Available online at www.sciencedirect.com
SCIENCE DIRECT®
Robotics and Autonomous Systems 52 (2005) 290–311
Robotics and Autonomous Systems
www.elsevier.com/locate/robot

Sensor-based robot motion generation in unknown, dynamic and troublesome scenarios

Javier Minguez*, Luis Montano

Instituto de Investigación en Ingeniería de Aragón, Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, Spain

Received 28 May 2004; received in revised form 15 May 2005; accepted 3 June 2005
Available online 28 July 2005

Abstract

A sensor-based motion control system was designed to autonomously drive vehicles free of collisions in unknown, troublesome and dynamic scenarios. The system was developed based on a hybrid architecture with three layers (modeling, planning and reaction). The interaction of the modules was based on a synchronous planner–reactor configuration where the planner computes tactical information to direct the reactivity. Our system differs from previous ones in terms of the choice of the techniques implemented in the modules and in the integration architecture. It can achieve robust and reliable navigation in difficult scenarios that are troublesome for many existing methods. Experiments carried out in these scenarios with a real vehicle confirm the effectiveness of this technique.

Keywords: Mobile robots; Sensor-based motion control; Architectures for navigation; Obstacle avoidance

1. Introduction

More and more research and industrial interests in robotics are focused on improving the degree of system autonomy. Robots are being developed that operate under a wide variety of conditions, with an emphasis on long work periods without human intervention. These include specific tasks that are tedious or involve dangerous or hostile surroundings. Autonomous navigation systems are used in applications like service robots, surveillance, or exploration, where the vehicle moves and carries out the main task at the same time.

One of the key aspects of these robots is mobility, since it is the basis on which to incorporate more subsystems with different functionalities. However, the performance of the motion system strongly affects task performance. Special problems arise in applications that may lead to fatal consequences (e.g., robots that transport dangerous materials).

Mobility is closely related with the nature of the scenario. In many applications, the environment cannot be specified with an a priori map and can be dynamic.

* Corresponding author. Tel.: +34 976 762 350; fax: +34 976 761 914.
E-mail address: jminguez@unizar.es (J. Minguez).

0921-8890/$ – see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2005.06.001

Under these circumstances, sensors collect information about the environment and adapt robot motions to any new contingency or event. Sensor-based motion systems appear to be the natural choice; however, most of them cannot carry out robust and trustworthy navigation in very complicated environments. These usually involve spaces with little room to maneuver, highly dynamic environments or that lead to trap situations. An example is an office (Fig. 1a), where the robot moves among the chairs, tables and shelves (all with unknown positions) and humans (who make the scenario highly dynamic). Maneuvering is also extremely difficult in certain circumstances when there is little room to maneuver (Fig. 1b). Here we address the motion control of a vehicle under these work conditions.

We designed and tested a sensor-based system with three modules that work together to carry out the motion task. Our design is constructed over some requirements that we identified to drive vehicles in troublesome scenarios. The system differs from previous works in the choice and implementation of the modules and in the architecture of integration. Key contributions include the functional and computational aspects of module design and integration, and the experimental validation with an emphasis on highly dynamic environments that are unknown and indoors. This paper includes a strong experimental component, and all the results are from tests using a real vehicle (see Appendix A for further details of the robot).

The sensor-based motion control subsystem was developed to move the vehicle to the desired positions without collisions. This functionality is only a subset of the complete mobility problem. Other aspects involve perception, planning, modeling and control. They will not be addressed in this manuscript, but are essential to construct a complete autonomous motion system. Related works include motion planning [1] location and map building [2–5] and supervision [6,7]. However, in this paper, we also discuss the integration of our motion subsystem in a complete motion architecture (high level planning, localization, motion control and supervision), and we present, discuss and compare the real results qualitatively and quantitatively with other methods.

The work is organized as follows: first we discuss related work (Section 2), and the basic requirements to design a sensor-based motion control system (Section 3). Next, we provide an overview of the system (Section 4), the modules and their integration (Section 5). Finally, we present the experimental results (Section 6) and the integration in a complete motion system. We compare out subsystem with other methods in Section 7 and present the conclusions in Section 8.

(a) (b)

Fig. 1. (a) A typical indoor environment with tables, chairs and closets whose positions cannot be specified a priori in a map. There are also people working in the office, which makes the surroundings highly dynamic. (b) Navigation in this type of scenario must overcome situations with difficult maneuverability (for example when the robot passes through a door with 5 cm maneuvering room on either side).

2. Related work

The topic of motion in evolving environments includes issues such as knowledge representation (model construction), global deliberation and reactivity. Navigational planning without considering execution is restricted to a small domain of the problem. This is because it becomes difficult to consider all contingencies and it is unrealistic to formulate plans that do not reflect a changing environment. On the other hand, reactive systems limit their scope of application to perception–action, gaining flexibility and robustness of motions. The overall problem cannot be solved by these systems individually, since they need more extensive models of knowledge and some way to incorporate memory. The interest is focused on synthesizing a control mode that incorporates both methodologies and not on extending both worlds separately [8]. Hybrid systems attempt to combine both paradigms by including the best artificial intelligence to represent and use the knowledge, with the best reactivity, robustness, adaptation and flexibility. Basically, these schemes should combine a planner (deliberation) and a reactor (execution). The main differences among them are: (i) the interaction between the planner and the reactor (i.e. how the reactive method uses the information available of the planner); and (ii) the techniques used to implement each module.

One way to specify the interaction between deliberation and reaction is to consider planning as a component that fixes the composition between different behaviors during execution [9]. For example, these behaviors can be implemented with potential fields [10], so that modifying their weights changes the overall behavior of the system. Another possibility is to use the planning to advise the reactive control [11], or as a system that adapts parameters of the reactive component based on the evolution of the surroundings [12]. In both cases, planning plays a tactical role while the reactor has the execution degree of freedom. The advantage of this planner–reactor configuration is that it combines the deliberative component (the plan is always available in execution and improves with the time [13]) and the reactive component (executor of the motion). A perspective on hybrid architectures is given in [14]. Focusing on the motion context, one common strategy is to compute a path and use its course to direct the reactive module [15–18]. These techniques require large computational resources, but use complete planners (they always find a path if it exists). Other techniques compute a path that is deformed in execution based on the evolution of the environment (in the workspace [19], or in the configuration space [20]). Nevertheless, these methods need to replan when the path is invalidated or when it moves far away from the initial configuration due to unexpected obstacles. Alternatively, [21] presents a strategy to create trees of paths obtained by executing the reactive algorithm some steps ahead of the execution. This system obtains good results in platforms with low computational resources, but does not assure a convergence to the target. Another possibility is to compute a channel of free space that contains sets of ways, leaving the choice up to the execution [22].

These issues are closely linked to the choice and implementation of techniques for each module. All the previous strategies use the planner to obtain a way to guide the reactive control. The planner is usually an efficient numerical technique executed in real time [23,24]. Another key module is the reactive method itself. Some techniques are based on potential methods [10,25,26], but they have limitations [27]. Other techniques compute a set of intermediate motion commands to choose the next one.

The commands are directions of motion [28,21], sets of speeds [29,30], or sets of trajectories [31,32]. However, these reactive methods are of limited use when the scenario makes it difficult to maneuver the vehicle (usually with high obstacle density) [33], identified some consequences like local trap situations, irregular and oscillating motions, or the impossibility of driving the vehicle towards areas with a high obstacle density or far away from the goal direction. These behaviors acquire greater relevance in the development of robust applications to navigate without depending on the difficulty of the environment. The navigation systems that rely on reactive methods also inherit these drawbacks, limiting their use in realistic applications. Finally, in this type of architecture, models are usually constructed to serve as the basis for the planner and provide short-time memory for the reactive behavior. In indoor environments, the occupancy grids are usually used with ultrasound [15,34,35] and laser [16–18].

In this paper, we describe a hybrid system with a synchronous and heterogeneous planner–reactor configuration, where both modules use the model constructed in execution to carry out the motion task. Our main

contribution is the choice and design of the modules and their integration. As a result, the new system can move vehicles in very difficult environments, where other systems may find difficulties.

3. Requirements of the sensor-based motion control system

The basic version of these systems moves the vehicle among positions without collision. The operation is governed by a perception–action process repeated at a high frequency. Sensors gather information from the environment (obstacles) and the robot, which is processed to compute the motion. The vehicle executes the motion and the process restarts. The result is an on-line sequence of motions that drive the vehicle to the destination without collisions. In this Section, we specify some general requirements for these systems:

(1) *Integration of information*: the successive sensorial measures must be stored or integrated to construct a representation of the environment. This is necessary to avoid obstacles that are not perceived at the moment (visibility constraints of the sensor), and to increase the reach of the information used (increasing the spatial dominion). In addition, changes in dynamic scenarios must be rapidly reflected in the model. Otherwise, the robot will avoid areas perceived as free space or will not avoid perceived obstacles (since in both cases, the information could not be still reflected in the model).
(2) *Avoidance of trap situations and cyclical behaviors*: the system has to be equipped with a strategy to avoid these situations. Many different configurations of obstacles can trap the robot (the most typical being U-shape obstacles or end-zones) or create cyclical motion (e.g., symmetrical distributions of obstacles). The robot will never reach the final location under these circumstances.
(3) *Generation of robust motion*: the final motion must be computed by a robust reactive method. This algorithm should be able to avoid collisions independently of the inherent difficulty of the environment. As a rule, the most problematic scenarios have a large obstacle density where maneuvering is difficult and critical.
(4) *Integration of functionalities*: all functionalities must be integrated within an architecture for specification, coordination and failure detection and recovery. The integration must close the perception–action cycle at a high frequency, since it fixes the reactivity of the system to unforeseeable changes (detected by the sensors). Furthermore, this favors the portability between different platforms and sensors (the interactions between modules are not designed from scratch when the modules are replaced).

Here we attempt to make our system fulfil these requirements.

