HyCAR – A Robust Hybrid Control Architecture for Autonomous Robots

Farlei J. Heinen, Fernando S. Osório
UNISINOS University – Universidade do Vale do Rio dos Sinos
Mestrado em Computação Aplicada - PIPCA
Av. Unisinos 950, São Leopoldo – RS - Brasil
{farlei, osorio} @exatas.unisinos.br
http://ncg.unisinos.br/robotica/simulador

Abstract. This work presents a new hybrid architecture applied to autonomous mobile robot control - HyCAR (Hybrid Control for Autonomous Robots). This architecture provides a robust control for robots as they become able to operate and adapt themselves to different environments and conditions. We designed this new hybrid control architecture, integrating the two main techniques used in robotic control (deliberative and reactive control) and the most important environment representation techniques (grids, geometric and topological maps), through a three-layer architecture approach (vital, functional and deliberative layers). To guarantee the robustness of our control system, we also integrated a localization module based on Monte Carlo localization method. This localization module possesses an important role in our control system, and supplies a solid base for the control and navigation of autonomous mobile robots. In order to validate our control architecture, a realistic simulator of mobile robots was implemented (SimRob3D) allowing the practical use of the proposed system. We implemented several three-dimensional environment models, as well as diverse sensorial and kinematics models found in actual robots. Our simulation results had demonstrated that the control system is perfectly able to determine the mobile robot position into a partially known environment, considering local or global localization, and also to determine if the robot needs to re-localize itself given an incorrect localization. In navigation tasks the robot was able to plan and follow self-generated trajectories in a dynamic environment, which can include several unexpected static and mobile obstacles. We also demonstrated that with the integration of topological and grid information we improved planning algorithm execution.

1. Introduction

The robotic mobile vehicles [1] are an interesting subject of research in the field of Artificial Intelligence, which tries to improve the autonomy and robustness of those vehicles. The main difference between mobile robots and other robotic research fields, such as the field related to robotic manipulators, is the fact of this kind of robotic application operates in complex environments that modify itself dynamically and usually are unpredictable. We are confronted to large scale environments due to robot locomotion capacity, and these environments can also enclose other static or mobile unknown obstacles. To operate in this kind of environments the robot must be able to acquire and use all available knowledge about the environment (map), to estimate the robot position in this environment, to possess the ability to recognize obstacles, and to answer in real time to different situations that can occur in this dynamic environment. Moreover, all these functionalities must operate in parallel, and then, modularity is indispensable. The tasks responsible for activities like to perceive the environment, to localize the robot in the environment, to plan trajectories and to move the robot preventing collisions, attempt to solve together the main problems treated in the study of the autonomous mobile robots.

The tasks that allow a mobile robot to move from a specified point to another point in the environment are called robotic navigation tasks. We are confronted to several problems that make the navigation in a real environment very complex: environments are dynamic and they can be modified as time passes (e.g. furniture can be moved from one position to another one); changes in the environment can obligate to change the initial task planning; mobile obstacles with unpredictable exact trajectories can be found moving continuously around the environment (e.g. walking humans); data from sensors can be inexact and subject to errors, as well as, commands sent to actuators can be incorrectly executed; among others problems. All these problems demand robots that are able to deal with dynamic and uncertain data.

The first main problem we need to solve in autonomous mobile robot navigation is the problem of robot localization. Without knowing the actual position of the robot related to the known map of the environment (some initial environment representation), a control system has a lot of difficulties to control the robot in order to perfectly accomplish one specific task. The execution of complex tasks can be made impossible without having an estimated almost precise robot localization. A robust control system must have the capacity of self-localization in an environment, using the sensorial information and also the available information representing the environment (map) to estimate the robot position. This problem is also complex, and usually represents the "essential base" of a robotic control system.

The main goal of this work is to develop a robust control system for autonomous mobile robots that are able to operate and automatically adapt their behavior accordingly to different environments conditions. The control system must be able to determine the robot position using sensorial information and available environment map, even if the data are approximate and imprecise. The system must be able to keep, during navigation and tasks execution, an estimation of the correct robot position, starting with an initial approximate known position (local localization), or yet without any information about the robot initial position (global localization). During navigation the system should be able to detect and recover from incorrect position estimations (robot positioning error detection and relocalization). The system must be able to control the robot and navigate in a dynamic environment, preventing collisions with static and mobile obstacles. In order to achieve this goal a hybrid architecture for robot control and navigation was proposed and implemented.

