A Role-Based Architecture For A Mobile Robot

Matteo Golfarelli*, **Stefano Rizzi****

DEIS - University of Bologna
Viale Risorgimento, 2
Bologna
Italy

Abstract. In this paper we describe the control architecture we are experimenting on a Pioneer I mobile platform. Its building blocks are called roles and are characterized by establishing a cognitive parallel with the roles played by the different crew members on a submarine entrusted with navigation tasks on unexplored soundings. The different behaviors are determined by a team-work involving all roles; thus, the definition of role is orthogonal to that of behavior. On the other hand, roles are tightly related to the knowledge architecture adopted to represent the environment, and reflect the separation between symbolic and sub-symbolic representation of the environment enforced by the presence of landmarks. After describing each role and its activities, and outlining the inter-role pattern of communication, we present some experimental tests.

1 Introduction

A robotic agent operating in dynamic and unknown worlds must be capable of exploring and examining the surrounding environment in order to execute appropriate actions for achieving its goals. In order to have analysis and action organized effectively, it is necessary to define how data from sensors can be combined and integrated, and with which priority the different perceptions from the environment should affect the robot's behavior. The role of a control architecture for a mobile robot is that of mapping the sensory information and the knowledge previously extracted into actions aimed at carrying out the tasks assigned.

The main issues to be considered when designing a control architecture concern the choice between centralized and distributed architecture, reactive and deliberative behavior, sensor fusion and command arbitration, bottom-up and top-down control. Even if some solutions explored in the literature are based on drastic choices, intermediate solutions often proved to work better [18]. In the following, the main categories of architectural approaches proposed in the literature are briefly outlined.

- *Deliberative planners* [14]. The key element of the architecture is a planner capable of determining the optimal sequence of actions to take the system from the initial into the target state. The plan produced is complete and can be directly executed. The main drawback is the high computational complexity of the analysis process.
- *Reactive architectures* [2]. Instead of building a complete model of the environment, reactive systems directly react to sensorial perceptions; no explicit evaluation of the action consequences is made. Since the agent can neither rely on an internal representation of the environment nor plan its actions, it can identify its goal only if it can be reached directly; besides, local optima in achieving the goal can hardly be avoided.
- *Layered architectures* [4]. They are structured as distributed systems in which the modules are organized on multiple control levels operating with different data granularity, at different abstraction levels and different time scale. This is aimed at achieving both deliberative (in the medium-long term) and reactive behaviors. Unfortunately, this kind of architecture turned out to be not very flexible.
- *Behavior-based architectures* [1]. They are still distributed systems where each module, instead of implementing an activity, achieves an elemental behavior and encapsulates perception, planning and action capabilities. These architectures are very robust, but it is hard to predict the overall system behavior which is defined by the interaction of the elemental behaviors with the environment.

* Email: mgolfarelli@deis.unibo.it
** Email: srizzi@deis.unibo.it

- *Blackboard architecture* [15]. The components communicate via a central database: the modules indicate their interest in certain data, and the database returns them the data required, if they are available. No explicit control flow between the modules is considered.
- *Subsumption architecture* [1]. It is based on a decomposition of the control system into layers, each achieving a different set of behaviors; each layer subsumes the lower levels, and can inhibit them by suppressing their outputs. There is no central control module.

In this paper we describe the control architecture we are experimenting on a Pioneer I mobile platform. Its building blocks are called *roles* and are characterized by establishing a cognitive parallel with the roles played by the different crew members on a submarine entrusted with navigation tasks on unexplored soundings. The different behaviors (react, explore, navigate, etc.) are determined by a team-work involving all roles; thus, the definition of role is orthogonal to that of behavior. On the other hand, roles are tightly related to the knowledge architecture adopted to represent the environment, and reflect the separation between symbolic and sub-symbolic representation of the environment enforced by the presence of *landmarks*.

In Section 2 a brief insight into the robot architecture and the environmental knowledge is given. Section 3 presents our control architecture, describes each role and its activities, and outlines the inter-role pattern of communication. Finally, Section 4 compares our approach with other well-known architectures, while Section 5 proposes some experimental results.

2 The robot and its knowledge

Several authors agree that navigation in autonomous robots should be accomplished by coupling reactive motion control and capability of planning paths at higher levels of abstraction [6]. Reactive behavior asks for a well-rooted correspondence between entities in the real world and their internal representation, and requires essentially local sensor-based information. On the other hand, too much detail in the description may become overwhelming for high-level path planning which is more easily carried out by adopting a symbolic representation where useless details are neglected.

