Integration of concurrent localization algorithms for a planetary rover

Simon Lacroix and Anthony Mallet
LAAS-CNRS
7 Av. du Colonel Roche - 31077 Toulouse Cedex 4 - France
Simon.Lacroix@laas.fr, Anthony.Mallet@laas.fr

Keywords: Rover localization, long range navigation.

Abstract

To achieve robust autonomous long range navigation, an estimate of the rover position is required at various levels in the whole system. No single localization algorithm can be robust enough to provide a good position estimate during long range navigation: a set of different algorithms have to be developed and integrated for that purpose. In this paper, we briefly present various localization algorithms developed in the lab, establish a typology of localization algorithms, and focus on the difficulties raised by their integration on board a rover.

1 Introduction

Future planetary rovers will have to autonomously navigate for several hundreds of meters or kilometers, exploring and gathering data in poorly known regions. It calls for the integration of a wide variety of processes, from low level actuator control, to higher level strategic decision making, via environment mapping and path planning. Among these various functionalities, self-localization is definitely one of the most important, as it is required at almost any level in the whole system, from fine trajectory control to mission supervision.

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. One must therefore consider the problems raised by the *integration* of these algorithms.

This paper discuses this issue. After a presentation of the importance of localization for long range autonomous rover navigation, section 3 presents a brief overview of the contributions related to localization in natural unstructured environments (*i.e.* planetary-like). Some developments integrated on board our Marsokhod robot Lama are briefly described, and the section concludes with a synthesis, that introduces a typology of localization algorithms. Section 4 presents the various problems raised by the integration of several localization algorithms, and how we began to tackle them with the robot Lama. A discussion that underlines some open issues concludes the paper.

Figure 1: *The Marsokhod Lama during a live demonstration at the "Cité de l'Espace" in Toulouse, Sept. 2000*

2 On the importance of localization

For long range autonomous navigation in planetary environments (and more generally for any mobile robot), the ability to localize, *i.e.* to estimate the rover position and the precision of 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). The knowledge of the rover position (and of the various positions reached during its traverse) is therefore required at the mission control level.

- 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. Note that in this latter case, some visual guidance techniques can help to drive the rover without the explicit knowledge of its position. However, up to now only a few planner generate such commands, and their domain of operation is essentially restricted to very local trajectories, in a manner similar to object catching tasks with an arm.

One can see that localization is required at various levels in the system. It is especially tightly linked with the mapping and exploration processes - the acronym SLAM that stands for "Simultaneous Localization And Map building" is widely understood in the roboticists community[1], and is a key component of the *robustness* of any navigation approach.

It is worth to note that among the various work related to the understanding of the intelligence of living beings, a lot are devoted to the *spatial reasoning*, in which the ability to localize play an important role. Note also that many technological developments in than mankind history have been devoted to the localization problem, from sextants and chronometers to Loran and GPS.

3 Localization algorithms

A vast amount of localization algorithms that use onboard data is available in the literature. We briefly review here the ones that can be relevant in the context of planetary exploration, classifying them from the data they use. At the end of the section, we propose a *typology* of these algorithms.

3.1 Odometry

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 terrains. Not only the trajectories are three dimensional (which calls for at least a tilt inclinometer), 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 all the contributions related to natural terrain odometry a rotation or orientation sensor is added (*e.g.* a gyrometer in [1, 2], or a sun tracker in [3]).

