Managing environment models in multi-robot teams

Pierrick Koch, Simon Lacroix

Managing environment models in multi-robot teams

Pierrick Koch and Simon Lacroix
CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
Univ de Toulouse, INSA, F-31400 Toulouse, France
pierrick.koch@laas.fr simon.lacroix@laas.fr

***Abstract*— Multi-robot cooperation in a dynamic environment implies that each involved robot is able to create, update and exchange environment models – which are essential to the autonomy of mobile robots. The ability to exchange data, in other words to communicate, rely on the fact that robots share a common language and common frames in space and time.**

In this paper we present an open-source architecture [1] dedicated to build a collection of environment models of a dynamic environment with multiple robots. This work is lead with air-ground cooperation scenarios in mind, such as the ones considered within the Action project [2]. Managing models during the mission means defining methods and protocols to transmit and merge models built from different sources and on different robots, while respecting time and bandwidth constraints and being robust to unavoidable inconsistencies between the models. Our first work builds upon GDAL, a well known standard in geographic information systems (GIS). The architecture is currently able to build models with different robots in simulation. Current work aims at detecting and solving inconsistencies between the various models, and at porting the architecture on-board real robots.

I. Introduction

The basic scheme to define autonomous operations of a robot is to predict the consequences of their actions, so as to select the ones that lead to the achievement of the given task ("decision"). Predicting actions consequences requires the application of the action models onto models of the environment, so the availability of environment models is therefore a key component of robots autonomy. In a partially unknown and dynamic world, such models must be derived from sensor data acquired by the robots, that are aggregated into the models to be made available to the decision processes.

A. A variety of environment models

In general numerous actions can be performed by robots, such as grasping objects, moving form one position to the other, gathering knowledge on a given area... We consider here the case of robots operating in large scale outdoor environments, to achieve exploration, patrolling or monitoring tasks. In such cases, the main actions come down to motions, observations, and communications – *e.g.* with the remote operators control station or between robots.

The variety of actions to plan and execute calls for a variety of environments models: it is indeed illusory to define a model structure that encodes all the information required to assess the outcome of all the considered actions. Instead, we favor the development of *layered representations*, each representation being dedicated to plan a given action (figure 1):

- Models that represent the geometry of the terrain are required to plan motions. A digital elevation model (DEM) is required for local mobility, achieved by a short term path planner that must be fast and real-time; a more abstract description of the world is needed for higher level of decision, as the mission planner, demanding lower resolution and less time constraints (traversability model);
- 3D models are required to evaluate the communication and visibility of an area from a given position
- *Utility* models encode the interest of observing areas, *e.g.* the probability that it contains an event to monitor (orthoimage);
- Dedicated models are required for localization, exhibiting characteristic elements in the environment (landmarks)
- ...

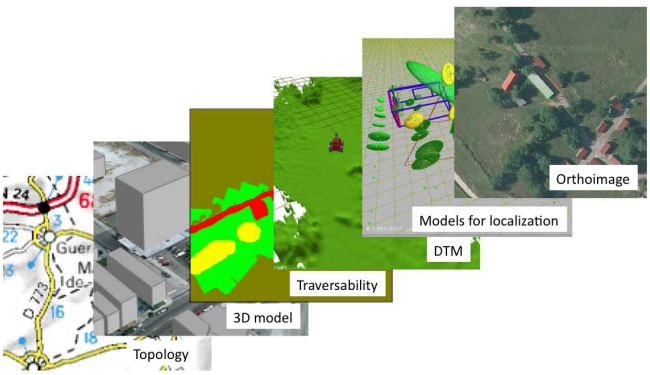

Fig. 1. Various semantic layers of environment representations. Each layer is dedicated to a given type of information, dedicated to a given decisional of functional process.

Note that one can distinguish three main types of data structure to encode information on the environment in the models:

- *raster* structures represent continuous information (surfaces, volumes) on a discretized representation, either 2D or 3D [1];
- *vector* structures represent a set of discrete information (landmarks, particular elements), that are localized with

[1] https://github.com/pierriko/atlaas#atlaas
[2] http://action.onera.fr

the associated uncertainties;
- *topological* structures represent spatial relations between areas or elements, representing accessibility (navigation graph), inter visibility...