4. Overview of the system

We provide a general view of our hybrid system which has three modules and one architecture for supervision. The functionalities of the modules are model construction, planning and reactive navigation:

- *Modeling module*: it constructs a model of the environment (integrating the sensorial information). We used a binary occupancy grid whose cells are updated whenever a new sensorial measurement is available. The grid has a limited size (representing a fixed portion of the space), and whose location is continuously recomputed to include robot location.
- *Planning module*: it extracts the connectivity of free space (used to avoid the trap situations and cyclical motions). We used the *Navigation Function 1* [23] (NF1 in short) which is based on the front-wave expansion by free space. The planner is free of potential minima, can work on a grid (existing representation), and can be efficiently executed in real time (at least once in each perception cycle).
- *Reactive navigation module*: it computes the collision-free motion. The method used is the Nearness Diagram Navigation [33] (ND method in short), which is based on selecting a navigational situation at every moment and applying the associated motion law. This method is very efficient and robust in environments with little space to maneuver.
- *Architecture of integration*: it integrates the modules following a synchronous planner–reactor configuration [12], where both parts use the model constructed in execution time. The synchrony of the modules avoids problems of time-inconsistencies.

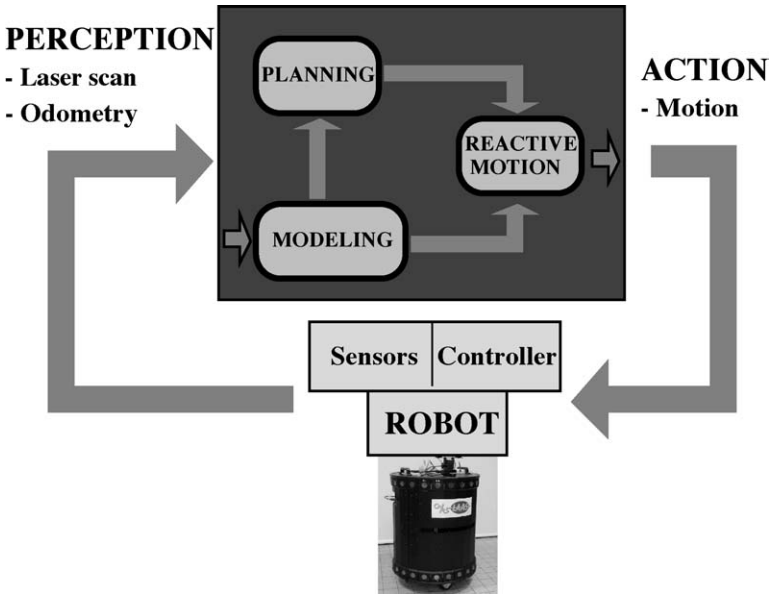

Fig. 2. It shows the perception–action cycle of the sensor-based motion control system. Perceptions are measures of the obstacles and the odometry of the robot, and the action computes the collision-free motion commands. The system has three modules that cooperate to carry out this task: the model construction, the tactical planner and the reactive navigation method.

The system works as follows (Fig. 2). Given a laser scan and the odometry of the vehicle, the model builder incorporates this information into the existing model. Next, the information about obstacles and free space in the grid is used by the planning module to compute the course to the destination. Finally, the reactive module uses the information about obstacles in the grid and information of the tactical planner to generate motion (to drive the vehicle towards the destination without collisions). The motion is executed by the vehicle controller and the process restarts. It is important to stress that the three modules work synchronously within the cycle of perception–action. This reinforces the importance of the choice and computational aspects of the techniques used in each module. Next, we describe the design of the modules and the integration architecture.

5. Design and integration of the functionalities

Here, we present the design of the modules in the system. We discuss the modeling module (Subsection 5.1), the planning module (Subsection 5.2), the navigation method (Subsection 5.3) and the architecture of integration (Subsection 5.4).

5.1. Model builder module

This module integrates the successive sensorial measures to create a local model of the environment. We chose a binary occupancy grid whose cells are *occupied*, *free* or *unknown*.[1] We did not use traversability or uncertainty factors, since the laser has a high precision in indoor environments. The grid has a fixed size that represents a limited part of the workspace (large enough to represent the portion of space necessary to solve the navigation) and whose position is recomputed to maintain the robot in its central zone (the obstacles that surround the robot are always available even if they are not visible from the current location).

The design of this module includes: (i) the integration of the scans in the model; and (ii) the update of the grid position to maintain the robot-centered. To integrate a laser scan in the model, we consider that a scan is a cloud of points where: (a) in each point, there is an obstacle (cell updated occupied); and (b) the line that joins each obstacle point and the sensor is free space (cells updated free), see Fig. 3a. We implemented this

[1] To all purposes, the unknown cells are assumed to be free (binary grid); however, we use them to clarify the figures.

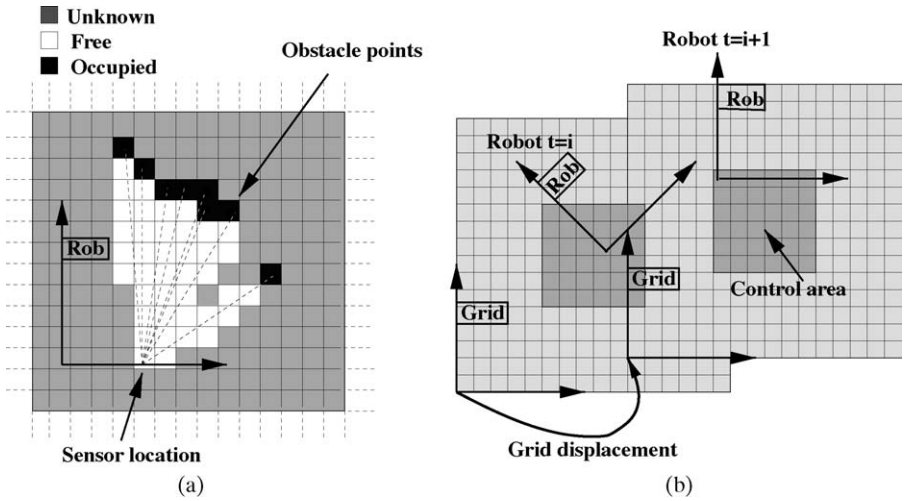

Fig. 3. It shows how the laser measures are integrated in the grid and how the grid position is recomputed for the robot to remain in the central zone. (a) In a laser scan, the cells are occupied for obstacle points, and the cells in the lines that join the obstacle points and the sensor are updated as free. (b) At time $t = i$, the robot is within the control zone of grid. Next, at $t = i + 1$, the robot has left the control zone and the grid location is recomputed (in multiples of cell sizes and without rotation) so the new position the robot is within this zone.

procedure using the *Bresenham* algorithm [36], which is optimal in the number of cells visited to project a line in a grid. This algorithm considerably reduces the integration time of a sensorial measurement.

Secondly, to keep the robot in the central zone of the grid, we define an area called *control zone*. When the robot leaves this area, the new position of the grid is recomputed to center the robot within this zone. The robot is always in the central zone of the grid, whose position does not change until the robot leaves. The recomputation of the grid position is always made in multiples of the cell dimension and rotation is not allowed (this strategy reduces the dissemination of false information of the obstacles in the cells, which is an important source of error). In addition, this strategy can be efficiently implemented with memory shifts to reduce the computation time.

Fig. 4 shows an experiment when two people move in front of the robot.

A typical strategy consists in integrating the scans as a short-time memory. If we use this integration (Fig. 4a–c), the result does not reflect the change rapidly, since the people leave a temporary shadow (obstacles that do not exist, Fig. 4d). For the same experiment, we used this module with a grid of 200×200 cells and 0.05 m each cell. The resulting grid (Fig. 4e) rapidly reflects the change in dynamic environments because the whole area covered by the last scan is updated (the obstacles and the free space). Furthermore, the obstacles not visible from the current location remain in the grid and can be used for the avoidance task. Other important issues include: (i) the last laser scan integrated in the grid does not have odometry errors with respect to the present position (only the cells not updated with this scan accumulate these errors); and (ii) the spurious measures are eliminated from the grid as new measures are added. These advantages justify the usage of binary grids for navigation purposes with regard to probabilistic approaches (that would require to accumulate evidence of both free and occupied space to be reflected in the model).

With respect to the functional and computational aspects of this module, the grid represents a portion of the environment of 10 m $\times$ 10 m around the robot (large enough to include the goal location), and with a cell size of 0.05 m (0.01 m larger than the sensor nominal error). With these settings, the module takes about 0.08 s, thus it can work within the laser cycle (0.20 s).

In summary, this module integrates the sensor data so that the spatial domain of the information is increased as the robot progresses. Furthermore, the model rapidly represents changes in evolving scenarios by updating the whole space covered by the last perception. For these reasons, this module complies with

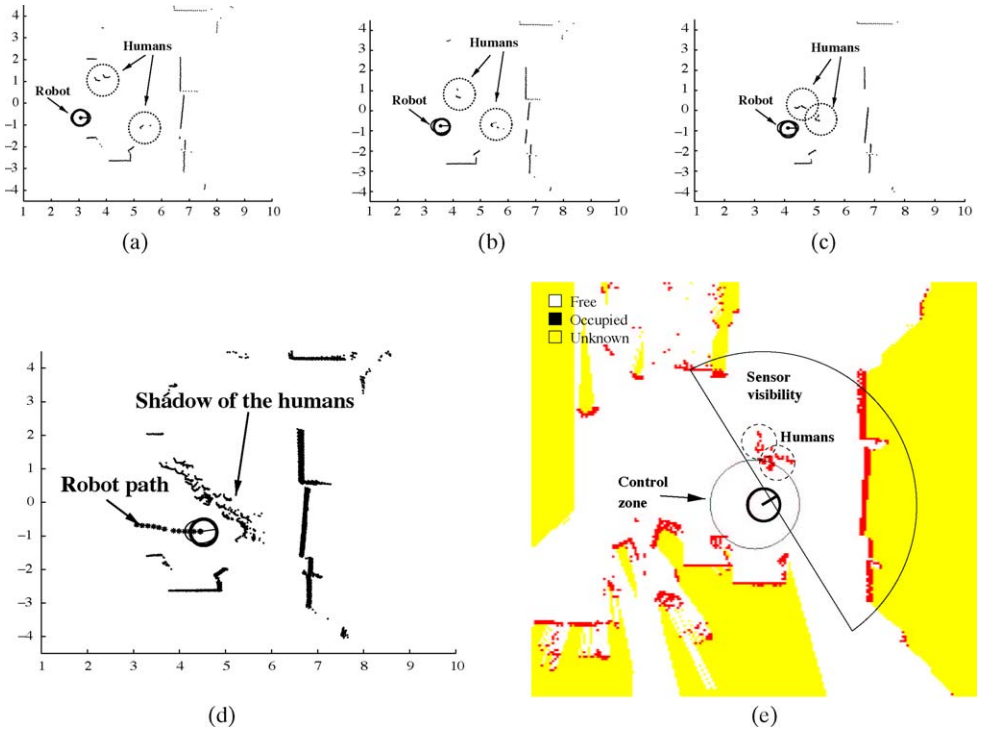

Fig. 4. It shows the calculation of the grid when two people walk in front of the robot. (a–c) show the robot location and the scan at $t = 0$, 1 and 2 s (3 of 10 measures). (d) The result of the 10 scans integrated as a short-time memory, where some obstacles appear where they do not exist (this representation is not a rapid reflection of the evolution of the environment), and (e) the grid obtained by this module using the same scans. The dynamism of the environment is rapidly represented because the scans are integrated when available, updating the obstacle and the free space in the model.

the *information integration* requirement in Section 3. Next, we address the planning module.