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

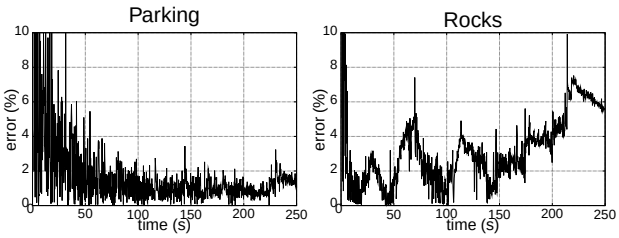

Figure 2: *Relative error on the traveled distance for two different kinds of terrain. Left: a flat parking lot, right: a rocky area. The error is computed using an RTK differential GPS (centimeter precision): the error values are quite big during the first tens of seconds because of the GPS noise.*

An interesting point is that there are means to *qualify* on-line the motion estimates. 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, can help to detect slippages (figure 3). Up to now, such an analysis remains however qualitative, and can to the best only detect gross errors: more work is required, in which the chassis configuration parameters should also be taken into account. Although odometry will hardly ever be useful over long ranges, it is required to control the rover motions: any progress in using such data is of course still welcomed.

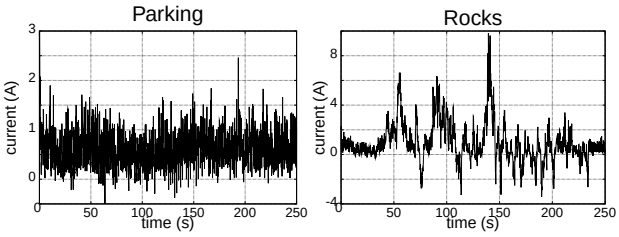

Figure 3: *Mean current consumed during the same trajectories as in figure 2. The application of simple thresholds over short periods of time could help to determine when odometry derives too much.*

3.2 Inertial navigation

Besides the use of a heading gyrometer, inertial measurements units are seldom considered in planetary rover prototypes: their extremely slow speed produce a very low signal/noise ratio. The development of low energy and extremely light strapdown units is however promis-

[1] Some author rather use CML for "Concurrent Mapping and Localization".

ing [4]. As for odometry, there is no hope that such sensors alone will help to localize the rover over long ranges. But the development of aided inertial units, that fuse their data with vision or odometry [5] (allowing the detection of cross-track velocities for instance, to which odometry is blind) will undoubtly popularize their use for rovers. Again, such sensors are essentially useful to servo the execution of motions.

3.3 Localization using range data

The localization problem being essentially a geometric one, it is not surprising that most contributions rely on the use of range data. One can distinguish among these contributions the ones that directly use raw data, the ones that relies on digital elevation maps, and the ones that detects and use landmarks.

3.3.1 Localization using raw range data

There are few contributions that directly use the range data to produce a position estimate, without any segmentation or 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 4), 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 [6] (a similar algorithm can be found in [7]).

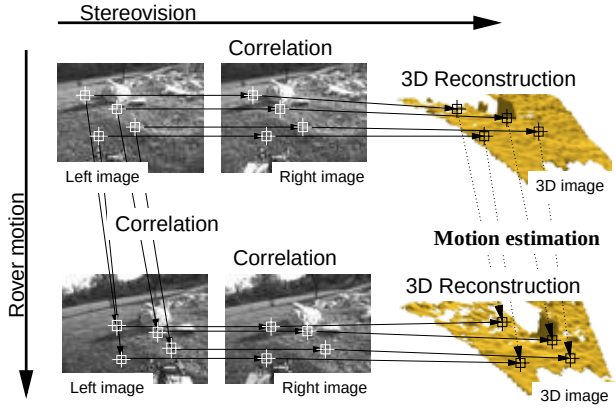

Figure 4: *Principle of the visual motion estimation technique based on stereovision and pixel tracking.*

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 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) and it gives translation estimates of about 2% of the distance (figure 5). Work related to this algorithm is still under way, with the goal of reaching a precision on translation estimates of about 1%. Also, the algorithm provides a possibility to get and estimation of the precision of the computed position.