These requirements cannot be easily met if only one representation formalism is adopted; thus, in [12] we argued that navigation-oriented knowledge of a structured environment should be represented at least at two levels, symbolic and sub-symbolic, using different formalisms. Assuming that it is possible to identify a set of landmarks within the environment, we outlined a hybrid approach to execution of navigation tasks, in which landmarks draw a "boundary line" in the environmental knowledge, corresponding to the separation between path planning and trajectory planning. The former is carried out on a hierarchical graph whose vertices and arcs correspond, respectively, to landmarks and inter-landmarks routes (*symbolic layer*), and produces sequences of landmarks to be visited; the latter is carried out in the space between landmarks, by means of a local sub-symbolic representation (*sub-symbolic layer*).

The Pioneer I platform (Real World Interface Inc.) on which we operate is equipped with an array of sonars, a compass and a camera. The robot moves in an office environment, where it recognizes simple shapes as landmarks by analyzing the images from the camera and the sonar patterns. Inter-landmark motion is based on local occupancy grids; obstacle avoidance is sonar-based. No absolute positioning sensors are used; the positions of landmarks are estimated by combining the compass readings with odometric measures, and metric error is reduced by adopting an error correction algorithm. Exploration of the environment, which is initially unknown, is supported by a graph-exploration algorithm and an agenda. High-level path planning is carried out on the symbolic layer by means of an heuristic algorithm which takes into account complex time and precedence constraints.

3 The control architecture

The distinction between a sub-symbolic and a symbolic world is strongly reflected in our control architecture which, coarsely, consists of three levels:

- *reactive level*, operating on sub-symbolic knowledge;
- *deliberative level*, operating on symbolic knowledge;
- *intermediate level*, carrying out symbol grounding (link between the sub-symbolic and symbolic worlds).

Within these levels, the behavior of the robot originates at each time from a team-work involving six roles, which operate simultaneously on different layers of knowledge. A *role* implements a set of functionalities and is characterized by the following properties:

- For each role, all functionalities operate on the same abstraction level and are strictly related.
- The knowledge supporting each role is not shared with other roles.
- The control flow between the roles is not hierarchical: each member has wide autonomy within its role and its abstraction level.

Inter-role coordination is asynchronous and message-based. Messages are used both for exchanging data and for generating control flows.

In order to characterize the architecture from the cognitive point of view, we consider the analogy between our robot and a submarine, operated by a crew where each member plays a different role in navigation. The data and control flows between the roles are outlined in Figure 1. The messages exchanged between the

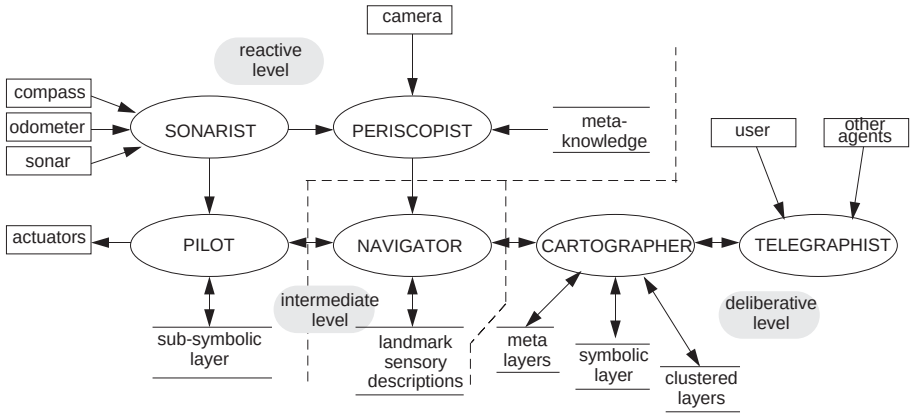


Fig. 1.: Data-flow diagram for the role-based architecture; arrows represent data flows, ellipses represent roles

different roles during a common task execution scenario are sketched in Figure 2 using the UML formalism, and will be explained in the next subsections.

3.1 The sonarist

The *sonarist* reads data from the sonar, the electronic compass and the odometer mounted on the robot. He and the periscopist are the only crew members who have direct vision of the outside world.