5.2. Planning module

This module uses a motion planner to obtain tactical information to avoid trap situations and cyclical motions (not to control the vehicle). The planner constructs a navigation function (NF1) over the grid of the previous module, and then computes a path to the destination using a steepest descendent strategy. We selected this planner because the navigation function does not have potential minima (if a path exists, it is found), and it is a numerical function that works efficiently on a grid (i.e., the existing representation).

The planner uses a two step process to compute a path from the robot location to the destination (Fig. 5). First, the navigation function is computed. Each obstacle is enlarged with the robot radius, and then the NF1 is constructed by propagating a wave from the destination over the free cells (each cell is labeled with the distance measured in number of cells until the goal is reached). Secondly, a path is calculated using a gradient search strategy on the potential. However, this path is optimal in the metric defined over the grid. Thus, in an iterative process, the path is stretched to make it optimal in configuration space. This avoids the border effects of the NF1 and paths that point out or graze the obstacles (Fig. 5a).

Two types of information are obtained from the path. First, the possibility of reaching the goal from the present position (the planner finds the path if it exists). Second, the *instantaneous motion direction* (main course of the first part of the path, see Figs. 5a and 6). This direction will be used to advise about the motion, but not as a path to execute (since the motion generation will be handled by the reactive module). That is, the instantaneous direction is the tactical

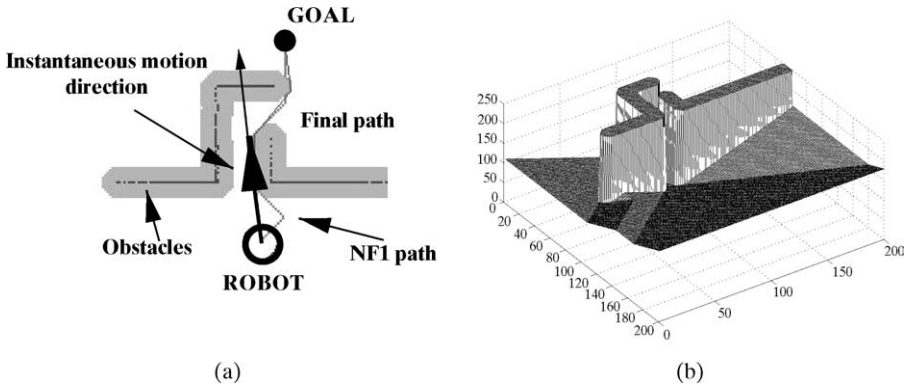

Fig. 5. It shows the operation of the planner. First, the obstacles are enlarged with robot radius (a). Next, the NF1 is computed by propagating a wave from the final position, where each cell is labeled with the accumulated cost of the wave (b). On this function, a path is computed with a gradient search strategy, NF1 path in (a). Finally, the path is stretched (to be optimized) in order to obtain the instantaneous direction of motion from the first part of the path.

information computed by the planner to avoid trap situations. This is graphically illustrated in Fig. 6, where the robot was in front of a U-shape obstacle where it could be trapped. Nevertheless, when we compute the path and extract the instantaneous direction, it aims towards the exit. Using this direction as course avoids the trap situation.

The computational aspect of this module is bounded with the size of grid. With a size of 200 × 200 cells, the module takes about 0.08 s (worst case) and thus can be used within the sensorial cycle after the modeling module. With regard to the integration, the computation time of the navigation function is proportional to the square of the number of cells. Thus, increasing the size of the grid would increase the computation time (penalizing the inclusion in the sensorial cycle). On the other hand, reducing the size of the grid or its resolution diminishes the spatial domain or the precision. That is, the size of the model (portion of the space and precision) creates a commitment with the computation time of the planner, and they both have to be balanced in the real implementation.

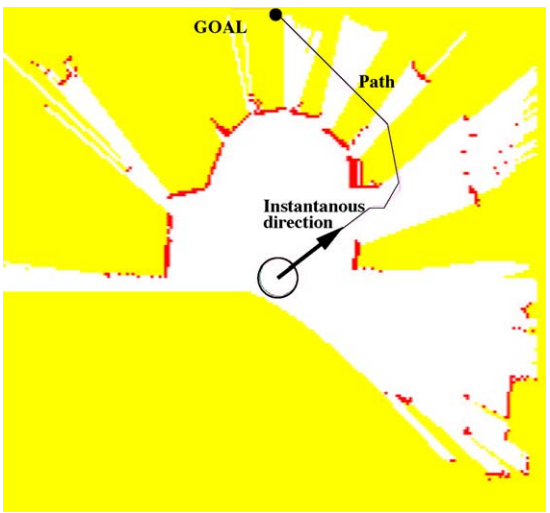

Fig. 6. It shows an experiment where the planning module computes a path to the destination over the grid. The instantaneous direction of motion is extracted from the main direction of the path. This direction aims towards the exit of the U-shape obstacle, which is used as a course to avoid the trap situation.

It should be noted that the instantaneous motion direction contains tactical information to *avoid cyclical motions and trap situations* (requirement settle in Section 3). In the following Sections, we analyze how the system uses this information to solve these situations. Next, we describe the module that computes the motion.

5.3. Reactive navigation module

The reactive navigation module computes the collision-free motion to drive the vehicle towards the final location. The ND method employs a “divide and conquer” strategy to simplify the navigation by identifying situations and applying the corresponding motion laws. It can solve complex navigation cases for robust motion in troublesome scenarios.

The ND method uses a methodology to design behaviors called the situated-activity paradigm (see [14]). First, a set of situations is defined to represent the navigational problem and mode of conduct (actions). Here, the situations represent all the cases between robot positions, obstacles and the goal (navigational situations). In addition, for each of these cases, a motion law (action) is associated. During the execution phase, we use information about the obstacles, the robot and the destination to identify one of these situations. Then the corresponding action is applied that computes the motion. The motion is executed by the robot and the process restarts (Fig. 7). Next, we outline the situations and actions.

We used a binary decision tree to define the situations (Fig. 7). The inputs are the obstacles, the robot and goal locations. The output is a situation that is chosen by selecting one branch after applying criteria that depend on the inputs and on their relations. The relations are obtained from a *security zone* around the robot boundaries and from an entity that we call the *free zone*. Using the security zone, we check whether there are risky obstacles, and with the free zone, we select a suitable area of motion (Fig. 8). The set of situations cover all the possible cases among robot and goal locations and obstacle distributions, and only one is identified in each case (the set of situations is complete and exclusive).

Each situation has an associated action that computes a motion command (**v**, **w**) to adapt the behavior to the navigational situation. To compute the direction of motion, we use geometric information about the obstacle distribution and the free zone. The module of the velocity depends on the security context, which is maximum when there are no obstacles within the security distance (safe situation). This module is reduced linearly with the distance to the closest obstacle. We establish a rotation speed **w** to align the robot with the instantaneous direction of motion.

In summary, this reactive method identifies a navigational situation and applies a motion law to obtain the motion commands. The commands are sent to the vehicle and the process restarts. The advantage of this method is that it can drive vehicles in very dense, cluttered and complex scenarios. As Minguez and Montano [33] point out, this method: (i) avoids local trap situations due to the perceived environment structure (i.e. U-shape obstacles and two very close obstacles); (ii) computes stable and oscillation free motion; (iii)

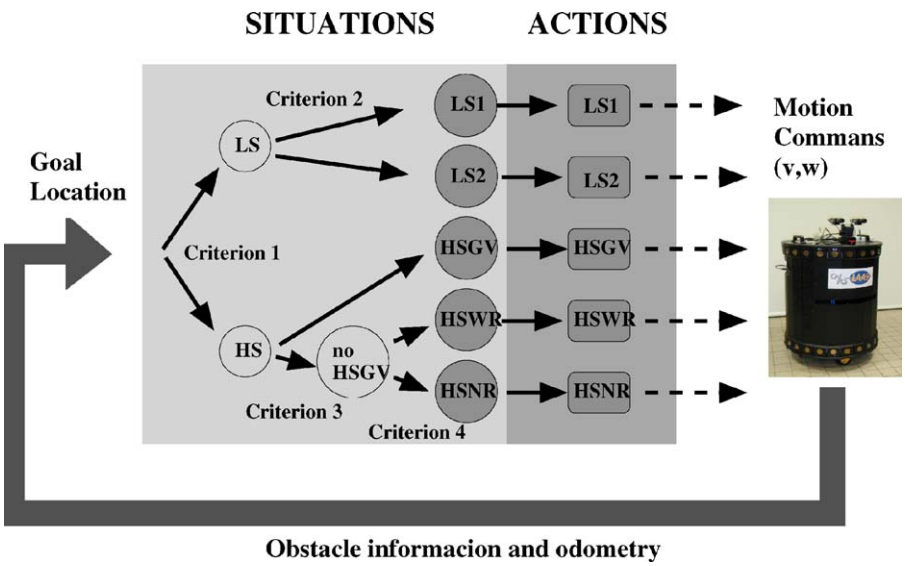

Fig. 7. It shows the ND method design, which is made up of a set of navigation situations and their corresponding actions (motion laws). During execution, the sensory information is used to choose a navigational situation (using a binary decision tree) and the corresponding action is executed to compute the motion. LS situations correspond to the case where there are obstacles within the security zone (LS1 at one side, and LS2 at both). HS situations are when there are not obstacles within the security zone. If the goal is in the motion area, HSGV, or out of this area but wide, HSWV, or narrow, HSNV.