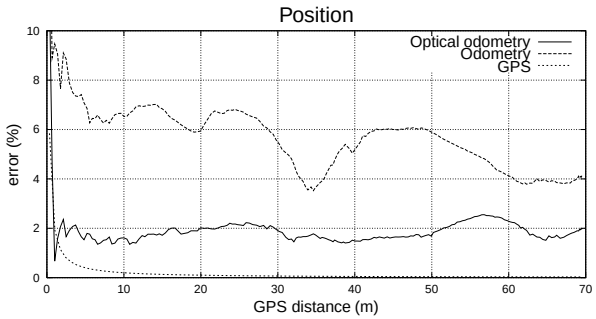

Figure 5: *Comparison of the error on the translations estimated by stereovision and pixel tracking (denoted "optical odometry") and odometry, during a $70m$ long run that includes sharp turns, hill climbing and rocky area traverses. The GPS precision curve is superposed, showing that it quickly becomes negligible.*

3.3.2 Landmarks-based localization

An efficient way to localize a rover is to rely on particular elements present in the environment, referred to as *landmarks*. Several authors presented range data segmentation procedures to extract salient objects [8, 9]. Such techniques are however efficient only in simple cases, *i.e.* in scenes were sparse rocks lie on a very flat terrain, but rather fragile on rough or highly cluttered terrains for instance. The difficulties come here from both the nature of the environment and the uncertainties on the 3D data: the most promising approaches are the ones that use both video and range data to extract landmarks. In section 3.3.3, we briefly presents how we can extract landmarks from digital elevation maps.

3.3.3 Localization using terrain maps

One of the most popular way to map natural environments is the use of digital elevation maps (DEMs), that represent the terrain as a set of elevations computed on a regular horizontal Cartesian grid. Although there has

been several contributions to the problem of building digital elevation maps (see *e.g.* [10]), 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 presents the problem for various cases of sensor configurations: ideally, a digital elevation map should be extracted from a 3D occupancy grid [11].

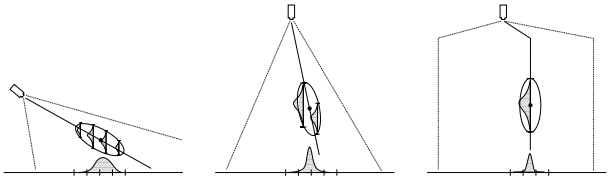

Figure 6: *Sensor errors with respect to an horizontal grid. In the leftmost case, which is the one encountered with range images acquired from a rover, the precision of the data, which is essentially in the range estimate, should be transformed as an occupancy probability in the DEM cells. In the rightmost case, which correspond to data acquired from a high altitude device, 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.*

A 3D occupancy grid is however not suited for large terrain models: we developped a heuristical algorithm to build an elevation map, for each cell of which 3 values are associated: an elevation, a value that represents the *precision* of the elevation, and a value that represents the *confidence* of this elevation.

Various contributions that use a digital elevation map to localize the rover can be found in the literature: one distinguishes the *iconic-based* techniques [12, 13, 14], from the *feature-based* techniques.

The iconic-based techniques give globally satisfactory results, but the position estimates they provide are hardly qualified in term of precision. With the robot Lama, instead of matching a local elevation map with a global one, we directly find the 3D transformation that minimizes a given distance between the last acquired range image and the current elevation map, using a simplex algorithm. Current results are encouraging, but not better that odometry up to now. Note that the way the precision of the range data and of the map is represented should play here a crucial role, especially to have an estimate of the error on the computed position.

Feature-based techniques rely on the extraction of landmarks in the digital elevation map [15]. We developed an algorithm to extract *local peaks*, that relies on the computation of similarity scores between a digital elevation map area and a pre-defined 3D peak-like pattern (a paraboloid for instance), at various scales. First results are encouraging (figure 7), and the detected landmark could be used to feed a position estimation technique.