The sonarist composes the readings from the compass and the odometer in order to estimate the current position of the robot with reference to a relative Cartesian system. He also combines sequences of sonar patterns into a local two-level occupancy grid describing the current neighborhood of the robot. Figure 3 shows the combination of a set of occupancy grids for a room in which three landmarks are present. The current neighborhood is associated to the positional estimate, and both are transmitted to the periscopist and to the pilot (*poseAndGrid* message).

Since the sonarist reads directly the sonar patterns, he is also entrusted with the task of recognizing potentially dangerous situations for the robot, typically unexpected obstacles or moving objects, and of informing the pilot (*warning* message).

3.2 The periscopist

The role of the *periscopist* is to identify landmarks in the environment. To this end, he can operate a camera mounted on a mobile support capable of executing pan and tilt movements; this enables the periscopist to examine not only the direction strictly ahead of the robot, but the whole half-circle in front of it.

Landmark recognition is based on a set of sample patterns corresponding to different *categories* of landmarks. Some categories may be characterized by a sonar pattern, some by a visual pattern, some by both.

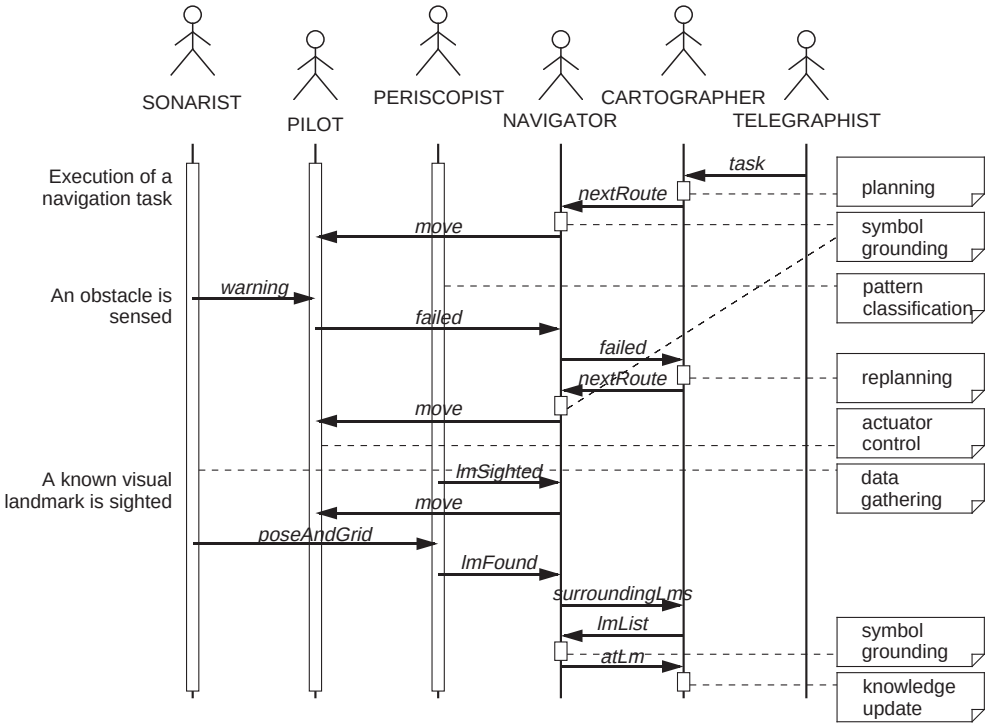
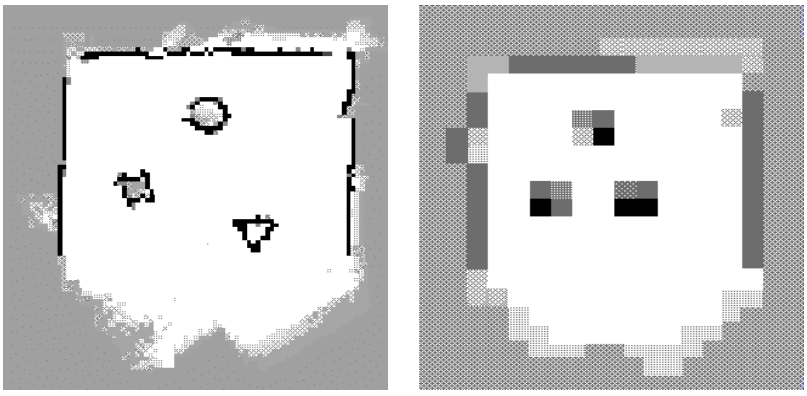


