rough terrain algorithm is applied. It is run until either the easy terrain algorithm succeeds again, or until no feasible arcs are found in the elevation map. In the latter case (that can be assimilated to a dead end), the navigation planner is activated, to select a sub-goal to reach. The whole strategy is then applied to reach the sub-goal, and so on. Figure 29 presents two autonomous runs of a few tens of meter, in which the rover successfully found its way through various situations.

Performances The experiments are run in a continuous mode, all the algorithms being integrated on board the rover. The implementation of the algorithms is not optimized[7], and vision algorithms limits the rover speed to only $0.05m.s^{-1}$, as each loop takes about $10s$. During these $10s$, the following algorithms are run: stereovision, qualitative and digital maps building, motion generation and visual position estimation. Among these functions, the visual position estimation takes about $6s$, about 500 pixels being tracked: current work on this algorithm will allow to reduce by around one order of magnitude the number of pixels to track, thus allowing the robot to run at $0.10m.s^{-1}$.

10 Discussion

We insisted on the fact that to efficiently achieve autonomous long range navigation, various algorithms have to be developed for each of the basic navigation functions (environment modeling, localization and motion generation). Such a paradigm eventually leads to the development of a complex integrated system, thus requiring the development of integration tools, at both the functional and decisional levels. We are convinced that such tools are the key to implement efficient autonomy on large time and space ranges.

There are however several open issues:

- The whole navigation problem should benefit from the availability of an initial terrain map, such as provided by an orbiter, whose spatial consistency is guaranteed. For that purpose, the development of algorithms that match locally built terrain models with such an initial map is required. These algorithms should allow to guarantee a bound on the rover position

[7] in particular, the AltiVec high performance SIMD instructions of the PowerPC processors are not used.

estimate, and also yield the merging of the initial map and the gathered data into a common map structure, on which the navigation planner should reason [35].

- Even with an initial coarse map, the development of robust long range localization algorithms is really needed. More generally, the *qualification* of each individual localization algorithm is a pre-requisite to fuse their estimates. This is a difficult issue, as there is little hope that such models can be derived analytically, *i.e.* by explicitly propagating errors through the data processing: intensive experiments are necessary here to derive statistical error models.
- Rough terrain mobility raise very difficult problems from the locomotion control point of view. For that purpose, the underlying physics (ground wheel friction forces, robot dynamics) has to be considered at both the trajectory planning and execution levels.
- The navigation strategy level plays a crucial role, as it is in charge of most of the important decisions. As mentioned in section 8, the determination of the sub-goals to reach, the motion mode to apply, the area to model and the localization tasks to perform should be considered as a whole, within an unified decision theoretic framework.

Finally, one must always keep in mind that any functionality will eventually fail in some cases. We think that our architecture can bear with such behaviors, thus enabling the development of dependable long range navigation.

References

[1] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. An architecture for autonomy. *International Journal of Robotics Research*, 17(4):315–337, April 1998.

[2] R. Alami, R. Chatila, S. Fleury, M. Herrb, F. Ingrand, M. Khatib, B. Morisset, P. Moutarlier, and T. Simeon. Around the lab in 40 days... In *International Conference on Robotics and Automation*, pages 88–94, San Francisco, CA (USA), April 2000. IEEE.

[3] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert. Multi robot cooperation in the martha project. *IEEE Robotics and Automation Magazine*, 5(1), March 1998.

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

[5] *6th ESA Workshop on Advanced Space Technologies for Robotics and Automation*, Noordwijk (The Netherlands), December 2000.

[6] S. Baker, S.K. Nayar, and H. Murase. Paramatric feature detection. *International Journal on Computer Vision*, 27(1):27–50, 1998.

[7] J Bares, M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons, and W. Witthacker. Ambler: An autonomous rover for planetary exploration. *IEEE Computer*, pages 18–26, June 1989.

[8] S. Betge-Brezetz, R. Chatila, and M.Devy. Object-based modelling and localization in natural environments. In *International Conference on Robotics and Automation*, pages 2920–2927, Nagoya (Japan), May 1995. IEEE.

[9] S. Betge-Brezetz, P. Hebert, R. Chatila, and M.Devy. Uncertain map making in natural environments. In *International Conference on Robotics and Automation*, pages 1048–1053, Minneapolis, MN (USA), April 1996. IEEE.

[10] L. Boissier and L. Marechal. Rover demonstrator for moon exploration. *Autonomous Robots Journal*, 2(4):353–362, 1995.