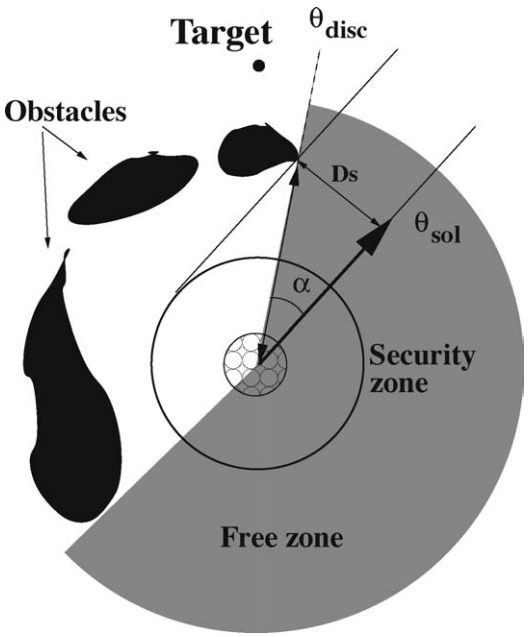

Fig. 8. It shows the computation of the motion solution with the ND in one situation. To identify the situation, the process is to check some criteria. For example in the figure, there are not obstacles closer that the security distance D_s. Next, the target is not within the motion area. Third, the motion area is wide. With these three criteria, we identify the current situation, HSWV. In this situation, the associated action computes the command $v_i = (\theta_{sol}, v_{sol})$, where v_{sol} is the maximum velocity and θ_{sol} is computed as a deviation α from the limit direction of the motion area.

selects directions of motion towards the obstacles; (iv) exhibits a high goal in-sensitivity (i.e. choosing directions of motion far away from the goal direction); and (v) selects regions of motion with a robust navigable criterion.

Fig. 9 shows an experiment carried out with this reactive module. The vehicle reached the final position (which is the only information provided a priori) at the end of the passage (Fig. 9a) without collisions. Notice how in some areas, it was very difficult to move (Fig. 9b) and motion directions towards the obstacles were required at every moment to solve this navigation. Furthermore, the trap situations or oscillating motions due to motion among very near obstacles were avoided (see the robot trajectory in Fig. 9b).

Finally, the module complies with the requirement of *generation of robust motion* (Section 3). This is because this method is able to drive vehicles in troublesome environments (dense, complex and cluttered), and the method works at a high rate (approximately at 0.04 s) for quick reactions in dynamic surroundings.

5.4. The architecture of integration

The architecture integrates the modules by considering the limitations and restrictions imposed by the physical (sensors and actuators) and logical parts (computers) of the robot. The architecture has a synchronous planner–reactor configuration, where both parts use the model constructed in execution time (Fig. 10). The functionalities of the modules are the model construc-

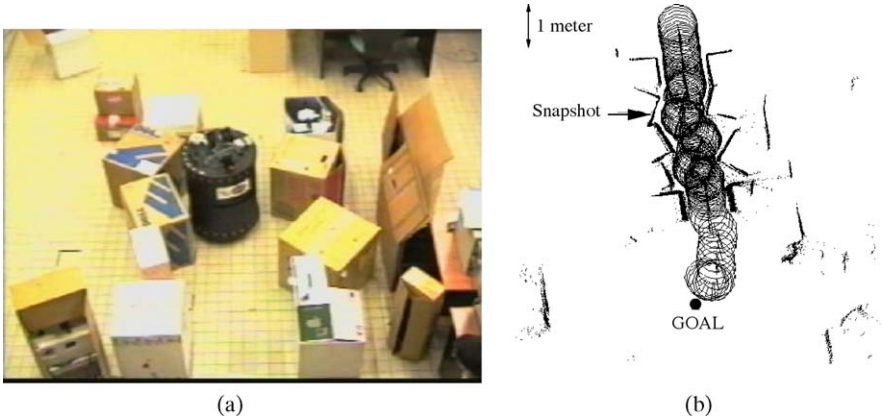

Fig. 9. This experiment shows how the ND method can drive the vehicle in difficult environments. (a) A snapshot of the experiment that illustrates the robot and the obstacles. (b) All the laser points and the trajectory executed by the robot.

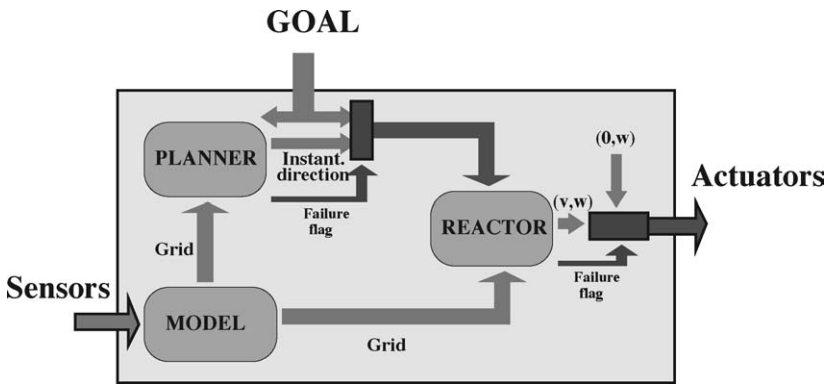

Fig. 10. It shows the architecture of integration. The modules work together synchronously and share some data. Furthermore, depending on some circumstances, the modules fail, and exceptions are launched to manage the situation in order to close the motion control loop.

tion, the computation of the tactical motion direction (to guide the reactive method), and the motion command generation.

In some situations, the modules produce failures that are managed by the architecture:

- Exception in the planning module: sometimes the planner does not find a solution, either because it does not exist (for example when the goal falls on an obstacle) or because a time out is launched when the module takes too long. A path may not exist in unknown environments when goals are placed for exploration and one falls on an obstacle, in dynamic surroundings when a dynamic obstacle moves or stops on the goal, in static environments when the goal is displaced due to robot drift, or when the goal or the robot are surrounded by obstacles[2].
- Exception in the reactive module: the robot is completely surrounded by obstacles when there are no areas of motion (internal piece of information in the ND method), and it cannot progress.

In both cases, we set flags to carry out strategies that can close the control loop (to avoid dead states). In the first case, the reactive module uses the goal location directly instead of the information from the planner. In the second case, the robot stops and turns on itself (this behavior updates the model in all the directions looking for a new path).

The modules are executed following the modeller–planner–reactor sequence (Fig. 11), dictated by the flow of data between modules. This flow is unidirectional, from the modeling module towards the planner and reactive module (with a bandwidth of 160 kbytes/s) and from the planner towards the reactive module (with a bandwidth of 8 bytes/s). The modules assure that their time constraints are in synchrony with the sensor rate 0.20 s. This is important to avoid inconsistencies in time that would arise using asynchronous strategies (the model is used for local planning and obstacle avoidance and must be consistent in time with both modules). We assigned

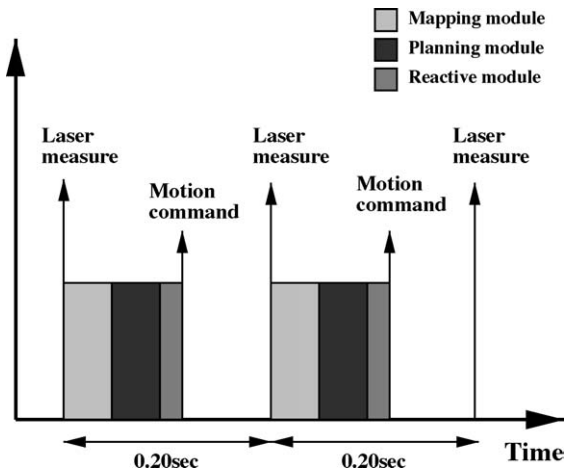

Fig. 11. It depicts the time diagram of the execution of the modules, which work synchronously within the sensor cycle. That is once a new laser measurement is available, the three modules are executed sequentially before the new measure event.

[2] Although these situations could be avoided by replacing the goal position, this is not the role of the navigation system. The consequences of this decision could affect the normal development of the robotic task in general.

time outs of 0.08, 0.08 and 0.04 s to each module to keep the motion control loop closed (the maximum execution time is 0.20 s).

To conclude, the modular structure of the system allows to replace the different modules easily, since the computational aspects and interfaces among the functionalities are clearly specified (requirement of *modular integration* specified in Section 3). We have considered how each module complies with the proposed requirements. Next, we show how they also comply when integrated.

