Managing environment models in multi-robot teams

Pierrick Koch, Simon Lacroix

Managing environment models in multi-robot teams

Pierrick Koch[1,2] and Simon Lacroix[1,3]

***Abstract*— Environment models are the primary matter to autonomous decisions for mobile robots, and also to cooperation within teams of robots that operate in the same environment. The decisions to take within a robot or a robot team relate to motions, perceptions and communications: various types of environment models are therefore required to evaluate and plan these actions. While the literature abounds with approaches to environment modeling using data perceived by the robots, very few work tackle the problem of *managing* such models within a team of robots. Managing environment models implies first defining the proper data structures and associated mechanisms that allow both their efficient update and use by the decisional processes that require them, and second ensuring the models consistency as the robots evolve. This article presents the definition of a framework dedicated to the managing of environment models within a robot team. It establishes the principles that govern the framework design, and illustrates them throughout some examples.**

I. INTRODUCTION

Environment models are at the heart for the autonomy of mobile robots, as they constitute the main information on which planning processes rely to select and configure the tasks to fulfill a given mission. Their *consistency*, in the sense of their fidelity to the reality of the information they represent, is therefore of utmost importance, as any discrepancy or error will eventually lead to wrong decisions. When it comes to teams of robots operating in the same environment, sharing environment models is a pre-requisite to cooperation: whatever the mission to achieve (*e.g.* exploration, surveillance), the robots must indeed have a common understanding of the situation at hand to cooperate.

The robotics literature naturally abounds with contributions on the *building* of various kinds of environment models from the data gathered by robots. Similarly, a vast amount of work has been devoted to robot localization, which is in particular required to ensure the *spatial consistency* of the built models, either in mono or multi-robot contexts. Yet much fewer contributions can be found on the *managing* of environment models, be it within a single robot or within a team of robots. Managing the environment models consists mainly in *(i)* structuring them so as to properly serve the processes that exploit them, *(ii)* making sure that the most precise information is extracted from the available data, in spite of the dynamics of the environment, the positioning errors, and the asynchronous information transfer between robots.

[1]CNRS, LAAS, 7 av du colonel Roche, F-31400 Toulouse, France; {pierrick.koch, simon.lacroix} at laas.fr
[2]Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France
[3]Univ de Toulouse, LAAS, F-31400 Toulouse, France

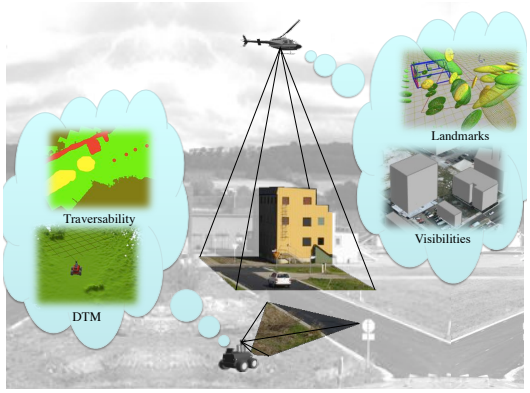


Fig. 1: An AAV and an AGV jointly operate in the same environment. For this purpose, they exploit a series of environment-related information to plan, coordinate and control their activities (*e.g.* environment traversability, visibility, communication and localization constraints, ...). How to structure these various environment models, so as to ensure their consistency within the team? How to enable their sharing between the robots?

This paper presents AtLaas[1], an open-source *framework* dedicated to the management of environment models among mobile robots. The primary considered context is large scale outdoor environments, of which initial models may or may not be known, and in which a team of robots of various kinds jointly achieve long duration missions that mainly relate to environment perception, *e.g.* in search and rescue or environment monitoring missions. The environment is supposed to be non-networked, in the sense that the robots can not rely on an ubiquitous, large bandwidth communication infrastructure that would allow them to benefit from remote powerful storage and computation servers. Yet the robots are of course endowed with communication abilities, that are however constrained by the environment in range and bandwidth: communication activities have to be actively controlled by the team, and the framework must also offer means for this purpose.