[11] R. Chatila. Deliberation and reactivity in autonomous mobile robots. *Robotics and Autonomous Systems*, 16(2-4):197–211, 1995.

[12] R. Chatila and S. Lacroix. A case study in machine intelligence: Adaptive autonomous space rovers. In *1st International Conference on Field and Service Robotics*, Canberra (Australia), August 1997.

[13] 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.

[14] C. DeBolt, C. O'Donnell, C. Freed, and T. Nguyen. The bugs 'basic uxo gathering system' project for uxo clearance and mine countermeasures. In *International Conference on Robotics and Automation*, pages 329–334, Albuquerque, NM (USA), April 1997. IEEE.

[15] M. Delpech, L. Rastel, and M. Lamboley. Enhanced path planning and localization techniques for autonomous planetary rovers. In *5th ESA Workshop on Advanced Space Technologies for Robotics and Automation*, Noordwijk (The Netherlands), December 1998. CNES.

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

[17] O. Faugeras, T. Vieville, E. Theron, J. Vuillemin, B. Hotz, Z. Zhang, , L. Moll, P. Bertin, H. Mathieu, P. Fua, G. Berry, and C. Proy. Real-time correlation-based stereo : algorithm, implementations and application. Rapport de recherche 2013, INRIA, August 1993.

[18] J. Gonzalez and S. Lacroix. Rover localization in natural environments by indexing panoramic images. In *to appear in IEEE International Conference on Robotics and Automation, Washington, DC (USA)*, May 2002.

[19] D.N. Green, J.Z. Sasiadek, and G.S. Vukovich. Path tracking, obstacle avoidance and position estimation by an autonomous, wheeled planetary rover. In *International Conference on Robotics and Automation*, pages 1300–1305, San Diego, CA (USA), May 1994. IEEE.

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

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

[22] L. Henriksen and E. Krotkov. Natural terrain hazard detection with a laser rangefinder. In *International Conference on Robotics and Automation*, pages 968–973, Albuquerque, NM (USA), April 1997. IEEE.

[23] R. Hoffman and E. Krotkov. Perception of rugged terrain for a walking robot: True confessions and new directions. In *International Workshop on Intelligent Robots and Systems*, pages 505–510, Osaka (Japan), November 1991. IEEE.

[24] A. Howard and H. Seraji. Real-time assessment of terrain traversability for autonomous rover navigation. In *International Conference on Intelligent Robotics and Systems*, pages 58–63, Takamatsu (Japon), November 2000. JPL, IEEE.

[25] D. Huber, O. Carmichael, and M. Hebert. 3d map reconstruction from range data. In *International Conference on Robotics and Automation*, pages 891–897, San Francisco, CA (USA), April 2000. CMU, IEEE.

[26] *6th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, June 2001.

[27] H. Ishiguro and S Tsuji. Image based memory of environment. In *International Conference on Intelligent Robots and Systems*, pages 634–639, Osaka (Japan), November 1996. IEEE.

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

[29] M. Khatib and R. Chatila. An extended potential field approach for mobile robot sensor-based motions. In *4th nternational Conference on Intelligent Autonomous Systems, Karlsruhe (Germany)*, pages 490–496, March 1995.

[30] E. Krotkov and J. Bitch. The defense advanced resarch projects agency (DARPA) tactical mobile robot program. *International Journal of Robotics Research*, 18(7):769–776, 1999.

[31] T. Kubota, S. Sawai, T. Hashimoto, J. Kawaguchi, and A. Fujiwara. Robotics technology for asteroid sample return mission MUSES-C. In ISAIRAS [26].

[32] I. S. Kweon and T. Kanade. High-resolution terrain map from multiple sensor data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):278–292, February 1992.

[33] I.S. Kweon and T. Kanade. Extracting topographic features for outdoor mobile robots. In *International Conference on Robotics and Automation*, pages 1992–1997, Sacramento, CA (USA), April 1991. IEEE.

[34] S. Lacroix and R. Chatila. Motion and perception strategies for outdoor mobile robot navigation in unknown environments. In *4th International Symposium on Experimental Robotics*, Stanford, CA (USA), July 1995.

[35] S. Lacroix, I-K. Jung, A. Mallet, and R. Chatila. Towards cooperative air/ground robotics: issues related to environment modeling. In *10th International Conference on Advanced Robotics*, Budapest (Hungary), August 2001.