6. Testing the sensor-based system

In this Section, we discuss some representative experiments carried out with the sensor-based navigation system on the real vehicle (see the Appendix A for further details). All environments were completely unknown, unstructured and dynamic with an a priori unforeseeable behavior. The experiments demonstrate the system working in: (i) very dense, complex and cluttered scenarios; (ii) avoiding trap situations, and (iii) in a highly dynamic environment. In addition, in

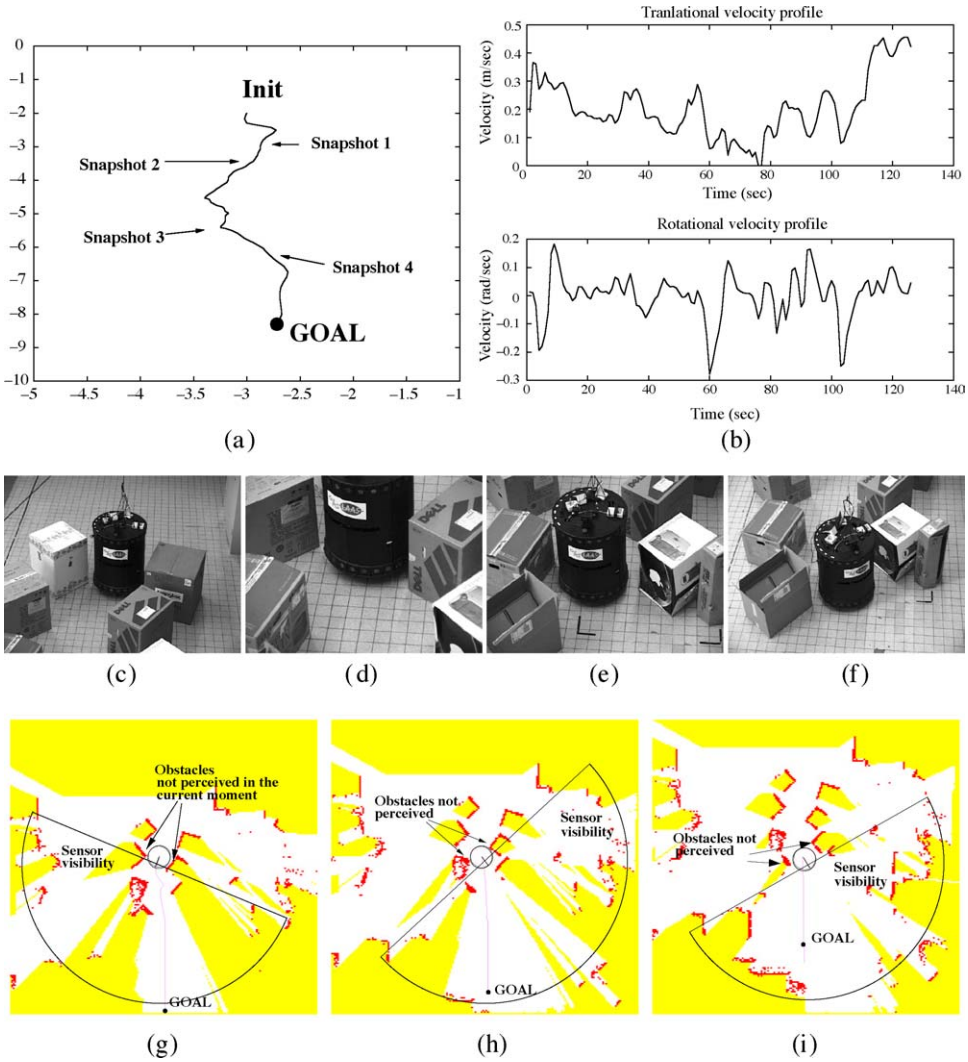

Fig. 12. Experiment 1: the robot must cross a narrow corridor with boxes to reach the goal. (a) Trajectory executed and (b) motion commands. (c–f) Some snapshots of the experiment (g–i) and the corresponding grid and robot, where (d) matches with (g), (e) with (h), and (f) with (i).

the last subsection, we explain how this sensor-based system is integrated in a complete motion system (with high level planning, global location and motion execution). Many tests ranging from short-term to long-term missions validate this integration.

6.1. Experiment 1: motion in very dense, complex and cluttered scenarios

This experiment highlights the framework in a difficult scenario, where the robot navigated along a very narrow passage with randomly distributed boxes (Fig. 12). The difficult part was inside the corridor where there was little space to maneuver (in some parts, less than 10 cm on either side of the robot, Fig. 12d. The average speed was 0.204 m/s, and the robot reached the final location in 127 s.

Key aspects of this experiment were the individual performance and the cooperation between the modeling and reactive modules. The laser scans in the modeling module were rapidly integrated in the grid. Thus, the reactive method avoided the new obstacles as soon as they were perceived. This reactivity is essential to move in dense environments (since delays in motion computation would lead to collisions). Second, last sensorial measures remained in the grid and were used by the reactive module for the avoidance task. This was important since sometimes the sensor did not perceive the closest obstacles due to visibility constraints. However, since they were perceived some time before, they remained in the grid and were avoided (Fig. 12d–g, e–h and f–i).

This experiment was particularly difficult from the point of view of motion generation due to the narrow space, because many of the existing techniques have intrinsic limitations that would penalize to compute motion under these circumstances (see [33] for a discussion on this topic). However, the ND method drove the vehicle free of collisions even among very close obstacles. This motion was smooth, free of oscillations (see the path executed in Fig. 12a and the velocity profiles in Fig. 12b) and free of any traps due to the obstacle density. Important requirements complied with here are the *generation of robust motion* and *integration of the information*.

6.2. Experiment 2: motion avoiding trap situations and cyclic behaviors

In this experiment, the navigation system solved several trap situations that occurred because the structure of the surroundings was modified. Fig. 13c–g shows the robot moving along a passage to reach the goal. When the vehicle was about to leave the passage, a human cleared a box in the initial part of the passage (white box in the Fig. 13c) opening a passage on the right-hand side. Next, the human closed the end of the passage (Fig. 13d) producing a trap situation in a U-shape obstacle. Rapidly, the next laser scan perceived the change and was integrated in the grid. The planning module computed the course towards the exit (Fig. 13h). The reactive method followed this tactical information, and the robot turned backwards and moved towards to exit. Nevertheless, while progressing, it perceived the new passage on the right-hand side and headed in this direction. Then, the passage was closed again, leading to another trap situation (Fig. 13e–i). Following the previous process, the robot left this passage and returned backwards (Fig. 13f), until leaving the passage and reaching the final position. The experiment took 200 s and the average speed was 0.233 m/s.

Cooperation among the three modules avoids cyclical motion and trap situations. First, we did not encounter obstacle configurations that produce trap situations (when a path exists within grid). This is because the direction of motion computed by the planner contains the tactical information necessary to avoid these situations (notice that the motion is computed by the reactive module). Secondly, the symmetric environments do not produce cyclical motions since this tactical direction discriminates between the possible motion zones. To conclude, the system moved the vehicle in an environment where the conclusions are similar to Experiment 1 regarding the *generation of robust motion* and the *integration of information*. *Cyclical motions and the trap situations* were also avoided.

6.3. Experiment 3: motion in a highly dynamic scenario

In this experiment, the system drove the vehicle in a populated and continuously changing scenario. Here, the inherent difficulty of the previous environments

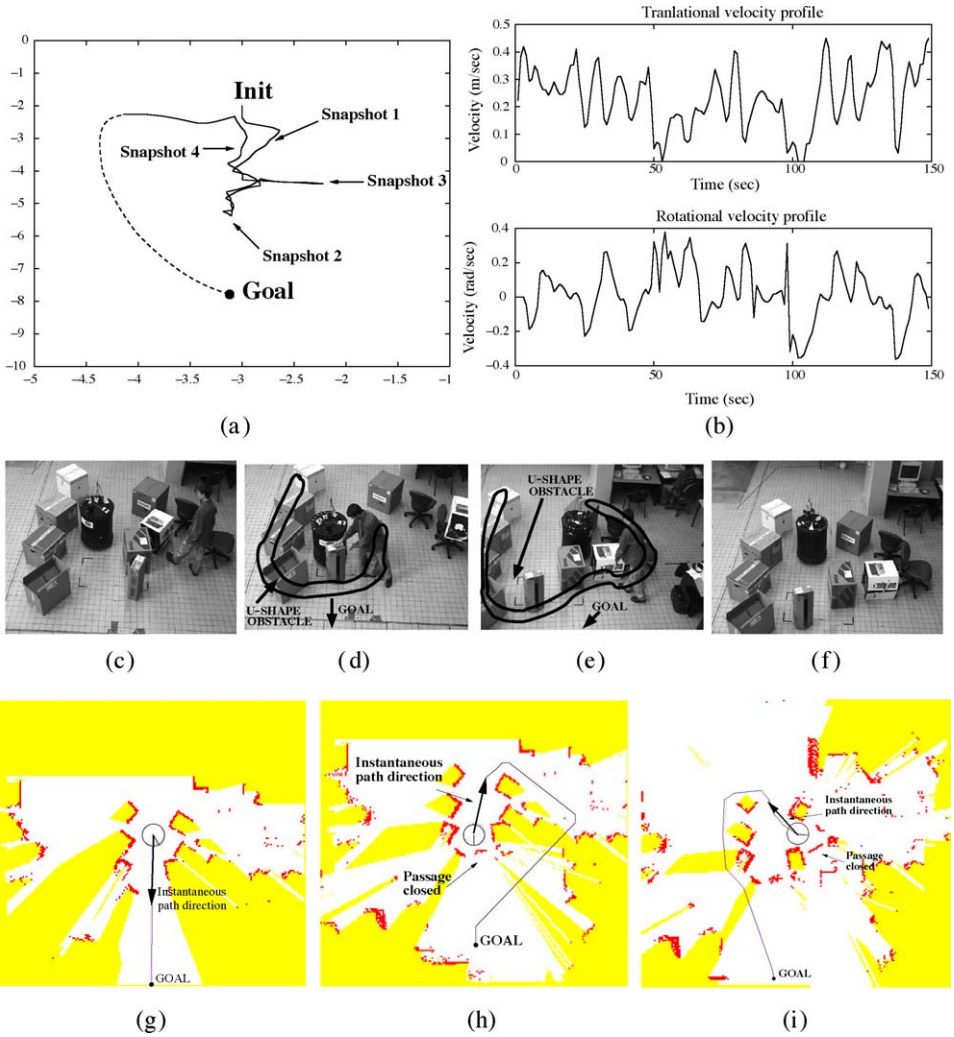

Fig. 13. Experiment 2: in this experiment, the robot avoided three successive trap situations dynamically created by a human. (a) Trajectory executed y (b) motion commands. (c–f) Some snapshots of the motion, and (g–i) the corresponding grid and robot, where (c) matches with (g), (d) with (h), and (e) with (i).

persists (obstacle density and trap situations), while adding the difficulty to model and maneuver among the dynamic obstacles. Fig. 14a shows the initial state, where the robot had to cross a large room to reach the destination. During the first part of the experiment, humans moved in front of the robot (Fig. 14a–c) to hinder the motion. Here, the model constructor module was important to successfully integrate the information. As a consequence, the system identified the areas of motion and performed the avoidance task. Later, the scenario evolved creating a U-shape obstacle that produced a trap situation (Fig. 14d and e). Rapidly, the planner computed the tactical information that was used by the reactive module to drive the robot out (Fig. 14f and g). Once outside, the system continued to react to the humans that disturbed progress towards the goal. Finally, the robot reached the goal location (Fig. 14h). The average speed was 0.196 m/s and the run time was 170 s.