B. Motivations

The basic procedure to build any environment model on the basis of the data provided by the environment sensors consists in acquiring the data and then to merge it in the data structure that defines the environment model. This calls for an *essential* requirement: the robot must be well localized, so that the environment model is *spatially consistent*. A lot of work has been dedicated in robotics to the localization problem, yielding in particular feature-based Simultaneous Localization And Mapping approaches (SLAM), in which landmarks are detected and memorized (which defines a specific layer of environment model – of vector type). Nevertheless, one can never guarantee that a robot is precisely localized in the long term, an naively integrating data in raster type data structures as the robot is not properly localized lead to inconsistent models, whose consistency can't be recovered afterwards (figure 2).

Fig. 2. Inaccurate localization leads to model corruption: here the DEM integrates data acquired while the robot was not always properly localized: building walls are represented twice at different positions.

Besides, the environment may contain dynamic elements, that are often hardly detected in the acquired data, and whose introduction in the environment models may lead to *temporally inconsistent* models, *e.g.* by memorizing in the same data structure the presence of elements that had different positions at different time instants (figure 3).

These two issues become even more important when considering teams of robots operating under communication constraints. To cooperate, the robots must have the ability to share tasks to achieve goals and missions. For this purpose, they need a common language, they need to share the information they have acquired on the environment, and they need to know where they are with respect to each other. The inter-robot communications being not permanent and subject to environmental perturbations, no central approach can be efficient: each robot must build and manage its own

Fig. 3. A car "ghost" on a parking lot map: points corresponding to the car have been perceived while it was moving, and have been merged in different map cells (the "trace" of the car can be seen starting in the center of the model, an going up-left). Here the resulting map is not temporally consistent, as it encodes time varying information gathered at different time instants.

environment representations, and data exchanges between the robots must be optimized.

Managing environment models in a multi-robot context also brings up additional issues with localization (the robots being not precisely localized one with respect to the others) and the presence of dynamic elements, the robots sharing information that may have been acquired at very different time instants.

C. Outline

We propose in this article a mean to manage raster-like environment models so that localization errors and dynamic elements do not corrupt them: each robot raster models are maintained as a series of "tiles", that is to say local environment models, whose spatial and time consistency can be guaranteed. When the robots move and/or exchange information, the spatial and time inconsistencies are actively detected and managed.

Section II presents digital elevation models, a basic data structure on which we have worked so far. Section III depicts our approach to manage such models over long ranges of time and distances. Finally section IV recaps our on-going and future work.

II. Digital Elevation Models

A Digital Elevation Model is a raster representation that encodes the terrain geometry as a function $z = f(x, y)$, estimated over a regular Cartesian grid: it is a 2.5D height-map where each cell value represents an elevation. It is a very common data structure to represent the terrain geometry, which is for instance used to assess the terrain traversability,

either to generate local motions of the robot to reach a goal avoiding obstacles, or to plan long term itineraries. Such a model is incrementally built by merging point clouds from the robot depth sensors (*e.g.* stereovision or LIDAR) into the raster. This process must run in real-time to enable the robot to continuously plan its trajectory [2]. Merging is basically done by averaging height of points perceived at each cell in the map.

However, dynamic elements of the environment create "ghosts" that should be removed from the model (see Figure 3), and the presence of vertical elements is poorly encoded by this merging process.

To solve this problem, we check whether a cell encodes a vertical area, an area that can be modelled as a surface, or an area that contains a dynamic object. This is made by computing a local (in space and time) DEM using the latest point could acquired, and by analyzing the variance σz on the elevation z of each cell. Depending on whether this variance exceeds a fixed threshold T or not and on the state of the corresponding cell in the current model, the state of the model cells is updated according to the merge logic shown Figure 4.