In the next sections we will describe the proposed architecture – HyCar/Cohbra (section 2), the implemented simulator – SimRob3D – that was used to validate and to test our hybrid architecture (section 3), and some important results demonstrating the robustness and performance of our system (section4).

2. HYCAR / COHBRA – Robot Control Architecture

In this work we present a new hybrid architecture of robot control - HyCAR/COHBRA (Hybrid Control of Autonomous Robots, or using the Portuguese acronym from "COntrole HíBrido de Robôs Autônomos") [10]. This architecture integrates several important modules and components (data structures and methods) currently used to control autonomous mobile robots in one single block.

The mobile robot control architecture was structured in 3 layers: vital layer, functional layer and deliberative layer (Fig. 1). Each one of them is responsible for the reactive control (short term action), task execution control (mid term actions), and task planning (long term actions), respectively. This type of system was also adopted in other similar works, like those developed by Gat [2] with the ATLANTIS architecture, and by Bonasso et al. [3] with the 3T architecture. However, our architecture proposes some extensions like the inclusion of the localizer module and the integration of multiple views of the environment (maps).

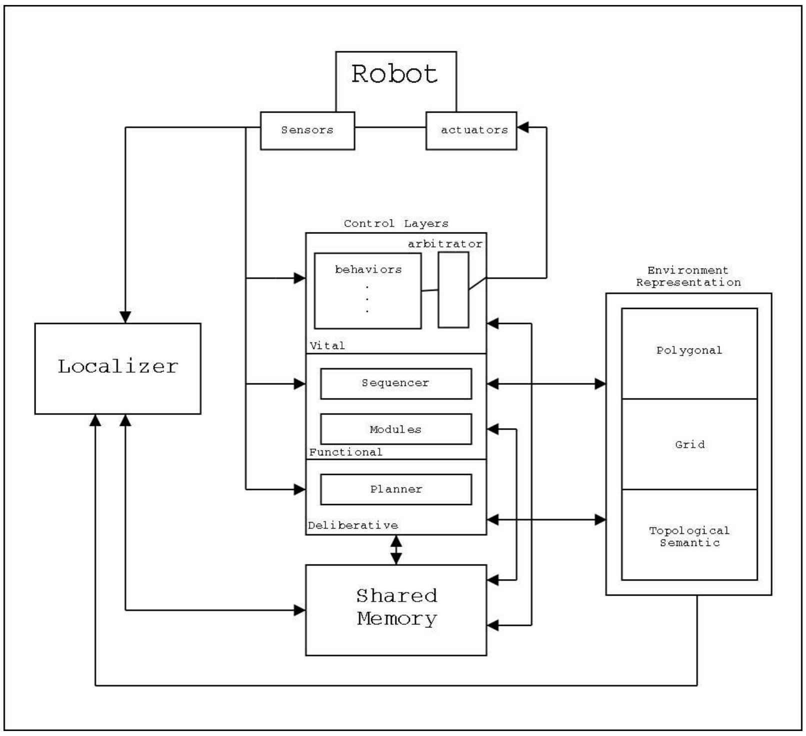


Figure 1 - Robot control architecture diagram, showing its main components: Control Layers (vital, functional and planner); Localizer; Environment Representation (polygonal, grid and topological maps); and Shared Memory.

In order to obtain a solid base for the execution of tasks implemented by the three Control Layers (vital, functional and planner), an additional module was integrated into the architecture: the localizer module. This localizer module must supply an estimated robot position related to the available environment map, and also using the sensorial data, it can continuously validate the estimated robot position. The localizer module is an essential part of our autonomous mobile robot control architecture, since we need an approximately correct robot position to perform tasks like: 'Move the robot from Office H to Room O'. If the system doesn't know where the robot is, it will be very hard to control it in order to achieve missions such as the above described task.

So, in the proposed architecture presented in this work, the localizer plays a central and essential role. The localizer allows us to determine the robot position, to detect changes in the environment maps and to detect static and dynamic obstacles. Since we can estimate the sensorial data according to a determined robot position and known environment configuration, then we can check it against the actual sensor data, and use this information to perform the above tasks (re-localize, detect changes and detect obstacles).