This example illustrates the importance of the model constructor module in dynamic surroundings, since it provides a base for the other two modules. If the

Fig. 14. Experiment 3: the robot moved in a highly dynamic and constantly evolving environment, where trap situations arise repeatedly. Some snapshots of the motion (a–h).

model did not work correctly (mainly in the first part of the experiment), a cloud of insurmountable obstacles would appear in front of the robot, similar to Fig. 4d. In addition, the reactive module reacted rapidly to the evolution of the surroundings and, guided by the planning module, avoided all the traps. This example illustrates how the system can drive the robot towards locations under realistic conditions.

6.4. Integration in an autonomous motion system

In this section, we describe the integration of the system proposed in this work in a real autonomous robotic system. The integration described next, the test in the real robot (same vehicle used in this paper) and the pertinent conclusions are the result of the work of Bennoit Morisset at LAAS-CNRS (France) and it is described in detail in [37] (see also [38]). The goal of his study is to build a autonomous system relying on several complementary behaviors called modalities. Each modality is generated as a HTN from a pool of sensory-motor functions dedicated to localization, obstacle avoidance and path planning. The complete system has been extensively tested to extract conclusions about performance, and in particular about the motion control subsystem presented in this paper. We understand that this is an objective evaluation of this research.

We describe next the sensory-motor functions and their particular instances:

(1) Global motion planning: computation of a geometrical path free of collisions between two locations taking into account the shape and kinematic constraints of the vehicle, given a priori model of the scenario. The planner used is based on Voronoi diagrams and is fully described in [39].
(2) Localization: construction of a model of the scenario and computation of the vehicle location within the model. There are two instances of this functionality, a SLAM one and another based on localization using visual landmarks.
   (a) SLAM: in an initial step, this module uses a simultaneous localization and map building method to incrementally construct a segment-based model of the scenario [40,41]. During the execution step, the system uses the laser information to construct segments, that are matched with the model in order to obtain the current vehicle location. This module works between 15 and 100 ms and has a precision of 1 cm in translation and 1° in rotation. When integrated, it is used at a frequency of 2.5 Hz (one each two laser measurements).
   (b) LPV: in an initial step, this module uses a monocular vision system to detect rectangular posters (landmarks) that are learnt in a supervised way. In execution, the landmarks are detected with the vision system. Then, the relative location of the vehicle is computed, and knowing the location of the landmarks the vehicle absolute position is deduced [42,43].
(3) Motion control: computation of the collision-free motion towards a given target location.

(a) EB: the Elastic Bands is a method that initially assumes the existence of a geometric path to the target location (computed by a planner). The path is assimilated with a band, subjected to two types of forces: an inner contraction force and an external force. The inner force simulates the tension of a strip and maintains the stress. The external force is exerted by the obstacles and moves the band far from them. During the execution, the new obstacles produce forces that remove the band far from them, guarantying their avoidance [20]. The implementation used here is described in [44,45].
(b) mpND: the mapping-planning ND is the motion system proposed in this paper.
(c) mND: the mapping ND is the motion system proposed in this paper but without the planning component.

In short, the system works as follows. Firstly, the motion planning method computes a path using the a priori model, which is converted next in a sequence of subgoals. During the execution, the motion control system computes the commands to avoid the unexpected obstacles gathered by the sensors while moving the vehicle towards the subgoals. At the same time, the localization system corrects the vehicle odometry error using the a priori model.

In order to implement the system, they built the so-called modalities M_i. A modality is a combination of particular instances of the functions in order to adapt the motion to different navigation contexts:

Modalities	Functionalities		
	Planning	Localization	Motion control
M_1	Voronoi	SLAM	EB
M_2	Voronoi	SLAM	mND
M_3	Voronoi	LPV	mND
M_4		SLAM	mpND

The experimentation consisted in a test of each modality in different environments in order to extract conclusions about the application domain of each of them. The environments were:

- E_1: long-term navigation in a 250 m circuit (Fig. 15).
- E_2: navigation in a long corridor (30 m).
- E_3: navigation in a long corridor (30 m), but random distributed obstacles. This affected the localization (since many segments of the model were occluded) and increased the difficulty of the navigation in the corridor.
- E_4: short-term navigation (around 12 m), but in very dense and narrow scenarios. Random distributions of obstacles created many areas with little space to maneuver and U-shaped obstacles.

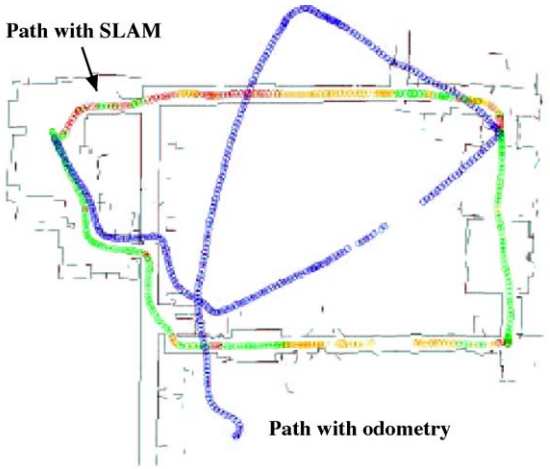

Fig. 15. Metric model obtained of the test arena (75 m × 50 m). Trajectory obtained using the localization with segments (SLAM) and only using odometry.

The results obtained are depicted in the next table:

E_1	E_2	E_3	E_4
M_1			
$N = 20$	$N = 20$	$N = 20$	$N = 5$
SR = 100%	SR = 100%	SR = 5%	SR = 0%
$d = 2320$ m	n/a	n/a	n/a
$v = 0.26$ m/s	n/a	n/a	n/a
M_2			
$N = 12$	$N = 12$	$N = 0$	$N = 5$
SR = 80%	SR = 80%	–	SR = 0%
$d = 870$ m	n/a	–	
$v = 0.28$ m/s	n/a	–	
M_3			
$N = 0$	$N = 20$	$N = 12$	$N = 0$
–	SR = 95%	SR = 100%	–
–	n/a	$v = 0.15$ m/s	–
–	n/a	n/a	–
M_4			
$N = 10$	$N = 12$	$N = 0$	$N = 10$
SR = 0%	SR = 80%	–	SR = 100%
n/a	n/a	–	$v = 0.14$ m/s
n/a	n/a	–	n/a

The parameters are: N the number of trials, SR the success rate, d the distance travelled, v the mean velocity and n/a is not available. The quantitative results change a lot among scenarios; however, they give and idea of modality performance. Notice that the number of tests is over 140, where 65 correspond to modalities that use the system presented here. We next outline some conclusions of the experiments, which are described in detail in Morisset's thesis.

The test E_1 corresponded to runs in a large scenario. In this scenario, modalities M_1 y M_2 carried out the task with a high success rate. The 20% of the failures with M_2 were due to the ND. This method produced oscillations in some conditions during the navigation in long passages due to oscillations between situations. The oscillations produced turns that degraded the segment-based localization, and in some cases, made it fail. When this situation arose a failure flag was launched. Another important remark is that in the experiments with M_1, none of the environments presented difficulty in the sense of maneuverability. Thus, the EB successfully drove the vehicle in all the situations. However, in the environments used for M_2, many closed obstacles and door passages were added (in some of them distances were about to 0.8 m). No collisions were found due to the obstacle avoidance difficulty. As it is described in the thesis, if M_1 were used in this environments, it would fail. Thus, the avoidance capabilities of the mND are better than the EB. In M_4, the high level planner is not used. Thus, there is no possibility to this modality to accomplish the task in this large scenario (250 m), since although the model used is 20 m width and is robot-centered, it is not sufficient to describe all the experimentation area (Fig. 15).

The experiment E_2 consisted in a large corridor (30 m). In this case, modalities M_2 y M_4 lose 20% due to the oscillations in the motion system that makes the localization fail. However, when the same experiment was carried out with M_3 (visual localization), the result is 95%. The only failure was due to the visual localization system (it did not detected a landmark). However, in E_3, some additional obstacles were added in the corridor. This converted the corridor in a very dense scenario where maneuverability turned troublesome. M_1 failed in the majority of the cases; however, M_3 achieved to solve all the cases. This is because the mND offers better motion control than the EB in confined spaces.

Finally, in E_4, the system was tested in an environment full of close obstacles (0.8 m clearance) and where U-shape and dead-ends where dynamically created everywhere. This type of experimentation would be similar to the one presented in the previous subsections. Modalities M_1 y M_2 did not achieve to solve this scenario. However, M_4 successfully solved all the tests. This is due to the local capacity of the mpND to address this trap situations. Property that the other motion systems do not have (in the mND the planning component is disabled and thus the method is pure reactive).

As personal conclusion of the authors paper, the previous results suggest that the modality more adapted to solve the navigation would be a new modality where the mpND proposed in this paper would be integrated with high level planning (Voronoi–SLAM–mpND). This would allow to obtain good results in large scenarios (as suggested by the performance of modality M_2 in E_1, where the mpND is integrated but without local planning, mND). Furthermore, these scenarios could have greater obstacle density, narrow passages and dead-ends (as suggested by the performance of M_4 in E_4).

The conclusion of Morisset is that each sensory motor function has its own advantage and its own drawback and no modality can cover efficiently any kind of environment or situation. The only way to cover these variety of cases is to exploit the complementarity of these modalities. Thus, Morisset built a supervision system [38] that works as follows. The robot performs some autonomous navigations. All along each navigation, the current state of supervision is updated. The relationship between supervision states and the appropriate modality for pursuing a task is learned as a Markov Decision Process (MDP) which provides a general policy for the task.