[36] 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, December 1994.

[37] Y-J. Lee and M-J. Chung. Absolute localization of a mobile robot with an omnidirectional vision sensor. In *10th International Conference on Advanced Robotics*, pages 529–534, Budapest (Hungary), August 2001.

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

[39] M. Di Marco, A. Garulli, S. Lacroix, and A. Vicino. A set theoretic approach to the simultaneous localization and map building problem. In *39th Conference on Decision and Control*, Sydney (Australia), December 2000. IEEE.

[40] Y. Matsumoto, M. Inaba, and H. Inoue. Memory-based navigation using omni-view sequence. In *1st International Conference on Field and Service Robotics*, Canberra (Australia), August 1997.

[41] Y. Matsumoto, K. Sakai, M. Inaba, and H. Inoue. View-based approach to robot navigation. In *International Conference on Intelligent Robotics and Systems*, Takamatsu (Japon), November 2000. Nara Institute of Science and Technology, IEEE.

[42] L. Matthies. Toward stochastic modeling of obstacle detectability in passive stereo range imagery. In *Conference on Computer Vision and Pattern Recognition*, pages 765–768, Champaign, IL (USA), 1992. IEEE.

[43] L. Matthies, A. Kelly, and T. Litwin. Obstacle detection for unmanned ground vehicles: A progress report. In *International Symposium of Robotics Research*, Munich (Germany), October 1995.

[44] M. Maurette. Cnes robotics program. In *5th ESA Workshop on Advanced Space Technologies for Robotics and Automation*, Noordwijk (The Netherlands), December 1998. CNES.

[45] D. P. Miller, F. J. Atkinson, B. Wilcox, and A. H. Mishkin. Autonomous navigation and control of a mars rover. In *11th IFAC Symposium on Automatic Control in Aerospace*, pages 127–130, Tsukuba (Japan), July 1989. JPL.

[46] S. Moorehead, R. Simmons, D. Apostolopoulos, and W. L. Whittaker. Autonomous navigation field results of a planetary analog robot in antarctica. In *5th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Noordwijk (The Netherlands), June 1999.

[47] R. R. Murphy. *Introduction to AI Robotics*, chapter Localization and Map Making, pages 375–434. The MIT Press, December 2000.

[48] D. Murray and J. Little. Using real-time stereo vision for mobile robot navigation. In *Workshop on Perception for Mobile Agents, Santa Barbara, Ca. (USA)*, 1998.

[49] C. Olson, L. Matthies, M. Schoppers, and M. Maimone. Robust stereo ego-motion for long distance navigation. In *Conference on Computer Vision and Pattern Recognition*, Hilton Head Island, SC (USA), June 2000. IEEE.

[50] C. F. Olson. Landmark selection for terrain matching. In *International Conference on Robotics and Automation*, pages 1447–1452, San Francisco, CA (USA), April 2000. IEEE.

[51] C. F. Olson and L. H. Matthies. Maximum likelihood rover localization by matching range maps. In *International Conference on Robotics and Automation*, pages 272–277, Leuven (Belgium), May 1998. IEEE.

[52] P. Putz. The esa research and development programme in space automation and robotics. In ASTRA [5].

[53] S. I. Roumeliotis and G. A. Bekey. 3d localization for a mars rover prototype. In *5th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, pages 441–448, Noordwijk (The Netherlands), June 1999.

[54] P. Schenker, T. Huntsberger, P. Pirjanian, and T. Baumgartner. Planetary rover developments supporting mars exploration, sample return and future human-robotics colonization. In *10th International Conference on Advanced Robotics*, pages 31–47, Budapest (Hungary), August 2001.

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

[56] R. Siegwart, T. Estier, Y. Crausaz, B. Merminod, M. Lauria, and R. Piguet. Innovative concept for wheeled locomotion in rough terrain. In *Sixth International Conference on Intelligent Autonomous Systems, Venice (Italy)*, July 2000.

[57] 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 *International Conference on Robotics and Automation*, pages 1194–1200, San Francisco, CA (USA), April 2000. IEEE.

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

[59] A. P. Tirumalai, B. Schunck, and R. C. Jain. Evidential reasoning for building environment maps. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(1):10–20, January 1995.

[60] R. Volpe. Navigation results from desert field tests of the rocky 7 mars rover prototype. *International Journal of Robotics Research*, 18(7):669–683, July 1999.