Defining such a framework requires to face the usual challenges with which environment modeling and robot localization algorithms have to cope: incomplete and noisy data, variety of information sources and quality (including initial environment models), environment dynamics... These points are crucial in the design of the modeling and localization algorithms, and have driven nearly three decades of research

[1]https://github.com/pierriko/atlaas

in robotics, that mostly relate to uncertain data fusion. But the maintenance of spatial and temporal consistency of the models within the team calls for additional concerns, such as the memorization of the gathered data, *e.g.* so as to rebuild a model corrupted by wrongly positioned data when past localization information are updated. The heterogeneity of the robots also impacts the management of the models: for instance some landmarks can be exploited to localize a given kind of robots and not others that have no mean to perceive them, or the traversability information are robot-dependent. Finally, the communication constraints impose a distributed solution to manage the various models.

Related work: Environment models management is mostly considered to tackle the dynamics changes, *e.g.* [1], [2]. When it comes to multi-robot systems, most contributions on the building of environment models come to localization, either of the robots themselves or of targets detected in the environment, for which the distribution of sources has fostered theoretical analyses that have led to practical implementations [3]. This is not surprising, as on the one hand localization is at the core of the building of environment models, and on the other hand multiple robots yields new means to ensure localization – *e.g.* by defining additional loop-closures in SLAM approaches [4]. Besides localization, environment related information being at the core of search or exploration missions, numerous contributions deal with the fusion of such information in a multi-robot context (*e.g.* [5], [6]).

Much less contributions can be found on the *managing* of environment models. By proposing a "stream-based knowledge processing middleware" DyKnow[2] rather focuses on symbolic information propagation within a system, handling in particular their time properties. When it comes to environment models, some contributions exploit "torrent-like" tools developed for web-based distributed map management [7], [8], which comes to manage a distributed database over a cloud infrastructure.

Approach and outline: The proposed framework implements two basic principles: *purposiveness* of the environment models, that are defined and built according to the processes that exploit them, and *economy of means*, be they related to computations, storage and communications. Section II analyses the relations between environment models and the various processes that exploit them (mainly decision and planning, but also localization), which naturally yields the development of layered *purposed* environment models. Section III introduces *AtLaas*, a framework which structures and manages the various environment models and gathered data in a dedicated database, and which provides requests that expose the environment models to client processes. Two illustrations of the mechanics encoded within *AtLaas* are provided in section IV, and a discussion concludes the paper.

[2]http://www.ida.liu.se/divisions/aiics/projects/dyknow.en.shtml

II. Environment models and decisional processes

A. A series of dedicated models

Decisional processes rely on both environment and action models, actions being environment robot motions, observations (perception), and communications with other robots or the remote control/monitoring station.

1) Robot motions: Robot motions are most often planned at two levels [9]: at the local *trajectory* level, optimizing safety (obstacle avoidance) and speed, and satisfying kinematic and dynamic constraints; and at a more global *itinerary* level, where a sequence of waypoints is determined, satisfying higher level constraints and optimizing higher level criteria. Local trajectory planning require a precise geometric model of the environment, possibly complemented with properties such as the nature of the terrain to traverse, whereas global itinerary planning reasons on a coarser representation that mainly expresses the cost of traversability for the mapped areas. Such a model is most often a region-based model, on which a traversability graph is defined. Additional information may be considered, *e.g.* the areas the robot will be able to observe along the itinerary, the possible communications along the itinerary, etc.

2) Environment observations: Environment observations pertain either to the mission at hand (*e.g.* exploring an environment or searching for a target) or to the robots own needs, *i.e. (i)* to ensure a proper localization or *(ii)* to enhance the knowledge on the terrain traversability. In both cases, the notion of *visibility* is to be considered first: it can be defined thanks to a 2.5 or 3D geometric model, from which the areas visible given a vantage point can be computed.

Localization is an essential process to ensure the *spatial consistency* of the built environment representations, the proper execution of the robot motions, and the achievement of the missions: planning observations targeted to localization often yields a better achievement of the robot tasks and motions. For the purpose of localization, observations that perceive already mapped features (landmarks) or areas are required, and therefore planning such observations exploits landmark maps or other maps from which *localizability models* can be derived: such models are used to express the expected localization precision as a function of the current localization precision and the planned observations. A map of GPS coverage, either pre-existing or built during the mission execution is also a localizability model.

When it comes to plan observations that augment the information contained in an environment model (of whatever type it is), the quantity of information (or its precision) encoded in the environment model is necessary to assess the interest of additional observations.