The specific form as the environment is represented internally in a control system, determines its precision and performance. Each one of the main known approaches applied to autonomous mobile robot control uses the most suitable environment representation adapted to some specific algorithm and/or purpose, choosing mainly between techniques like: discrete grids, geometrical maps, topological maps, probabilistic maps, Voronoy diagrams or potential field [11,12]. The proposed architecture we adopt in this work integrates some of the most important approaches of environment representation, composing a hybrid scheme of environment maps, divided into layers: polygonal layer, grid layer and topological/semantic layer (Fig. 1).

The communication between all system components of our control architecture is allowed through a common data integration area implemented by shared memory. Through the use of this central repository of information, the several components of our architecture can exchange vital information for the perfect functioning of the autonomous mobile robot control system.

2.1 HYCAR / COHBRA Control System

Based on the concepts proposed in the HyCAR/COHBRA architecture [10] was implemented one specific control system. A special attention was given to the localizer module in the development of this system, as it is considered the main component of our control system and one of the main focus of this work. If the autonomous mobile robot possesses a good estimation of its actual position in the environment, then the navigation task becomes simpler and more precise. In the next sections we will describe the main components of the HyCAR hybrid control system: environment representation, localizer module and control layers.

2.1.1. Internal Representation of the Environment

In the internal representation of the environment we used together the three environment representation approaches defined in the control architecture: polygonal map, grid map and topological/semantic map. The polygonal map is mainly used by the localizer module to estimate the robot position, and it is initially supplied by the user in DXF format (AutoCAD Data) defining a blueprint of the environment (complete or approximate). The grid map is used by the deliberative layer to plan trajectories from a starting point to a goal destination (A* Algorithm [8]). The grid maps are generated from the polygonal map splitting the environment into cells using a predefined target resolution. The main function of topological/semantic maps is to improve planning performance, given some additional information that helps the planner to reduce space-state search and optimize the robot trajectory calculation.

2.1.2. Localizer Module

The Localizer Module was implemented using a Monte Carlo Localization (MCL) technique based on Fox et al [4] work. Monte Carlo localization approach possesses several advantages: it requires less computational resources than the majority of other techniques; it concentrates the resources and computational efforts in the areas of bigger interest for the localization; and as it is a probabilistic technique it can aggregate additional information about the robot position (e.g. certainty values associated to each estimated robot position).

One of the main reasons of our choice of a Monte Carlo localization method implementation and its integration into the HyCAR control system was its capacity to solve the three great problems related to robot localization: local localization, global localization, and relocalization.

In the local localization, the Monte Carlo localizer is able to keep a correct robot position from an initial specified position with an acceptable level of error. This method performs well during continuous robot displacements without the necessity of an external calibration function. The local localization is usually done when we start to use a mobile robot, where the user assigns the approximate position of the robot in the environment (e.g. Office H on the left of the desk). The Monte Carlo localizer is also capable to perform a global localization, without need of any initial robot position information in order to determine the actual robot position. This localization technique is also capable of re-localize the robot, first detecting when the estimated robot position, apparently correct, does not reflect the actual robot position, and then performing a global localization to search for a new approximately correct robot position.

Monte Carlo Algorithm

The Monte Carlo localization (MCL) algorithm is divided in two phases: one phase of movement and the other phase of sensorial reading. Initially '*N*' samples (particles) are generated, uniformly distributed in the entire environment map (global localization), or distributed around the most likely robot position (local localization). Each sample is composed of a robot estimated position (x, y, direction) and the associated certainty.

In the ***movement phase***, the robot is activated and the MCL algorithm generates new '*N*' samples, used to approximate the new robot position after this action. Each sample is randomly generated from the previous samples set. Samples are chosen from the old samples set with basis on their certainty that affects the probability of being preserved. Suppose L' being one position in the old sample. The new position L in the new sample is generated using P(L | L', a), thus considering the observed action of movement 'a'. Before the correct robot position be achieved, it is possible to apply a certain threshold specifying the generation of some really new samples (samples renewal threshold).