7. Discussion and comparison with other methods

A sensor-based motion control system was developed based on a hybrid architecture with three layers (modeling, planning and reaction). The interaction among the modules is a synchronous planner–reactor configuration where the planner computes tactical information to direct the reactivity. Our hybrid system differs from previous ones in the integration archi-

tecture and in the techniques used in each module. We discuss here alternative designs to the hybrid systems that do not use the reactor module performing an "any-time planning", and the impact of the techniques used to implement each module regarding other hybrid schemes.

As mentioned in Section 2, hybrid systems attempt to combine planning abilities with the best reactivity, robustness, adaptation and flexibility. Basically, these schemes combine a planner (deliberation) and a reactor (execution). The common strategy is to compute a path and use its course to direct the reactive module. Another architectural alternative to hybrid systems are those based on two modules, one to model and the other to plan (to drive the vehicle). They perform an "any-time" planning instead of using the reactive module to compute the motion. In some of them, the modeling and the planning are synchronous [46,47], whereas in others, the planning step is an asynchronous process that is only carried out in the areas that influence the progress to the goal [48–50]. These systems do not have the advantages of reactive behavior, such as rapidly adapting to changes and flexibility under unforeseeable circumstances. All these systems use planners that usually compute trajectories that skirt the obstacles, which appears to contrast with the avoidance task (along these lines, Murphy et al. presented a planner that spaces out the trajectory of the obstacles [49]). However, in our system, this situation is managed in a natural way by the reactor. Another difficulty of most systems is the reliance on a path to the goal, which is not always available in realistic scenarios (Section 5.4) leading to a failure in the motion system. Furthermore, these systems are not consistent with the idea that a vehicle should not be controlled directly with a planner [51], and it would be difficult to obtain the computation time to put them into practice [52].

Recently, both systems (hybrid and "any-time planning") have been combined by Ranganathan and Koenig using both paradigms to adapt the operation to the progression of the robot in the environment [53]. However, the above conclusions are derived when only the planner is used to drive the vehicle.

Focusing on hybrid systems, another matter is the influence of the techniques used to implement the planner, the model and the obstacle avoidance. With regard to the planner Ulrich and Borenstein [54] use a look ahead verification before executing the reactive algorithm [28]. The local trap situations are avoided by running the reactive method some steps before the algorithm is executed. The completeness of this strategy depends on the number of previous steps (maximum distance inspected). This solution is well suited for robots with limited computational capabilities because good navigational results are obtained even by reducing the maximum distance inspected. Our framework uses a complete planner which assures the tactical information to avoid the trap situations.

Our model is similar to [16,18]. However, their model represents the configuration space which increases with the distance traveled. The advantage is that global knowledge is incorporated as the robot progresses, and the navigation function of the planner does not have to be recomputed as long as the scenario remains the same. However, our model represents a local portion of the workspace, so it does not depend on the distance traveled. That is, the memory and computational time are fixed, whereas in the other methods, they increase as the robot progresses, since the model is continuously enlarged. In our model, the information around the robot is always available. In the other methods, it depends on the availability of time and memory. Furthermore, our model represents the workspace that can be rapidly updated in changing scenarios, which is not the case in methods that represent the configuration space.

Another key aspect is the reactive module. Brock and Khatib [16], Arras et al. [18], Hebert et al. [32], Ulrich and Borenstein [54], Gat [51] among others, are systems that use reactive planners that have problems driving in dense, complex and troublesome scenarios. This is not a problem in our system, since the typical limitations of other methods are avoided, such as a local trap, oscillations in dense scenarios, and the impossibility to obtain motion direction towards the obstacles or towards areas with large density of obstacles [33]. This results in a safe and robust motion in scenarios that are still troublesome for many existing methods. This point is also stressed in the experimental results section. There we show a quantitative/qualitative objective comparison with another obstacle avoidance method [20]. The conclusion is that our system exhibits greater maneuverability capabilities specially in dense, complex and troublesome scenarios.

8. Conclusions

Many industrial applications of vehicles for evolving scenarios could benefit from the technology of motion generation. These techniques would increase the degree of autonomy of the robots and reduce human intervention (which is especially important in dangerous or hostile environments). We present a sensor-based motion control system as a subset of a complete navigation system. The main contributions include the functional and computational aspects of the modules in the system and their integration, in addition to the strong experimental validation. As a consequence, this hybrid system is able to move vehicles robustly in very difficult environments, which are still troublesome for of the most motion systems.

The key result is the real time cooperation between the modules in the system. The information integrator module constructs a representation of the environment which is the base of the rest of modules. Then the planning module computes tactical information to direct the vehicle and the reactive module controls the motion. The three modules are integrated with an architecture that follows a synchronous configuration planner–reactor. It concentrates the best of the deliberative and reactive worlds, since the planning information helps to guide the motion towards zones without traps, and the reactive component quickly directs the execution according to the evolution of the environment (also considering areas that are not visible from the present position available in the model). All the modules are integrated so that the control loop is always closed, providing a motion command (there are no deadlocks in the system) with real-time performance. Furthermore, the modular structure of the system allows to replace the different modules easily, since the functional and computational aspects and their interfaces are clearly specified.

The functional design of each module and the specification of its interfaces leads to an adaptable and portable system. Thus, the system has been used on other robots [55], including two indoor and one outdoor scenarios at the *LAAS-CNRS*, France; one indoor in the *Technical University of Lisbon*, Portugal; and two indoor in the *University of Zaragoza*, Spain. In these cases, the modeling module was replaced by modules to process a 3D laser, ultrasounds following [35], or a pair of cameras [56]. In some cases, the reactive navigation module was replaced by another ND method adapted to work on platforms with a non-circular geometry and kinematic and dynamic constraints [57]. The substitution of these modules within the architecture was straightforward.

Another important point is the interaction of the system with other subsystems which are needed to construct a complete navigation system, such as planning, location and map building. For example, our system was integrated with a topological navigation system [58]. It constructs a global model of the environment in execution while relocating the vehicle and placing goals for exploration (our system drives the vehicle among the desired locations). The sensor-based system has also been incorporated in the *Robels* system [38], where it is used as one of the four sensory-motor functions that execute motion. We report in this paper this integration and many experimental results that validate the system. In addition, our system has been successfully integrated as a low level motion generator within the $G^{en}oM$ architecture [59] on a *Nomadic XR*4000 in the LAAS-CNRS, France, and is used to move the vehicle on a daily basis [60].

Acknowledgements

We would like to thank all the members of the R. Chatila group that hosted J. Minguez during his visit to the *LAAS-CNRS*, and especially R. Alami and T. Simeon for the helpful comments and discussions and S. Fleury for her help with the algorithm implementation on *XR*4000 *Nomadic*. We also thanks Bennoit Morisset of SRI International for his help in preparing the integration subsection of this paper. This work was partially supported by DPI2003-07986 Spanish project.

Appendix A

The *Nomadic XR4000* is a circular and holonomic vehicle equipped with a 2D SICK placed 0.24 m in the frontal part, with a 180° field of view, a reach of 32 m and a time period of 0.20 s. The robot has a *Pentium II* where all the computations are carried out.

References

[1] J.C. Latombe, Robot Motion Planning, Kluwer Academic, 1991.

[2] J.A. Castellanos, J. Montiel, J. Neira, J.D. Tardos, The SPmap: a probabilistic framework for simultaneous localization and map building, IEEE Trans. Robotics Automation 15 (5) (1999) 948–952.

[3] J.J. Leonard, H.J.S. Feder, A computationally efficient method for large-scale concurrent mapping and localization, in: D. Koditschek, J. Holler-bach (Eds.), Robotics Research: The Ninth International Symposium, Springer Verlag, Snowbird, Utah, 2000, pp. 169–176.

[4] M.W.M.G. Dissanayake, P. Newman, H.F. Durrant-Whyte, S. Clark, M. Csorba, A solution to the simultaneous localization and map building (SLAM) problem, IEEE Trans. Robotics Automation 17 (3) (2001) 229–241.

[5] S. Thrun, W. Burgand, D. Fox, A real-time algorithm for robot mapping with applications to multirobot and 3D mapping, in: IEEE International Conference on Robotics and Automation, San Francisco, CA, 2000, pp. 321–328.

[6] J.M. Buhmann, W. Burgard, A.B. Cremers, D. Fox, T. Hofmann, F.E. Schneider, J. Strikos, S. Thrun, The mobile robot RHINO, AI Magazine 16 (2) (1995) 31–38.

[7] S. Koenig, R. Simmons, Xavier: a robot navigation architecture based on Partially Observable Markov Decision Process models, in: D. Kortenkamp, R.B. Murphy (Eds.), Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems, MIT Press, 1998, pp. 91–122.

[8] R. Arkin, Towards the unification of navigational planning and reactive control, in: Working Notes of the AIII Spring Symposium on Robot Navigation, Stanford University, 1989, pp. 1–6.

[9] R. Arkin, Motor schema based navigation for a mobile robot: an approach to programming by behavior, in: IEEE International Conference on Robotics and Automation, Raleigh, USA, 1987, pp. 264–271.

[10] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, Int. J. Robotics Res. 5 (1986) 90–98.

[11] P. Agree, D. Chapman, What are the plans for? J. Robotics Autonomous Syst. 6 (1990) 17–34.

[12] D. Lyons, A. Hendriks, Planning, reactive, in: S. Saphiro, J. Wiley (Eds.), Encyclopedia of Artificial Intelligence, 1992, pp. 1171–1182.

[13] T. Dean, M. Wellman, Planning and control, in: Morgan-Kauffman, San Mateo, CA, 1991.

[14] R. Arkin, Behavior-Based Robotics, The MIT Press, 1999.

[15] S. Ratering, M. Gini, Robot navigation in a known environment with unknown moving obstacles, in: International Conference on Robotics and Automation, Atlanta, USA, 1993, pp. 25–30.

[16] O. Brock, O. Khatib, High-speed navigation using the global dynamic window approach, in: IEEE International Conference on Robotics and Automation, Detroit, MI, 1999, pp. 341–346.