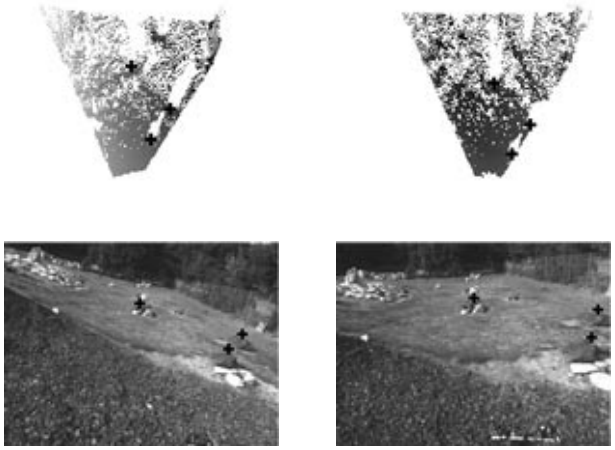

Figure 7: *Landmarks (black + signs) detected on the locally built digital elevation maps (top), and re-projected in the camera frame (bottom). Landmarks are here from 3 to 10 meters away, and 3 meters separate the two image acquisitions.*

Map structure. One of the problem of digital maps is their rigid structure: when the rovers re-localize itself after a large drift, there is no way to update the whole map, as it is possible with landmark maps for instance. For that purpose, we developed a specific map structure that supports big updates of the rover position. It consists in building a set of local maps, each one attached to a single terrain perception: only the local maps are memorized, and fusion is done when an external functionality asks it (figure 8).

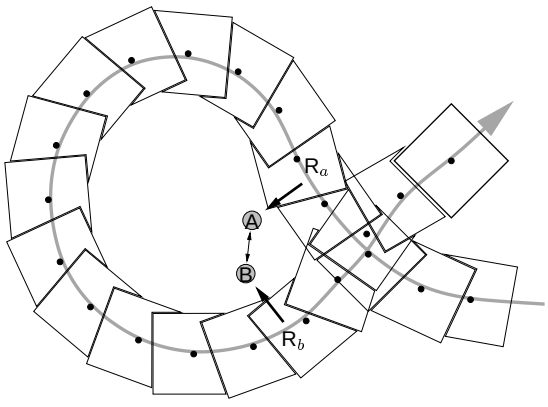

Figure 8: *Principle of the map structure: the black dots represent the center of each sub-map, the last one (thick square on the right) being not yet sub-divided. Note that where some maps are superposed, no fusion has been performed: if any localization algorithm states that point R_a and R_b coincide, the map can be updated without any damage.*

3.4 Localization using vision

The problem referred to as "motion and structure from motion" in the vision community [16], that estimates both the camera motions and the environment geometry on the basis of a sequence of monocular images, could be used in robotics. It remains however computationally very expensive, and to our knowledge none of these technique has ever been used to localize a rover.

More interesting are the contributions that extract and track landmarks from monocular video images. Some rely on features present on the horizon line (or the whole horizon profile) [17, 18], and match them with features derived from an initial elevation map. Other use closer features [19], and localize the rover using only landmarks bearings.

Finally, the use of image indexation techniques with panoramic images is a very promising way to tackle the place recognition problem [20], which would ensure the possibility to re-localize the rover after large loops for instance.

3.5 Topological localization

Topological (or qualitative) localization consists in finding the robot position with respect to a topological representation derived from sensory data. Such algorithms do not rely on metric information, but rather on place recognition, using object identification and qualitative neighboring relations [21]. Such approaches have essentially been applied to indoor navigation, where the representation is a graph of possible robot paths [22]. In natural environments, where such graphs can hardly be built, image registration techniques could be applied. Such algorithms can be useful to re-estimate the rover position after large loops, but have not yet been applied on-board real rovers.

3.6 Synthesis: a classification of localization methods

One can see from this brief overview that there is a big variety of approaches to localize a rover. We propose here a typology of these approaches, that helps to have a better understanding of the problem, and to see the implications of localization 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 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 numerical terrain models derived from orbital imagery).

We presented these algorithm classes according to the kind of data they use (with or without memory, with or without initial data), but there are actually four more criteria that lead us to establish such a classification: the evolution of the error on the position estimate, 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 9).