Fig. 2.: Sequence diagram for a possible scenario occurring during the execution of a navigation task (time flows downwards). According to the UML formalism, white rectangles represent actors' activities



(a) (b)

Fig. 3.: A two-level occupancy grid: fine-grained (a) and coarse-grained (b)

Sonar landmarks are recognized by comparing the local occupancy grid transmitted by the sonarist with the sample patterns [7]; in particular, the coarse-grained grid is used to decide when and where template matching is useful, while the fine-grained one is used for template matching. Visual landmarks should be recognized even if the image is distorted or partially hidden by opaque obstacles [3]; for each sighting of a "promising" pattern, the periscopist transmits the direction and estimated distance of the potential landmark to the navigator (*lmSighted* message), who may decide to verify the sighting by asking the pilot to take the robot towards the supposed landmark. As to categories having both a visual and a sonar description in the meta-knowledge, the object is accepted as landmark only if both sensors agree. When a landmark is classified with a sufficient degree of reliability, its sensory description and its category are signaled to the navigator, together with its estimated position (*lmFound* message).

Figure 4 the shows the artificial pattern which identifies visual landmarks. Using this pattern, the pose of landmarks can be precisely estimated even using a monocular device. Landmark recognition is carried out searching the environment for sequences of equidistant color gaps and then reconstructing the landmark

Fig. 4.: The pattern identifying a visual landmark

pose by means of the inverse projection technique [10].

3.3 The pilot

The *pilot* controls directly the actuators and moves the robot. He consults the sub-symbolic layer, which he builds upon the data transmitted from the sonarist, to plan trajectories in the environment. The sub-symbolic layer is structured as a set of local occupancy grids, each describing the area which surrounds a route [5]. Using local instead of global grids keeps wrong local positional estimates from affecting the whole knowledge [17].

The navigator tells the pilot in which direction to head the robot; the pilot computes the detailed trajectory considering the distribution of occupancy probability. The sonarist may warn the pilot of a potential collision with unexpected obstacles: people, other moving robots and new objects in the environment; also a known object may become an unexpected obstacle due to an error in the estimate of the current position. The pilot, depending on the signaled gravity of the risk and considering the sub-symbolic knowledge, decides which strategy to adopt: move away quickly if the risk is high, replan a trajectory for reaching the target landmark by avoiding the obstacle if the risk is low. If he cannot reach the target, he informs the navigator (*failed* message).

3.4 The navigator

The *navigator* manages the correspondence between symbolic and sub-symbolic knowledge (*symbol grounding*); thus, he is in contact with both the cartographer and the pilot. The cartographer suggests the navigator which route should be followed; the navigator, in turn, tells the pilot to head the robot in the corresponding direction (*move* message).

Every time the periscopist signals that a landmark of a given category has been reached, the navigator must determine and communicate to the cartographer whether that landmark has already been experienced (*atLm* message) or not (*newLm* message). To this end, he asks the cartographer for the list of the surrounding landmarks (*surroundingLms* message), and compares their descriptions (obtained by *lmList* message) with the current one.

3.5 The cartographer

The *cartographer* operates on the symbolic side of environmental knowledge. When the robot moves within an unknown area of the environment, he is informed of the landmarks being experienced and of their positions and categories by the navigator, and updates the symbolic layer accordingly. Routes are stored in the symbolic layer as well; each represents an abstraction of a physical trajectory between two landmarks, and is associated to a *cost* expressing the length of this trajectory.

The cartographer is involved in three basic activities:

- *Exploration.* The cartographer carries out exploration at the symbolic level, aiming to acquire knowledge of the graph representing the environment. Every time the robot reaches a landmark, the navigator informs the cartographer of the routes which presumably exit from that landmark; the cartographer decides which route the navigator should follow next in order to accomplish a given exploration strategy [13].