[17] J. Minguez, L. Montano, N. Simeon, R. Alami, Global nearness diagram navigation (GND), in: IEEE International Conference on Robotics and Automation, Seoul, Korea, 2001, pp. 33–39.

[18] K. Arras, J. Persson, N. Tomatis, R. Siegwart, Real-time obstacle avoidance for polygonal robots with a reduced dynamic window, in: IEEE International Conference on Robotics and Automation, Washington, USA, 2002, pp. 3050–3055.

[19] O. Brock, O. Khatib, Real-time replanning in high-dimensional configuration spaces using sets of homotopic paths, in: IEEE International Conference on Robotics and Automation, San Francisco, USA, 2000, pp. 550–555.

[20] S. Quinlan, O. Khatib, Elastic bands: connecting path planning and control, in: IEEE International Conference on Robotics and Automation, vol. 2, Atlanta, USA, 1993, pp. 802–807.

[21] I. Ulrich, J. Borenstein, VFH+: reliable obstacle avoidance for fast mobile robots, in: IEEE International Conference on Robotics and Automation, 1998, pp. 1572–1577.

[22] W. Choi, J. Latombe, A reactive architecture for planning and executing robot motion with incomplete knowledge, in: IEEE/RSJ International Workshop on Intelligent Robots and Systems, Osaka, Japan, 1991, pp. 24–29.

[23] J. Barraquand, J. Latombe, On non-holonomic mobile robots and optimal maneuvering, in: Intelligent Symposium on Intelligent Control, Albany, 1989, pp. 340–346.

[24] A. Stentz, The focussed *D** algorithm for real-time replanning, in: International Joint Conference on Artificial Intelligence (IJCAI), Montreal, CA, 1995, pp. 1652–1659.

[25] B.H. Krogh, C.E. Thorpe, Integrated path planning and dynamic steering control for autonomous vehicles, in: IEEE International Conference on Robotics and Automation, San Francisco, USA, 1986, pp. 1664–1669.

[26] R.B. Tilove, Local obstacle avoidance for mobile robots based on the method of artificial potentials, in: IEEE International Conference on Robotics and Automation, vol. 2, Cincinatti, OH, 1990, pp. 566–571.

[27] Y. Koren, J. Borenstein, Potential field methods and their inherent limitations for mobile robot navigation, in: IEEE International Conference on Robotics and Automation, vol. 2, Sacramento, CA, 1991, pp. 1398–1404.

[28] J. Borenstein, Y. Koren, The vector field histogram-fast obstacle avoidance for mobile robots, IEEE Transactions on Robotics and Automation 7 (1991) 278–288.

[29] R. Simmons, The curvature-velocity method for local obstacle avoidance, in: IEEE International Conference on Robotics and Automation, Minneapolis, USA, 1996, pp. 3375–3382.

[30] D. Fox, W. Burgard, S. Thrun, The dynamic window approach to collision avoidance, IEEE Robotics and Automation Magazine 4 (1) (1997).

[31] W. Feiten, R. Bauer, G. Lawitzky, Robust obstacle avoidance in unknown and cramped environments, in: IEEE International Conference on Robotics and Automation, San Diego, USA, 1994, pp. 2412–2417.

[32] M. Hebert, C. Thorpe, A. Stentz, Intelligent Unmanned Ground Vehicles: Autonomous Navigation Research at Carnegie Mellon, Kluwer Academic Publishers, 1997.

[33] J. Minguez, L. Montano, Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios, IEEE Transactions on Robotics and Automation 20 (1) (2004) 45–59.
[34] A. Elfes, Sonar-based real-world mapping and navigation, IEEE Journal on Robotics and Automation 3 (3) (1987) 249–265.
[35] J. Borenstein, Y. Koren, Histogramic in-motion mapping for mobile robot obstacle avoidance, IEEE Journal on Robotics and Automation 7 (4) (1991) 535–539.
[36] J. Foley, A.V. Dam, S. Feiner, J. Hughes, Computer Graphics, Principles And Practice, second ed., Addison Wesley, 1990.
[37] B. Morisset, Vers un robot au comportament robuste. Apprendre a combiner des modalities sensori-motrices complementaires, Ph.D. thesis, LAAS-CNRS, November 2002.
[38] B. Morisset, M. Gallab, Learning how to combine sensory-motor modalities for a robust behavior. Advances in Plan-Based Control of Robotic Agents, Lecture Notes in Artificial Intelligence, 2466, Springer, 2002, pp. 157–178.
[39] T. Simeon, B.D. Wright, A practical motion planner for all-terrain mobile robots, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1993.
[40] P. Moutarlier, R. Chatila, Stochastic multisensory data fusion for mobile robot location and environment modelling, Proceedings of ISER, June 1989.
[41] J. Leonard, H. Durrant-Whyte, Mobile robot localization by tracking geometric beacons, IEEE Transaction on Robotics and Automation 7 (1991) 367–382.
[42] V. Ayala, J. Hayet, F. Lerasle, M. Devy, November visual localization of a mobile robot in indoor environments using planar landmarks, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2000, pp. 275–280.
[43] J. Hayet, F. Lerasle, M. Devy, Planar landmarks to localize a mobile robot, in: SIRS, July 2000, pp. 163–169.
[44] M. Khatib, H. Jaouni, R. Chatila, J. Laumond, Dynamic path modification for car-like non-holonomic mobile robots, in: IEEE International Conference on Robotics and Automation, Alburquerque, Mexico, 1997, pp. 2920–2925.
[45] M. Khatib, Sensor-based motion control for mobile robots, Ph.D. thesis, LAAS-CNRS, 1996.
[46] N. Roy, S. Thrun, Motion planning through policy search, in: IEEE-RSJ International Conference on Intelligent Robots and Systems, Switzerland, 2002, pp. 2419–2424.
[47] C. Stachniss, W. Burgard, An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments, in: IEEE-RSJ International Conference on Intelligent Robots and Systems, Switzerland, 2002, pp. 508–513.
[48] A. Stentz, M. Hebert, A complete navigation system for goal acquisition in unknowm environments, Autonomous Robots 2 (1995) 127–145.
[49] R. Murphy, K. Huges, E. Noll, An explicit path planner to facilitate reactive control and terrain preferences, in: International Conference on Robotics and Automation, Minneapolis, Minnesota, 1996, pp. 2067–2072.
[50] C. Urmson, R. Simmons, I. Nesnas, A generic framework for robotic navigation, in: IEEE Aerospace Conference, 2003, pp. 1–8.
[51] E. Gat, On three-layer architectures, in: D. Koternkamp, R. Bonasso, R. Murphy (Eds.), Artificial Intelligence and Mobile Robots: Case of Studies of Successful Robot Systems, MIT Press, 1998, pp. 1171–1182.
[52] R. Fikes, N. Nilsson, Strips: a new approach to the application of theorem proving to problem solving, Artif. Intell. 2 (1971) 189–208.
[53] A. Ranganathan, S. Koenig, A reactive architecture with planning on demand, in: International Conference on Robotics and Automation, Las Vegas, Nevada, 2003, pp. 1462–1468.
[54] I. Ulrich, J. Borenstein, VFH: local obstacle avoidance with look-ahead verification, in: IEEE International Conference on Robotics and Automation, San Francisco, USA, 2000, pp. 2505–2511.
[55] J. Minguez, L. Montano, Robot navigation in very complex dense and cluttered indoor/outdoor environments, in: 15th IFAC World Congress, Barcelona, Spain, 2002.
[56] H. Haddad, M. Khatib, S. Lacroix, R. Chatila, Reactive navigation in outdoor environments using potential fields, in: IEEE International Conference on Robotics and Automation, vol. 2, Leuven, Belgium, 1998, pp. 1232–1237.
[57] J. Minguez, J. Osuna, L. Montano, A divide and conquer strategy to achieve reactive collision avoidance in troublesome scenarios, in: IEEE International Conference on Robotics and Automation, Minessota, USA, 2004.
[58] D.V. Zwynsvorde, T. Simeon, R. Alami, Building topological models for navigation in large scale environments, in: IEEE International Conference on Robotics and Automation, Seul, Korea, 2001, pp. 23–29.
[59] S. Fleury, Architecture de controle distribuee pour robots autonomes: principes, conception et applications, Ph.D. thesis, Universite Paul Sabatier, 1996.
[60] R. Alami, I. Belousov, S. Fleury, M. Herb, F. Ingrand, J. Minguez, B. Morisset, Diligent: towards a human-friendly navigation system, in: IEEE-RSJ International Conference on Intelligent Robots and Systems, Takamatsu, Japan, 2000, pp. 2094–2100.

Javier Minguez received the physics science degree in 1996 from the Universidad Complutense de Madrid, Madrid, Spain, and the PhD degree in computer science and systems engineering in 2002 from the University of Zaragoza, Zaragoza, Spain. During his student period, in 1999, he was a research visitor in the Robotics and Artificial Intelligence Group, LAAS-CNRS, Toulouse, France. In 2000, he visited the Robot and ComputerVision Laboratory (ISR-IST), Technical University of Lisbon, Lisbon, Portugal. In 2001, he was with the Robotics Laboratory, Stanford University, Stanford, USA. He is currently a full-time Researcher in the Robot, Vision and Real Time Group, University of Zaragoza. His research interests are obstacle avoidance, motion estimation and sensor-based motion systems for mobile robots.

Luis Montano was born on 6 September 1958 in Huesca, Spain. He received the industrial engineering degree in 1981 and the PhD degree in 1987 from the University of Zaragoza, Spain. He is an Associate Professor of Systems Engineering and Automatic Control at the University of Zaragoza, Spain. He has been Head of the Computer Science and Systems Engineering Department of the University of Zaragoza. Currently he belongs to the staff of the Aragon Institute of Engineering Research of the University of Zaragoza, coordinating the Production Technologies and Logistic area. He is the coordinator of the Robotics, Perception and Real Time group at the Aragon Institute of Engineering Research of the University of Zaragoza, and he is the principal researcher in several robotic projects. His main areas of interest in robotics are mobile robot navigation, dynamic environments modeling and cooperative robots.