Class / Criteria	Motion estimation	Pose 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 brutes	Landmark or environment models	Landmark or environment models
Control	No	Data acquisition control, influences motion generation	Drives the motion generation

Figure 9: *Behavior of the four criteria in our typology of localization algorithms.*

A complete navigation system should be endowed with at least one instance of 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.

4 Integration of several localization algorithms

Some particular integration problems are related to the coexistence of several localization algorithms running in parallel on board the robot. To tackle this in a generic and reconfigurable way, we developed a particular module

named *PoM* (position manager), that receives all the position estimates produced by the localization algorithms (each of them being integrated within a specific *module*) as inputs, and produces a single consistent position estimate as an output. *PoM* addresses the following issues:

• *Sensor geometrical distribution.* The sensors being distributed all over the robot, one must know their relative positions, which can dynamically change. Indeed, we want all the modules to be generic, *i.e.* to handle data without having to consider the position from which it had been acquired. This is particularly true for video images, since Lama is equipped with two orientable stereo benches: we may want to switch benches whenever required, with the minimal effort. For this purpose, we developed a framework named "*InSitu*" (internal situation, section 4.1) which is part of *PoM*.

• *Localization modules asynchronism.* The localization algorithms have individual time properties: some produce a position estimate at a regular high frequency, while others require some non-constant computation time. To be able to provide a consistent position estimate at any time, the "time-management" part of *PoM* has been developed (section 4.2).

• *Fusion of the various position estimates.* The fusion of various position estimates is a key issue in robot localization. This is done within the "fusion" part of *PoM* (briefly discussed in section 4.3).

4.1 Internal situation

Some problems arise when the sensors of a robot are geometrically distributed. For instance, the vision-based localization module has to know the orientation of the cameras for each image it processes, whereas the digital elevation map module needs the relative and/or absolute position of the *3-D* images it is using in a predefined coordinate frame.

Distributing the geometrical informations in the modules is not satisfying: some informations are hard-coded and duplicated within the modules, and it complicates the porting of a module to another robot or another sensor. For that purpose, we use *InSitu*, a centralized geometrical description of a robot. It reads a configuration file upon startup, and it 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 with the necessary informations.

The configuration file is the textual description of a geometrical graph (figure 10). 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. This graph is the only robot-specific part of *PoM*.

The links between frames are either *static* (rigid) or

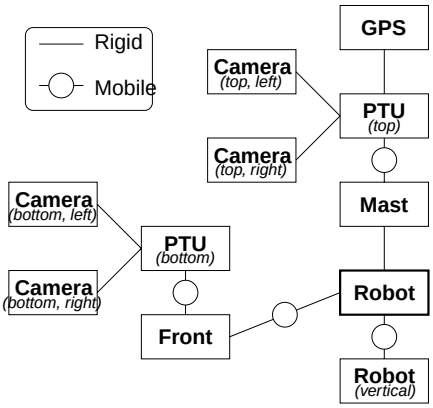

Figure 10: *The geometrical graph used on the robot Lama. Rectangular boxes are frames configurations that are exported in the system. The thick box (*Robot *in the figure) is the* main *frame. Solid lines are links (either rigid or mobile) that are declared in the configuration file. Each mobile link has an associated function that gets the link parameters from the underlying hardware, computes the current transformation matrix and sends it back to* PoM.

dynamic (mobile). A static link cannot change during execution and is usually related to some mechanical part of the robot. On the other hand, dynamic links, that depend on the chassis configuration or on a PTU configuration for instance, are updated continuously at a predefined frequency. Updates are made possible with the definition of link specific functions that get links parameters from the underlying hardware system. These functions are grouped together in a library associated with the configuration file. Last, the configuration file defines a *common* frame (also called *main* frame). To facilitate and homogenize inter-module data transfers, the various modules data are expressed in this frame.

To ease data manipulation and transfer among the modules, *InSitu* continuously exports all the frames configurations found in the configuration file with the structure shown in figure 11.