- *Error Correction.* The estimates of landmark positions transmitted by the sonarist are made unreliable by the errors from metric sensors. In order to correct positional errors, the cartographer considers each new estimate within the framework of the map global knowledge; the relevance given to the former depends on the reliability of the latter. The correction technique we adopt is carried out in two phases: the first is aimed at eliminating the metric inconsistencies which may appear in the symbolic knowledge when unknown routes are experienced; the second is aimed at improving the estimates on the landmark positions when known routes are followed again [9].
- *Path Planning.* The cartographer is informed by the telegraphist of the tasks the user is requesting; in general, each task is characterized by one or more actions to be executed on a resource in the environment. On the symbolic level, a *path* is a sequence of adjacent landmarks; its cost is the sum of the costs of the routes which connect each pair of consecutive landmarks. The cartographer must thus be capable of planning the cheapest path which allows for a set of resources to be visited, in observance of temporal and precedence constraints. Once the cartographer has planned a path, he transmits the routes to be followed, one by one, to the navigator (*nextRoute* message). If the pilot gets lost, or if some tasks could not be executed for any reasons (an unexpected change in the environment, a wrong estimate of the time required to perform a task, etc.), the navigator informs the cartographer who adapts the previous plan to cope with the new situation (*failed* message).

3.6 The telegraphist

The telegraphist manages the communications between the robot and the other agents who may inhabit the environment, including the users. Communication with artificial agents may be aimed for instance at cooperation during exploration or to task negotiation during path planning [13]. The negotiation protocol we implemented is an extension of the Contract Net Protocol [16]; it assumes that no explicit support for utility transfer is available and thus is based on task swaps [8]. The Contract Net Protocol requires the telegraphist to evaluate the utility of a swap (*evaluateSwap* message); this utility is estimated by the cartographer on the symbolic knowledge in terms of the reduction in the execution cost and is returned to the telegraphist (*swapUtility* message). If the swap takes place, the telegraphist informs the cartographer (*swap* message) who replans the new path.

Communications usually take place also between the telegraphist and the users, who may be interested in monitoring the robot behavior, in ordering the robot to execute tasks (*task* message), or in querying the symbolic knowledge of the environment. sectionArchitecture analysis Even if our role-based architecture belongs to the distributed architectures category, the way the tasks are decomposed is innovative. Though commands flow from the telegraphist to the pilot through the cartographer and the navigator, each role interprets the commands received and decides whether to execute or delay them. Nevertheless, the agents behavior is still guided by a plan generated at a high level of abstraction, which ensures good performance even for complex tasks.

In particular, the presence of roles impacts on the planning activity. As already stated, roles can be classified as belonging to three different levels: deliberative (telegraphist and cartographer), reactive (pilot, sonarist and periscopist) and intermediate (navigator). Planning is carried out at the deliberative level and produces a sequence of landmarks and routes to be covered. The navigator is responsible for filling the gap between each route and the physical trajectory allowing navigation in the real environment. As proved by experiences on real robots, the percentage of plan failures increases proportionally to the level of detail of the plans [11]. This is a consequence of the impossibility of keeping the environment dynamism and sensor errors into account in medium and long term plans. At the deliberative level, the robot is *guided* instead of being directly *controlled*.

The choice of supporting each role with a specific knowledge representation formalism encourages independence between roles and enhances their capabilities. Differently from behavior-based systems, the different knowledge representations are related, though not strictly. Thus, the bottleneck due to continuously integrating data is overcome, yet the actions of the different roles are based on consistent information.

Message-based communication and the adoption of independent knowledge representations makes each role an atomic unit which can be modified independently of the others, assuming that the inter-role interface is stable. Hence, roles represent robust modules to be used for system design.

4 Conclusions and tests

In this paper we presented a new architecture for mobile robot control. The architecture is characterized by roles which correspond to the architecture modules. Our solution tries to overcome the drawbacks of the previous architectures by providing at the same time a wide autonomy to each role and a hierarchical control flow. We are currently implementing the architecture on a Pioneer I robotic platform; in the following we report the results of some experimental tests.

Figure 5 shows the trajectory followed by the robot while exploring an unknown environment where visual landmarks are present. In approaching the landmarks sighted, the robot follows trajectories shaped as