3) Planning communications: planning communications requires an environment model on which radio propagation models can be applied, so as to predict the link budget between two given positions. Radio propagation models are very complex, and require not only the geometry of the environment, but also the physical nature of the various volumes it contains – not to mention atmospheric conditions,

that also play a significant role. Most roboticists consider a conservative line of sight communication model [10], in which the link budget is only defined by the distance, which can be assessed on a 2.5 or 3D geometric model of the environment. An other solution is to indirectly learn the communication model by assessing areas in which communications have proven feasible.

4) Overall mission planning: Besides the three former classes of actions, planning the actions so as to achieve the given mission calls for specific data structures that mostly rely on the environment models: the notion of idleness of the visited areas is for instance a key element for patrolling missions [11], as is the quantity (precision) of the information encoded in the models for exploration missions. These "meta-information" on the environment, actually more related to the robots mission than to the environment itself, are grounded on the environment models.

B. A layered structure

No single environment model can represent the whole spectra of information required to plan these various processes. A series of environment models is required, each one encoding a particular layer of information with the proper data structure dedicated to its exploitation by a specific decisional process. One can view all these environment models, structured into a set of *layered representations*, as a Geographic Information System (figure 2): the main requests to this system are depicted in section III-B.

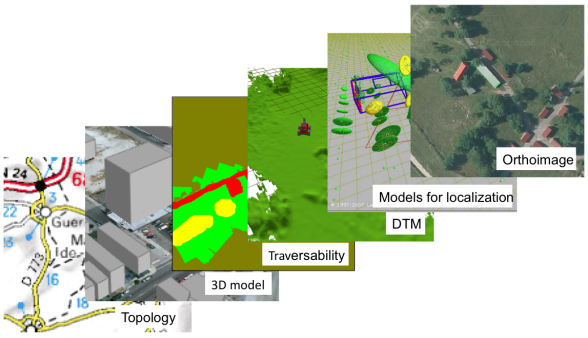


Fig. 2: Illustration of the various layers of environment models that may be required to autonomously operate a robot.

Geometry is of course at the core of all these models, that all represent spatial information. The spatial consistency of the models must be ensured, hence the importance of localization algorithms. Still, the robot position may not always be perfectly known, and then the environment model built from the acquired data are not spatially consistent: one must provide means to recover this consistency when more precise localization information become available, such as after a loop closure in SLAM-based localization, or a GPS fix after a period of GPS signal outages. The way we tackle this "re-localization" issue is depicted in section IV-A.

C. Illustration

This section presents an instance of an environment model made of three distinct layers, that are used on board our field robots, all equipped with a panoramic Lidar (figure 3). These three layers are a Digital Terrain Model (DTM), a coarser traversability map (the "region-map"), and a database of localized point clouds (the PCDdb). These three layers are built thanks to the following four modeling and localization following processes:

Fig. 3: The three robots Minnie, Mana and Momo

1) A Digital Terrain Model (DTM) 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, that presents a good compromise between expressiveness, simplicity and compactness.

On board the robots, a precise DTM is incrementally built in real time by merging point clouds acquired by the Lidars (PCD), and is used to plan local elementary trajectories, given a waypoint to reach. Merging is basically done by averaging the height of points perceived at each cell in the map (figure 4). Most importantly, the *quality* of each cell is memorized, by integrating a sensor noise model and the current robot pose uncertainty. This DTM can also be used to assess visibilities and communications between two given positions.

2) The region-map (RMap) is a coarser traversability map derived from the DTM, by analyzing the distribution of the normal vectors defined on patches of DTM cells. This model is exploited to plan long range itineraries, that defines the waypoint to be reached by the local trajectory planner. It also encodes the quantity of information, derived from the quality of the DTM cells (figure 4).

3) High rate localization is performed thanks to a visual-inertial SLAM approach [12] that does not memorizes landmarks over long ranges. However, a second localization algorithm that relies on scan-matching technique (similar to [13]) directly exploits memorized PCDs to ensure loop closures and long range localization.

4) A map-based localization approach matches PCDs with an initial model described as a DTM (for instance, most commercial aerial mapping systems generate DTMs along with orthoimages) to produce absolute localization estimate (*e.g.* as in [14]).