The header of exported data contains three fields: the first field is the current *PoM*-date expressed in ticks since boot-time. The next two fields are positions: the *Main to Origin* transformation is the current *absolute* position of the *main* frame relative to an absolute frame. The *Origin* frame is usually the position of the robot at boot-time but it can be specified anywhere. The *Main to Base* transformation is a relative position which cannot be used as such. The sole operation permitted with this frame is the composition with another *Main to Base* transformation (see section 4.2 for a detailed description). *Base* is a virtual frame that is maintained and computed by *PoM*.

After the header, there is a variable number of transformations which are the current frames configuration. They are named according to the scheme "*Frame to Main*" and are the transformation matrices that map *3-D* points in the "*Frame*" frame to *3-D* points in the "*Main*" frame.

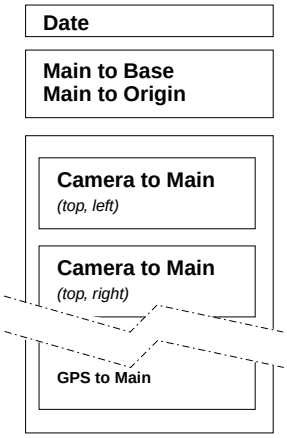

Figure 11: *Structure exported by* InSitu.

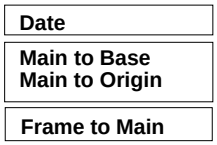

Figure 12: *Structure for data tagging.*

Every data acquisition module reads the *InSitu* frame which it relates to, and associate to its data a "tag" structure (figure 12). Thanks to this tagging, data acquisition modules only have to know the name of the frame they are connected to, which can be specified dynamically. Once tagging is done, clients using such data do not have to care from where the data comes, since all the necessary geometrical and time information is contained in the data itself. The tag is propagated along with the data between modules, thus making inter-module data communication very flexible.

4.2 Position Management

In addition to internal frame configurations, *PoM* collects the various position estimators present in the system. It is the place where positional information is centralized and made available for any module that require it.

Positions computed by the various position estimators are always produced with some delay, that depends on the computation time required to produce a particular position. Thus the robot has always to deal with outdated positions. One of the role of *PoM* is to maintain the time consistency between the various position estimates.

Internally, there is one time chart for each position estimator handled by *PoM*, plus one particular chart for the *fused* position. The *fused* position is the result of the fusion of every position estimator (see section 4.2). All charts have the same length and hold every produced positions since the beginning of the chart, up to the current date. Length is variable but is computed so that we always have at least two positions of every motion estimator (figure 13).

PoM periodically polls every position estimator and look if a position have been updated. When a new position is found, it is stored in the chart of the corresponding motion estimator, at date its corresponding date ("updates" arrows in figure 13). If the chart is not long enough to store a too old date, it is enlarged as needed. Once every motion estimator has been polled, every fused position from the oldest update to the current time is marked for re-estimation.

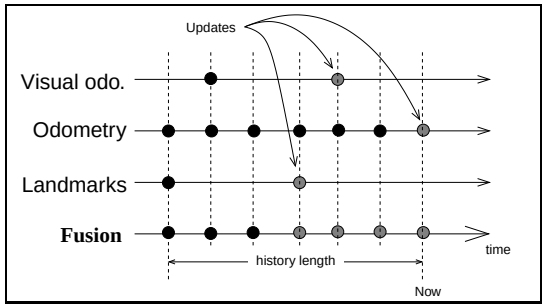

Figure 13: *Time management and internal representation of motion estimators in* PoM. *Position estimators shown here are only examples. Black dots show previously stored positions. Dashed dots show new upcoming positions.*

Once the current best position is computed (see section 4.3, *PoM* stores it in the *fusion* time chart (figure 13). The important thing to note is that we cannot simply store the new position as such because it might be very different from the previous position. Since some module may use the robot position continuously (such as servoing on a computed trajectory for instance), virtual jumps are not permitted. This is why we defined a *reference position estimator*, and a virtual frame named *Base*. The fused position computed by *PoM* is the *Robot to Origin* position. This one is exported as such and used by modules that require *absolute* position. The second exported position, *Robot to Base* is the position of the *reference estimator* and can only be used locally, *i.e.* to compute a *delta* position. *PoM* computes a *Base to Origin* transformation for every position estimator found in the system. This transformation is an indicator of the drift of each position estimator.