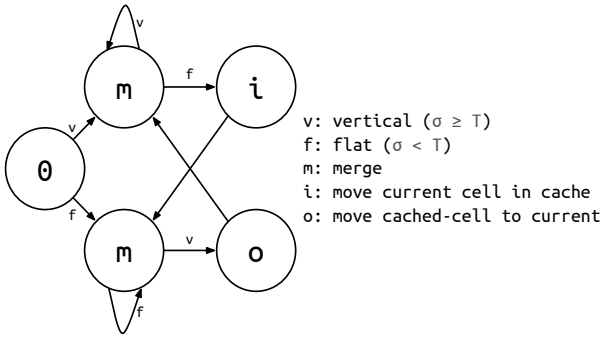

Fig. 4. Merge logic of the local instantaneous cell of the DEM with the current DEM. A cell is assessed "vertical" if $\sigma_z > T$, "flat" otherwise.

This graph does not show the specific case where both current and new cells are considered flat, but their height differs: cells can not be merged in this case, and are hence assessed as "dynamic", the current cell value is kept in a cache, and replaced by the new one.

Following a cell state allows to assess whether it contains a dynamic element or not. Such information is also relevant for map registration, in order to filter landmarks, keeping the most stable areas for model matching. This is an important point from the lifelong autonomy perspective: the model must be re-usable and adaptable, even though drastic changes occur in the environment (*e.g.* on a parking lot).

III. ATLAAS

We propose to use tiles to keep track of past modelled areas, so as to cope with robot localisation and environment dynamics issues, and also to ease the exchange of information between robots. This is achieved by subdividing space into a set of smaller maps, allowing to cache data, dump and load them as the robot moves to different areas.

A. Pile of tiles

A tile is composed by different layers, for each cell, we are currently storing the lowest and highest elevations perceived, the elevation mean and variance, the number of points it contains, and the timestamp of the last update.

The current implementation merges incoming point clouds in a map of 9 tiles ($3 * 3$). When the robot moves out of the central tile, we dump the ones out of the map and load existing if any, using a simple hysteresis threshold. For local path planning purposes, we build maps of 0.1 meter resolution, of size $40 \times 40m^3$: this size is selected so as to make sure each tile is spatially consistent, making the hypothesis that the robot localization drift is not exceeding the cell resolution over the tile size (we are using inertial-visual SLAM [3]).

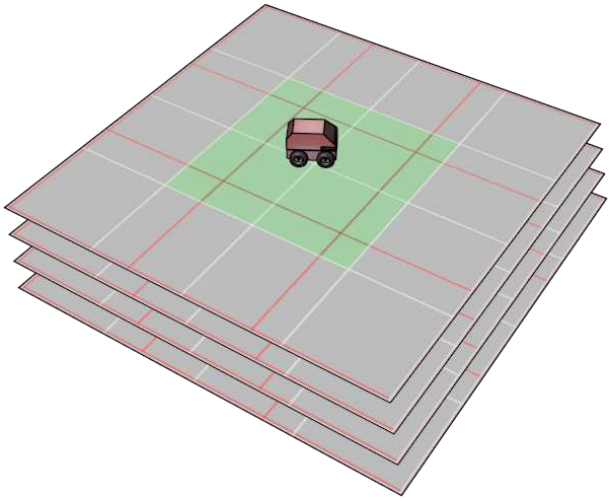

Fig. 5. Current map of $3 * 3$ tiles with their layers. You can see the tiles delimitation in red, and in green the central "hysteresis" robot area (best view with colors).

Each tile being spatially and temporally consistent, and can be treated as an independent dataset containing its own relative frame and the transformation to the global frame. This hierarchy [4] gives the possibility to reshape the overall model as the robot position is updated – *e.g.* after a loop closure detection and correction by the SLAM algorithms. The overall model is generated by composed tiles, hence can quickly be re-arranged.

Space subdivision is also a key for sharing data between robots, sending chunk of information on demand, and registering them in another robot frame using remote synchronization methods.

B. Position referencing

In order to share models in a multi-robot scenario, it is first necessary to rely on standard for sharing frames in space and time. We are using the Universal Transverse Mercator (UTM) coordinate system to refer the position of each of the tiles, to be consistent with GPS position estimations – and with existing maps.