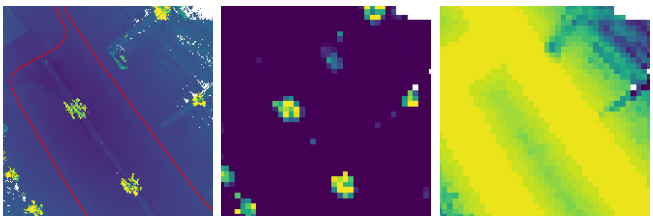


Fig. 4: From left to right: DTM (0.1 m resolution, colors encode height), 1 m resolution region-map derived from the DTM, and precision information encoded in the region-map. The modeled scene is an empty parking lot with sparse trees.

Figure 5 summarizes the relations established by these four processes between the initial data, the acquired data (PCDs), the localization information and the three layers of the model. The client processes that exploit the layers of the model are also shown.

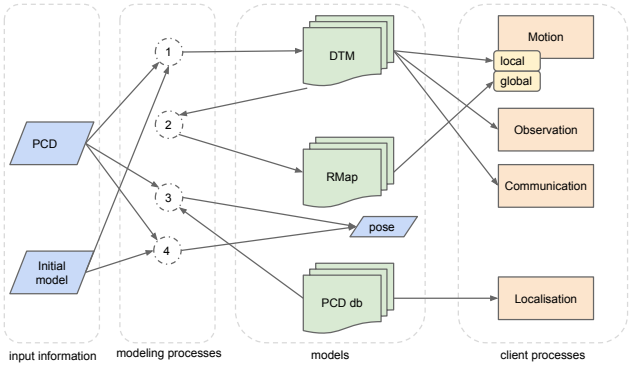


Fig. 5: An environment model composed of three layers (in green). The 4 numbers relate to the four modeling and localization processes sketched in the text, and the right column lists generic client processes.

III. Managing and Exposing Environment Models

To embed the various environment models and expose information to the client processes, the framework AtLaas structures these information in a dedicated database, so as to allow their dynamic management and the satisfaction of a series of external requests.

A. Spatial structuring: Pile of tiles

As it is done in most geographic information systems, all the data are indexed using a Cartesian structure of *tiles*. This classic way to hash spatial data provides numerous advantages to store and access the information. The tile structure is geo-referenced once and for all: it is their information content which is updated as information are gathered by the robot, or as "re-localization" events occur.

Each tile indexes the series of semantic layers required by the considered robot system. In our illustrative case presented section II-C, these are the DTM, the RMap and the acquired PCDs. Apart from the information encoded in these layers, the tiles exhibit some *meta-information* for each layer, that is required for their management and use. These meta information are:

- For each PCD, the best position estimate at which it has been acquired (associated with its covariance matrix), and the date of acquisition.
- For the DTM, the timestamp of its last update, the coverage (percentage of area modeled), and the associated "precision", which is heuristically derived from inner DTM information (inverse of the averaged variances of the height estimate in each DTM cell)
- Similarly, for the RMap, the timestamp of its last update, its coverage and precision.

To each tile are also associated meta-information derived from the embedded layers: number of PCDs, precision of the raster model and last update timestamp.

In order to handle arbitrary large maps, the tile structure is also used to manage the on-bard memory. A set of contiguous 3×3 tiles is kept in memory, the central tile being the one in which the robot is located. When the robot moves and enters one of the neighboring tiles, we dump the non-contiguous tiles to a file and load existing tiles if any. This is dampened by a hysteresis like mechanism (figure 6)

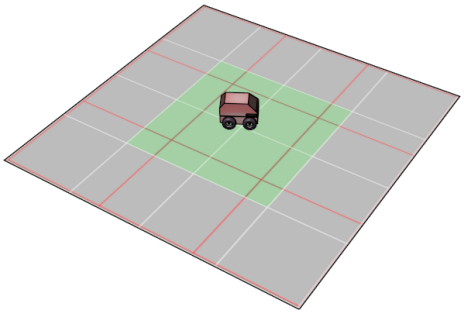


Fig. 6: Current map of 3×3 tiles. The tiles are delimited in red, and the green square and white limits represent the boundary around the current tile that enforce a hysteresis during tile shift: the robot is considered to exit the central tile only when it exits the green one square

The tile size is defined by the maximum sensor range, so that only the embedded layers of information associated to tiles in the current memory need to be updated when new data is acquired. Given the precision and resolution of the on-board Lidars, we set the tile size to 30 m – and the depth measures farther than 30 m are simply discarded from the PCDs. Tiles are referenced using the Universal Transverse Mercator (UTM) coordinate system, to be consistent with GPS position estimates and existing maps.

