Robotic mapping: an architectural approach

Simone Bonetti, Francesco Fiamberti, Daniela Micucci and Francesco Tisato[1]

Abstract— **Robotic mapping aims at building a spatial model of a physical environment. The usage of low-cost sensors to perceive the environment poses interesting challenges because acquisition and related elaboration should be carefully driven in order to balance costs and benefits. We face the problem from an architectural perspective. The result is a modular, open, and scalable software architecture.**

I. Introduction

The issue of robotic mapping concerns the acquisition of spatial models of physical environments through mobile robots [1]. Many approaches to robot mapping have been proposed, which differ in the sensors they exploit and in the algorithms they use to update the world model.

The use of low-cost devices (i.e., micro-controller and sensors) is particularly interesting but poses some interesting challenges. Indeed, the sensed data may be inaccurate and their acquisition and elaboration may result in high computational costs.

A multisensorial approach with heterogeneous sensors may compensate for the defects of the individual sensors but does not offer any solution to reduce computational costs. Such a reduction may be achieved by activating the acquisitions so as to maximize the gain while minimizing the computations. This turns into making the activation of the sensors *adaptive* to the actual *context*, which includes the actual effort in performing an acquisition/elaboration combined with the gain in terms of world map enrichment. Thus, the activation of the individual sensors is decided dynamically, according to the effort spent to sample an area of the environment.

It is also important that who decides whether a sensor must be activated or not is completely unaware of why a certain area should be explored and sampled. This allows decoupling the application strategy (what is to be explored) from the activation strategies (what actually will be sampled) that can depend on the specific sensor. For example, if the area to be explored and sampled is large and the exploited sensors are a camera (that generally acquires a lot of information on the environment with a single activation) and a laser range finder (that, with a single activation, acquires very little information on the environment), it is reasonable to think that the strategy driving the camera will decide to perform the acquisition, whereas the strategy that drives the laser will decide that the acquisition is disadvantageous. This allows to reuse the same application strategy with different sets of sensors and, vice versa, to reuse the same set of sensors with different strategies. For example, the same hardware and sensor components may be used both for an application dedicated to accurate but slow mapping of an environment, and in an application focused on simpler but fast mapping of small areas close to the mobile entity.

The above considerations lead to think that a holistic approach is inadequate as usually results in huge and monolithic pieces of software that intermix strategic decisions and technological issues related to sensors.

An architectural approach founded on separation of concerns may help in devising a software architecture whose *modularization* (i.e., partitioning the software specification into a number of modules that together satisfy the original problem statement [2]) yields at least openness, reuse, and scalability.

We propose an architectural solution that carefully takes into consideration the above suggestions. The result is a layered architecture that includes, from the bottom upwards, *sensors*, *integration*, and *application* layers.

II. The architecture

The *sensors* layer deals with physical sensors. Each sensor is encapsulated in a software module called *sensor component*, whose aim is to acquire and elaborate raw data. Data are contextualized in a spatial model that is provided to the upper layer (integration). Each sensor component encapsulates sensor dependant details by providing a common, single, homogeneous representation of the world. Such a representation is termed *Sensor Occupancy Map* (SOM), that is, an occupancy map [3] whose probabilities are updated when new data are available. A probability value associated with a cell indicates the likelihood that the cell is free or occupied.

At the *integration* layer, the *world map builder* component reads data from all the SOMs and produces the *World Occupancy Map* (WOM), which is assumed to be the best possible representation of the environment according to the available data. Likewise SOMs, the WOM is an occupancy map. The internal policies and algorithms used in the world map builder are application-dependant and are not constrained at all by the architecture.