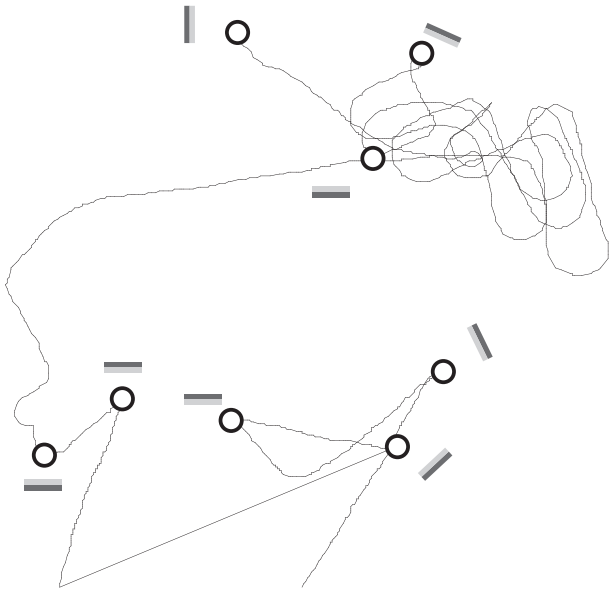


Fig. 5.: Exploration of a $27 \times 27m$ arena with visual landmarks, represented by colored segments (the visual pattern is on the light-gray side); circles represent the positions the robot had when landmarks were recognized

Bezier curves. In the top right area of the figure, the robot wanders around for a while since the landmark is too oblique to be recognized.

Figure 6 shows the trajectory followed by the robot while exploring an unknown environment where sonar landmarks are present. As the robot moves around each landmark, its confidence about the correct

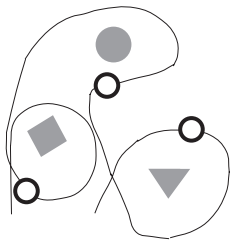


Fig. 6.: Exploration of a $9 \times 9m$ arena with sonar landmarks, represented by their shapes; circles represent the positions the robot had when landmarks were recognized

landmark category increases; it is remarkable that a satisfactory level of confidence is already obtained when only a partial view of the landmark is available to the robot.

Figure 7 shows a navigation test in which the robot was asked to plan a path to reach landmark D starting from A; the path planned entails reaching landmarks B and C before D. Obstacle avoidance is

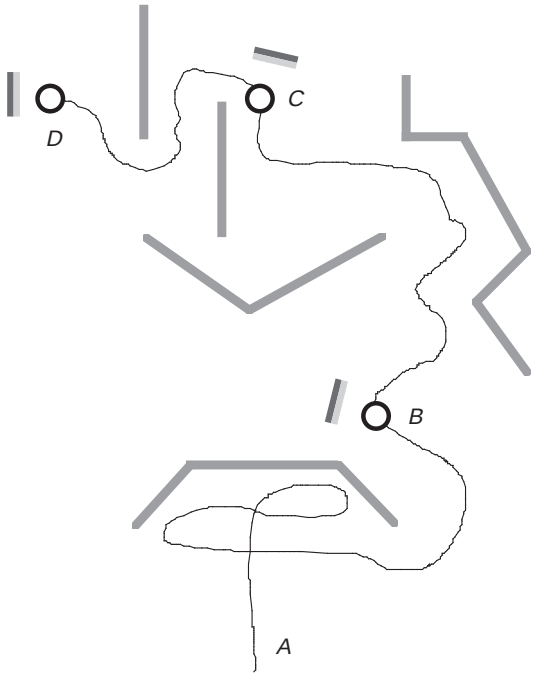
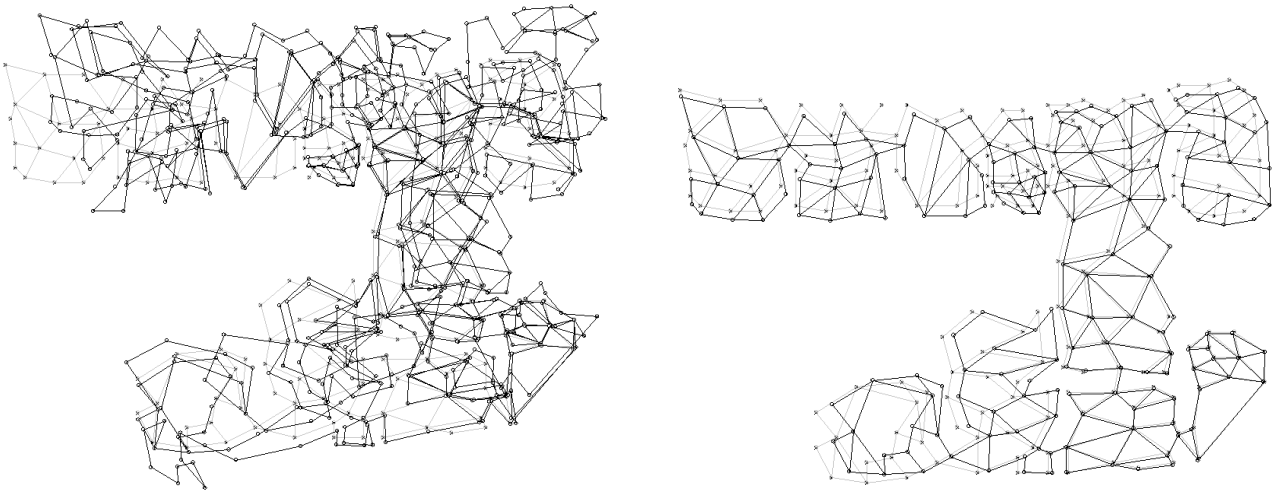