B. Model interfaces

Generally speaking, planning requires to "convolve" the environment model with the models of the actions to select, so as to assess their outcome and drive the search algorithms. But the action models may vary from one robot to the other: following a clear separation of concerns principle, the various environment models encoded in AtLaas are abstracted within a library that act as a server for the decisional processes.

Each function provided by the library has an argument that is the action model of the considered robot, and output the outcome of the evaluated action.

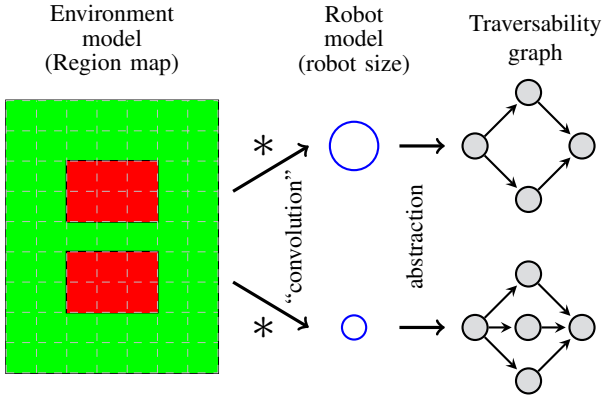


Fig. 7: Illustration of the model abstraction process, applied here to motion actions: by convolving an environment-centric (robot independent) region-map with two different robot motions models, two different accessibility graphs are produced.

Function	Action	Arguments	Returned value	Algorithm(s)
isAccessible	motion	r_m, p, c_{max}	$\{ p' \}$	Dijkstra
navigate	motion	r_m, p_1, p_2	$(path, c)$	A*, D*
canSee	perception	r_p, p	$\{ (p, \phi) \}$	ray tracing
test_comlink	com.	r_c, p_1, p_2	α	ray tracing

TABLE I: The main functions provided by the Gladys library. r_{action} denotes the action model of a robot, p a position, c a motion cost, and ϕ the quality of an information.

Table I presents the main interface between the layers encoded in AtLaas and the action evaluation algorithms exploited by the client planning processes. Motion planning requests are solved using either Dijkstra, A*, or D* shortest path-finding algorithms, exploiting a navigation graph, which is built upon the RMap as shown in figure 7, using cost from the RMap and the robot motion model r_m. For the perception tasks, the visibilities are assessed using Bresenham's line algorithm in the DTM, taking into account the robot's perception model r_p, which encodes the sensor field of view and quality. Finally requests regarding communications return a binary value depending on the visibility between two positions, and a maximum distance encoded in the communication model r_m. An open-source implementation ("Gladys") of these functions is available on-line [3].

IV. Exploiting AtLaas

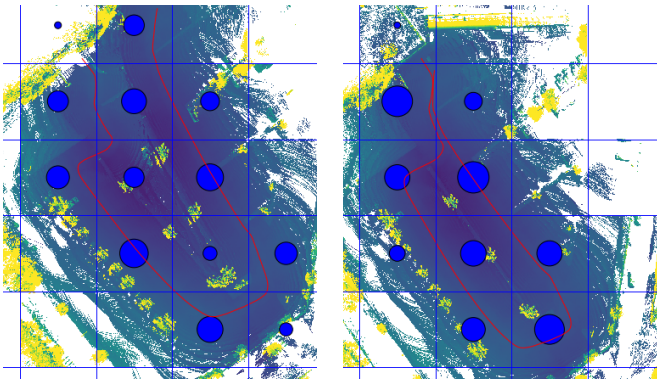
We present here two simple scenarios that exhibit the interest of the model structuring of AtLaas: dealing with a re-localization event, and cooperation between robots for autonomous navigation.

[3]https://github.com/pierriko/gladys

A. Handling re-localization

In case of position drift, the dense continuous environment models built during navigation (DTM and RMap) loose their spatial consistency. And after a re-localization event occurred, *e.g.* after a loop-closure or a GPS fix, the acquired data corrupt even more the models. The *simplest way* to recover the spatial consistency is to rebuild from scratch the environment models, using the newly estimated positions (note that the problem does not occur with landmark maps built by feature-based SLAM approaches: recovering the spatial consistency of the landmark maps is seamlessly achieved by the update of their position). To our knowledge, the sole contribution in the literature to maintain the spatial consistency of dense continuous models is [15], in which the dense model is defined by the concatenation of triangular patch models anchored to landmarks managed by a SLAM approach.