[3]For 9 tiles of $40 * 40m$, the memory footprint is 33 MB

A tile is one file, a multi-layer GeoTiff/Float32 saved using the Geospatial Data Abstraction Library (GDAL). We built a generic library wrapping GDAL I/O using modern C++11 STL containers for in-memory storage [4].

We keep an history of past tiles for the same area and use registration based on "stable" cells by computing the probability of the dynamic of each cell. It is when tiles overlap that we will handle the issues, by properly "merging" tiles, after having registered them (either by directly correlating them [5], or matching landmark maps associated to the tiles).

Furthermore, keeping only one frame of reference is not sufficient. It is desirable to have a model that is still fixed as the robot moves forward and when loop closure appears. Hence the track of local reference in the global model should be kept and transformed to local chunk of data to prevent corruption. Doing so while sharing models means that tiles with reference to the global frame and information about its potential error (covariance) must be memorized.

The architecture as it is today provides the foundation of tiles referencing, containing valuable information on all cells... We could test in simulation using the MORSE simulator [6].

IV. Discussion

We have sketched the foundation of a new architecture for environment raster models management. It is still in early stage development, being ported on multi-robots at the time of writing. It has proved to work in simulation, most of the work is now focusing on better handling of the dynamics, and better handling of the localizations update from loop-closures.

So far we have restricted its use to 2.5D raster models, which is a good compromise of level of detail versus computational time for our needs. But other data structure could be exploited, such a 3D hierarchized raster structures. Interesting framework have been proposed by the community, as in "octomap" [1] where using ray-tracing allow to clear cells between the sensors and every points perceived. Such method allow to take advantage of the sensor's characteristics. However such algorithm require a considerable amount of computation and time which are not available in many autonomous robotic system.

Further, one could also define tiles whose spatial consistency can be updated over time, as in the HYMM [7] framework. This would allow to propagate SLAM corrections into the raster models.

We are also considering dividing sensor data in two categories, the close points are considered valid and use directly for local path planner in the real-time loop, and points above a certain threshold needs some filtering and re-localization using matching algorithm [8], such points are not critic in the locomotion process thus can be treated in a parallel thread.

Upcoming challenges are a fine time management, allowing to come back in a past state. Bandwidth constraint, how to efficiently exchange data. And guarantee the global consistency, by registering externally loaded models.

[4] http://trac.laas.fr/git/gdalwrap

References

[1] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.

[2] A. Kleiner and C. Dornhege, "Real-time localization and elevation mapping within urban search and rescue scenarios: Field reports," *J. Field Robot.*, vol. 24, no. 8-9, pp. 723–745, 2007.

[3] C. Roussillon, A. Gonzalez, J. Solà, J.-M. Codol, N. Mansard, S. Lacroix, and M. Devy, "Rt-slam: A generic and real-time visual slam implementation," in *ICVS*, pp. 31–40, 2011.

[4] P. Beeson, J. Modayil, and B. Kuipers, "Factoring the mapping problem: Mobile robot map-building in the hybrid spatial semantic hierarchy," *The International Journal of Robotics Research April 2010 vol. 29 no. 4 428-459*, pp. 428–459, 2010.

[5] B.-V. Pham, A. Maligo, and S. Lacroix, "Absolute map-based localization for a planetary rover," in *12th ESA Workshop on Advanced Space Technologies for Robotics and Automation, Noordwijk (The Netherlands)*, 2013.

[6] G. Echeverria, S. Lemaignan, A. Degroote, S. Lacroix, M. Karg, P. Koch, C. Lesire, and S. Stinckwich, "Simulating complex robotic scenarios with morse," in *SIMPAR*, pp. 197–208, 2012.

[7] J. I. Nieto, J. E. Guivant, and E. M. Nebot, "The hybrid metric maps (hymms): A novel map representation for denseslam," in *ICRA*, pp. 391–396, 2004.

[8] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing icp variants on real-world data sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, 2013.