Fig. 7.: Execution of a simple navigation task

based on a neural network.

Figure 8 shows the result of a simulation conducted on the map of a real environment in order to test the error correction algorithm. Table 1 reports the average errors on the length and orientation of the routes



(a) (b)

Fig. 8.: Error correction. The real map is in light gray; the measured one (a) and the corrected one (b) in black

before correction and after the map has been covered repeatedly. The reason why the error appears to increase after the first tour is that, as already stated in Section 3.5, the first correction is primarily aimed at restoring topological consistency.

	error on length (%)	**error on orientation** (*rad*)
before correction	6.8	0.071
after one tour	7.3	0.079
after two tours	4.9	0.049
after three tours	3.6	0.035
after four tours	2.8	0.031
after five tours	2.7	0.030

Table 1.: Average error on route length and orientation during exploration for the map in Figure 8

References

1. R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, 1986.
2. R.A. Brooks. Intelligence without Reason. In *Proc. Int. Joint Conf. Artificial Intelligence*, 1993.
3. T. D'Orazio, F.P. Lovergine, M. Ianigro, E. Stella, and A. Distante. Mobile robot position determination using visual landmarks. *IEEE Trans. Industrial Electronics*, 41(6):654–662, 1994.
4. A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, RA-3(3):249–265, 1987.
5. A. Elfes. Occupancy grids: a stochastic spatial representation for active robot perception. In S.S. Iyengar and A. Elfes, editors, *Autonomous mobile robots*, volume 1. IEEE Computer Society Press, 1991.
6. E. Gat. On the role of stored internal state in the control of autonomous mobile robots. *AI Magazine*, 14(1):64–73, 1993.
7. M. Golfarelli, D. Maio, and S. Rizzi. A Hierarchical Approach to Sonar-Based Landmark Detection in Mobile Robots. In *Proc. 5th Symp. Intelligent Robotics Systems*, pages 77–84, Stockholm, Sweden, 1997.
8. M. Golfarelli, D. Maio, and S. Rizzi. Multi-Agent Path Planning Based on Task-Swap Negotiation. In *Proc. 16th UK Planning and Scheduling SIG Workshop*, pages 69–82, Durham, England, 1997.
9. M. Golfarelli, D. Maio, and S. Rizzi. Elastic Correction of Dead-Reckoning Errors in Map Building. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robotic Systems*, pages 905–911, Victoria, Canada, 1998.
10. R.C. Gonzales and R. E. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, 1992.
11. J.C. Latombe. *Robot motion planning*. Kluivert Academic Publishers, Norwell, MA, 1993.
12. D. Maio, D. Maltoni, and S. Rizzi. Dynamic clustering of maps in autonomous agents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(11):1080–1091, 1996.
13. D. Maio and S. Rizzi. A Multi-Agent Approach to Environment Exploration. *Int. Journ. Cooperative Information Systems*, 5(2-3):213–250, 1996.
14. N. Nilson. *Principles of Artificial Intelligence*. Tioga Pub. Company, Palo Alto, CA, 1980.
15. S. Shafer, A. Stentz, and C. Thorpe. An Architecture for Sensor Fusion in a Mobile Robot. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 1986.
16. R.G. Smith. The Contract Net Protocol: high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, 1980.
17. C.J. Taylor and D.J. Kriegman. Exploration strategies for mobile robots. In *Proc. IEEE Int. Conf. on Robotics and Automation*, Atlanta, Giorgia, 1993.
18. M.J. Wooldridge and N.R. Jennings. Intelligent Agents. In *ECAI-94 Workshop on Agent Theories, Architectures and Languages*, Amsterdam, The Netherlands, 1994.

This article was processed using the TEX macro package with SIRS99 style