Figure 8a shows the DTM layer of a series of tiles built after a loop trajectory. The robot has strongly drifted, due to the use of wheel odometry only, which badly estimates rotations. A scan-matched ICP-based localization technique allows to close the loop, and a graph optimization re-estimates the best positions for the series of acquired PCDs [13]: figure 8b shows the DTM that is rebuilt after this re-localization. Note that the re-localization also impacted the overall tile structure, as some PCDs have been associated to new tiles. An average PCD produced by the Velodyne Lidar contains 300K points, and takes only 20ms to be fused in the DTM – the time to regenerate DTM is therefore bearable (deriving RMap from the DTM is even faster, and linearly depends on the processed surface).



(a) before re-localization (b) after re-localization

Fig. 8: DTM encoded in the tile structure before and after re-localization, for a loop trajectory. Red lines represent the trajectory, the blue grid represent the tiles, and circles diameters represent the number of PCDs associated to each tile. Note that re-localization induced changes in the affection of PCDs to the tile, and also that the resulting DTM is spatially consistent – see *e.g.* the trees along the top-left bottom-right diagonal, that were duplicated in the DTM before the loop-closure and trajectory optimization has been enforced.

B. Distributing environment models among robots

The tiles structure allow to manage and share environment models in an easy and efficient way. Here we illustrate how it can be used in a simple multi-robot scenario, in which one robot (R3) starts to navigate without any knowledge in an environment in which robots R1 and R2 have already started to evolve (*e.g.* to patrol an area – see figure 9). When robot R3 is tasked to reach a goal in such a situation, the best strategy is that it retrieves the available information on the environment to plan an itinerary.

To avoid heavy requests we propose to simply share data on a distributed client-server model. Each robot serves all its data via an HTTP server, and when needed can check on other servers whether they do contain data of interest or not, by simply parsing the tiles meta-data, encoded in XML files.

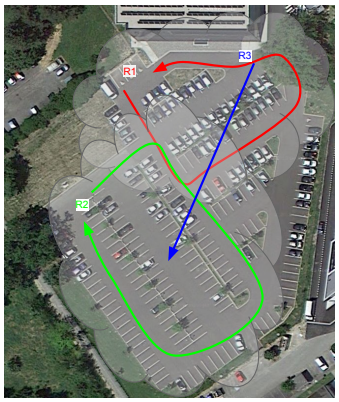


Fig. 9: A cooperative patrolling mission in a parking lot. R1 and R2 have already mapped parts of the parking when R3 starts its mission, knowing nothing about the environment. R3 is tasked to reach a waypoint in the middle of the parking (blue arrow).

The sequence diagram in figure 10 present a time-line of the requests and responses exchanged between the three robots. R3 first assesses the information it requires to reach its goal: a call to the "navigate" function on its empty RMap structure allows to identify the traversed tiles (the result of the A* algorithm is trivially a straight path) process will check whether there is data for running the A*, or if not, only compute the path with Bresenham line algorithm. We can then trivially deduce the required tiles from the resulting path.

R3 then gathers the meta-data on these tiles from the reachable robots (here R1 and R2), and assesses to which robot it will ask each tile, considering the tiles with the highest coverage are the better (the precision meta-data could have been considered to weigh the coverage). A second communication exchange is then launched, this time to effectively transmit the data contained in the tiles (the RMaps).

Once R3 knows which tiles it needs to reach its goal, it gathers all tiles meta-data from all reachable robots, the request can safely time-out, in which case, we just skip the server and try the next one. Figure 11 shows the itinerary

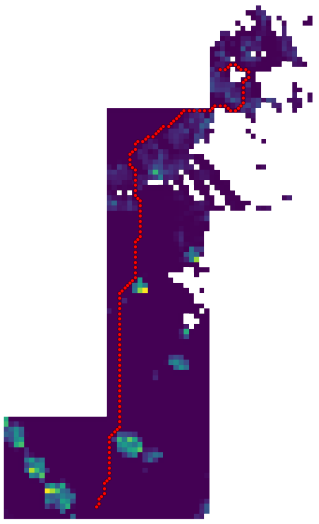


Fig. 11: The itinerary planned by R3 on the the basis of the RMaps collected from R1 and R2, result of a call to the "navigate" function (A*).