4.3 Fusion of the estimates

Sections 4.1 and 4.2 presented 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, 1.0 being the best) that

is hard-coded for each position estimator.

5 Discussion

We sketched the various problems raised by the integration of localization algorithms, and proposed a structural framework that allows the consideration of various concurrent unsynchronized algorithms. The most important issue it now to derive an *on-line* error model for each algorithm, which calls for both empirical tests and algorithm analysis. Besides this, the development of localization algorithms with respect to an initial model of the environment is required to robustly tackle long range navigation: digital elevation maps appear to be the best environment model for that purpose.

References

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

[2] S. Roumeliotis and G. Bekey. 3d localization for a Mars Rover Prototype. In *5th International Symposium on Artificial Intelligence, Robotics and Automation in Space, Noordwijk (The Netherlands)*, pages 441–448, June 1999.

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

[4] B. Barshan and H. Durrant-Whyte. Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation*, 11(3):328–342, 1995.

[5] Y. Fuke and E. Krotkov. Dead-reckoning for a lunar rover on uneven terrain. In *IEEE International Conference On Robotics and Automation, Minneapolis, Minesota (USA)*, pages 411–416, April 1996.

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

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

[8] M. Hebert. 3d landmark recognition from range images. In *IEEE International Conference on Computer Vision and Pattern Recognition, Champaign, Illinois (USA)*, pages 360–365, 1992.

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

[10] 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, Feb. 1992.

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

[12] M. Asada. Building a 3d world model for a mobile robot from sensory data. In *IEEE International Conference on Robotics and Automation, Philadelphia (USA)*, pages 918–923, 1988.

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

[14] C. Olson. Mobile robot self-localization by iconic matching of range maps. In *8th International Conference on Advanced Robotics, Monterey, Ca (USA)*, pages 447–452, July 1997.

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

[16] T.S. Huang and A. N. Netravalli. Motion and structure from feature correspondences: A review. *Proceedings of the IEEE*, 82(2):252–258, Feb. 1994.

[17] S. Suh, J. Kang, W. Jee, M. Jung, and K. Kim. Estimating alv position in mountainous area. In *IEER/RSJ International Conference on Intelligent Robots and Systems, Yokohama (Japan)*, pages 2178–2185, July 1993.

[18] F. Cozman and E. Krotkov. Automatic mountain detection and pose estimation for teleoperation of lunar rovers. In *International Symposium on Experimental Robotics, Barcelona (Spain)*, pages 164–172, June 1997.

[19] M. Deans and M. Herbert. Experimental comparison of techniques for localization and mapping using a bearings only sensor. In S. Singh, editor, *Experimental Robotics VII*, Lecture Notes in Computer Science. Springer-Verlag, 2001.

[20] Y. Matsumoto, K. Sakai, M. Inaba, and H. Inoue. View-based approach to robot navigation. In *IEEE/RSJ International Conference on Intelligent Robots and systems*. Nara Institute of Science and Technology, Nov. 2000.

[21] B.J. Kuipers and Y.T. Byun. A robust, qualitative method for robot spatial learning. In *7th National Conference on Artificial Intelligence, Saint Paul (USA)*, pages 774–779. Uni. of Texas at Austin, 1988.

[22] K. Nagatani, H. Choset, and S. Thrun. Towards exact localization without explicit localization with the general voronoi graph. In *IEEE International Conference on Robotics and Automation, Leuwen (Belgium)*, pages 342–348. CMU, May 1998.