The *application* layer hosts the *application strategy* component, whose task is to analyze the current state of the world occupancy map (WOM) and to decide which areas must be investigated in the immediate future. The application strategy lets the lower layer (integration) know its choices by

[1]Department of Informatics, Systems and Communication, University of Milano-Bicocca, Viale Sarca 336, 20126, Milano, Italy
simone.bonetti987@gmail.com,
francesco.fiamberti@disco.unimib.it,
daniela.micucci@disco.unimib.it,
francesco.tisato@disco.unimib.it

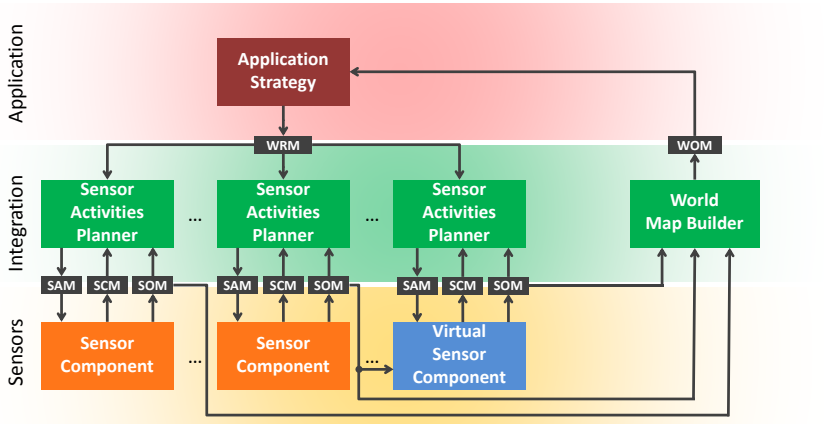

Fig. 1. Complete architecture

means of another occupancy map, the *World Relevance Map* (WRM), where each cell is associated to a relevance. Again, nothing is said about the internal policies adopted by the application strategy.

The loop is completed by a set of control components at the integration layer. For every sensor component, a *sensor activity planner* reads the WRM and determines which cells should actually be analyzed by the sensor. The choice is made by analyzing the current SOM and the current *Sensor Cost Map* (SCM), both provided by the associated sensor component. Likewise all the introduced maps, the SCM is an an occupancy map that is made available by each sensor component to expose the cost of the analysis of each cell by the associated sensor. The cells that must be analyzed are communicated to the associated sensor component by means of another occupancy map called *Sensor Attractiveness Map* (SAM). The sensor component in the sensors layer analyzes the SAM to opportunely drive the associated sensor. Again, nothing is said about the strategy used to select the cells to be inspected. It is reasonable to think that the strategy will balance costs and benefits. Moreover, it is possible to use the same algorithm for all the sensor activity planners, or to exploit a different algorithm for each component.

The basic architecture can be enriched by introducing the concept of a *virtual sensor component*. A virtual sensor component behaves like a standard sensor component, but instead of directly interfacing with a physical sensor, it gets data from the SOM of one or more sensor components. Virtual sensors allow for higher modularization. In fact, a virtual sensor may perform additional elaboration on data acquired by a physical sensor, without requiring that such computations are performed directly by the component in charge of managing the physical device. Complex hierarchies and arrangements of sensors can be realized by exploiting the concept of virtual sensors, with the only constraint that a virtual sensor component must take data from one or more SOMs, whereas a standard sensor component produces data on its own by directly interfacing with a physical device.

The complete architecture is shown in Figure 1. All the described components are designed to work independently of each other. Every communication between components is asynchronous and realized by means of the different occupancy maps, which are the only shared data.

III. Conclusions

Preliminary tests of the proposed architecture have been performed using a mobile entity featuring a laser range finder (mounted on a rotating support) and a fixed CMOS camera (see Figure 2). The future developments will focus mainly on additional testing in different scenarios.

To conclude, a system based on the proposed architecture may be easily realized by:

- specifying/selecting (if already defined) an application strategy that fulfils domain-dependant constraints/requirements;
- specifying/selecting (if already defined and available from a library) the set of sensor components and related sensor activity planners;
- specifying/selecting (if already defined) the world map builder.

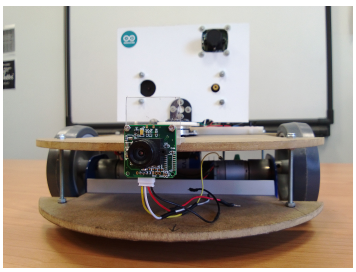

Fig. 2. The mobile entity

References

[1] Sebastian Thrun. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[2] David N. Card, Gerald T. Page, and Frank E. McGarry. Criteria for software modularization. In *Proceedings of the 8th international conference on Software engineering*, ICSE '85, page 372377, Los Alamitos, CA, USA, 1985. IEEE Computer Society Press.

[3] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46 –57, June 1989.