R3 could eventually compute. Note the RMap shown in this figure spans $4800\,\mathrm{m}^2$ and weights only 2KB: not a big bandwidth is required for such exchanges.

This solution is completely decentralized, it does not rely on a particular server, and connections a just assessed by testing HTTP GET requests – in case of timeout, the robot is considered out of reach. The current implementation requires each robot to have a list of URIs pointing to other servers data. But this list could be dynamic using service discovery, *e.g.* with Zeroconf.

V. Discussion

We have proposed a simple yet efficient approach to structure spatial related information acquired and produced on-board a robot. Besides exhibiting a simple API to expose the spatial information to client processes, the structure allows to maintain the spatial consistency of models after re-localization events, and can be used to build multi-robot cooperation schemes. Sharing the information on the environment is indeed a pre-requisite to cooperation, and can easily be achieved thanks to the proposed structure and through simple communication mechanisms. The proposed structuring is generic enough to handle additional layers of information, be they represented through a raster or a vector data structure.

Besides various improvements (e.g. managing tiles at different scales, as in [16], to allow more efficient data transfer), the main additional functionality to develop is to handle environment dynamics. This can be achieved by detecting and tracking dynamic elements at the lowest level of data processing, and defining an additional layer that encodes dynamic information in the tile structure.

The proposed framework embeds the various spatial environment modeling functions, and handles the evolution of the positioning errors and the asynchronous information

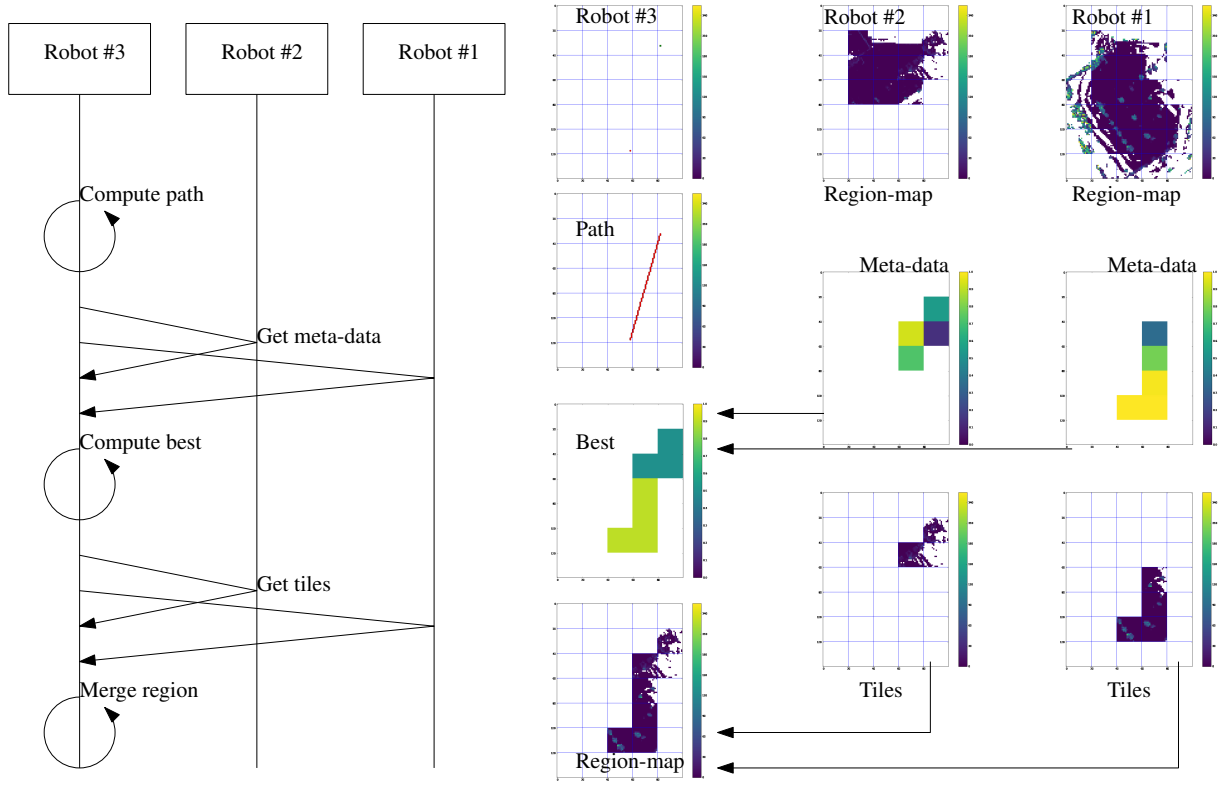


Fig. 10: Requests sequence diagram for the scenario of figure 9.The left part is the trace of the communications (requests), the right part illustrates the actual data flow between the robots.

transfers between robots. All these processes need to be actively triggered and controlled: further work is required to define a specific supervision scheme.

ACKNOWLEDGMENTS

This work was partially supported by a DGA-MRIS scholarship.

REFERENCES

[1] T. Krajník, J. P. Fentanes, J. ao Santos, K. Kusumam, and T. Duckett, "Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments," in *ICRA Workshop on Visual Localization in Changing Environments*, 2015, in review. [Online]. Available: http://strands.acin.tuwien.ac.at/publications/2015/krajnik_ICRA15_WVPRCE.pdf

[2] J. Santos, T. Krajník, J. P. Fentanes, and T. Duckett, "Lifelong exploration of dynamic environments," in *ICRA*, 2015, late breaking poster session. [Online]. Available: http://strands.acin.tuwien.ac.at/publications/2015/santos_ICRA15_LBP.pdf

[3] D. T. Cole, P. Thompson, A. H. Göktogan, and S. Sukkarieh, "System development and demonstration of a cooperative UAV team for mapping and tracking," *International Journal of Field Robotics*, vol. 29, no. 11, pp. 1371–1399, 2010.

[4] T. Vidal-Calleja, C. Berger, J. Solà, and S. Lacroix, "Large scale multiple robot visual mapping with heterogeneous landmarks in semi-structured terrain," *Robotics and Autonomous Systems*, vol. 59, no. 9, pp. 654–674, Sept. 2011.

[5] G. A. Hollinger, S. Yerramalli, S. Singh, U. Mitra, and G. S. Sukhatme, "Distributed Data Fusion for Multirobot Search," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 55–66, Feb. 2015.

[6] A. Khan, E. Yanmaz, and B. Rinner, "Information Merging in Multi-UAV Cooperative Search," in *IEEE International Conference on Robotics and Automation*, 2014.

[7] T. Cieslewski, S. Lynen, M. Dymczyk, S. Magnenat, and R. Siegwart, "Map API - Scalable Decentralized Map Building for Robots," in *IEEE International Conference on Robotics and Automation*, 2015.

[8] G. Mohanarajah, V. Usenko, M. Singh, R. D'Andrea, and M. Waibel, "Cloud-based collaborative 3d mapping in real-time with low-cost robots," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 423–431, April 2015.

[9] A. Stentz and M. Hebert, "A complete navigation system for goal acquisition in unknown environments," *Autonomous Robots*, vol. 2, no. 2, Aug. 1995.

[10] L. De Floriani, P. Magillo, and E. Puppo, "Line of sight communication on terrain models," *Intl. J. Geographic Information Systems*, vol. 8, no. 4, pp. 329–342, 1994.

[11] A. Almeida, G. Ramalho, H. Santana, V. Corruble, and Y. Chevaleyre, "Recent Advances on Multi-Agent Patrolling," *Advances in Artificial Intelligence*, 2004.

[12] C. Roussillon, A. Gonzalez, J. Solà, , J.-M. Codol, N. Mansard, S. Lacroix, and M. Devy, "Rt-slam: a generic and real-time slam architecture," in *International Conference on Vision Systems, Sophia Antopolis (France)*, Sept. 2011.

[13] R. Valencia, J. V. Miro, G. Dissanayake, and J. Andrade-Cetto, "Active pose slam," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[14] B. Desrochers, L. Jaulin, and S. Lacroix, "Set-membership approach to the kidnapped robot problem," in *IEEE/RSJ International Conference on Robotics and Automation, Hamburg (Germany)*, Sept. 2015.

[15] J. Nieto, J. Guivant, and E. Nebot, "DenseSLAM: Simultaneous localisation and dense mapping," *International Journal of Robotics Research*, vol. 25, no. 8, pp. 711–744, Aug. 2006.

[16] K. P. Joan Mas and N. Juli, "Opengis web map tile service implementation standard," Mar. 2010. [Online]. Available: http://www.opengeospatial.org/standards/wmts