In the ***sensorial reading phase***, information are incorporated readapting the weight (certainty) of each sample in the set, using the sensorial values to estimate the certainty of each specific sample related to its position. This phase is mainly based on the idea that we can estimate the sensorial reading of the robot for a specific position related to the known environment map, and then compare this estimated sensorial data within the actual sensorial data obtained from the robot. From this comparison we can obtain the certainty value related to an estimated robot position.

To determine if the robot is or not correctly localized, we use the certainty value of the best sample (the one with the biggest certainty into the set) plus a dispersion measure of the whole set (that indicates if the particles are mainly concentrated around one specific position). These two information can help us to determine if the robot position was correctly estimated.

One problem of the Monte Carlo localization is that it is assumed that: (*i*) the environment is static; (*ii*) the representation of the environment corresponds to the real environment; and (*iii*) mobile obstacles do not exist. This kind of situation is not usually found in a real environment, for this reason it was necessary to use certain special techniques to treat this problem in our control system.

One approach to solve this problem was proposed by Fox [6] using "filters" to ignore certain readings from the sensors when these do not match with the expected inputs. Fox considered two techniques, entropy filters and filters of distance. The technique we adopted in our implementation of MCL module was the distance filter. This filter was better suitable to the kind of sensors implemented in our system (distance sensors).

Besides allowing the autonomous mobile robot to localize itself in a dynamic environment, the distance filter helps other modules of HyCAR control system. It makes available in the shared memory information about the filtered sensors, thus allowing the system to detect new static obstacles not present in environment maps. Filtered sensors indicate a difference between the internal representation and the real environment. The available data about filtered sensors is used by the functional layer to update the internal representation of the environment. The only restriction of our system is during the localization process when we can work into a partially known environment but with no dynamic (mobile) obstacles. Once the robot position is known (correctly localized) then we can work in an environment with no restrictions: including unmapped static and mobile obstacles.

2.1.3. Vital Layer

The **vital layer** is responsible for reactive control of the autonomous mobile robot, composed of several simple process executed in parallel: elementary robot behaviors. These behaviors made an association between sensorial entrances and commands send to the actuators. Each behavior can be seen as a "sensorial-motor reaction", reacting directly to the environment stimuli.

We implemented 5 elementary behaviors in the vital layer, targeting to enable the robot: to follow trajectories calculated by the deliberative layer; to help the localizer module to determine the robot position; and to preserve the physical integrity of the robot. When following a pre-defined trajectory the autonomous mobile robot must be able to avoid and deviate from the unexpected obstacles (static or dynamic). The five behaviors implemented in the vital layer are: (*i*) stop; (*ii*) wander; (*iii*) avoid obstacles; (*iv*) move in direction to near target position; and (*v*) go back. The “avoid obstacle” behavior is based on the potential field method (VFF) as used by Borenstein & Koren [5].

The interactions between these behaviors are managed through an *arbitrator*. The arbitrator has the function of activate or inhibit certain behaviors, depending on the commands received from the sequencer in the functional layer. The arbitrator can also be programmed to compute fusion rules of behaviors outputs, in order to integrate ambiguous outputs from different behaviors.

2.1.4. Functional Layer

The **functional layer** is composed of various modules that interact among themselves, executing different functions responsible for the integration of the control system components. One of these functions of this layer is to select which elementary behaviors from the vital layer will be executed at a time, and to provide parameters to these elementary behaviors (e.g. next target position to move). With the information provided by functional layer and using the specification of the execution sequence of these elementary behaviors, the robot can execute high level tasks defined by the deliberative layer planner.

The *sequencer* of the functional layer was implemented in the form of a finite-state automaton. Each state of this automaton indicates for the *arbitrator,* in the vital layer, which behaviors must be set on or must be inhibited. So, the sequencing is executed through inhibitions of some specified outputs coming from elementary behaviors. There is no module in the functional layer that can act directly in the control of the robot actuators, which are controlled by the arbitrator in the vital layer. The modules of the functional layer are responsible for supplying information to the elementary behaviors of the vital layer that will be active, and inhibit the outputs of the behaviors that should be deactivated.

The functional layer is also composed by other functional modules that play auxiliary tasks in the control process of the autonomous mobile robot. In the HyCAR control system where specified a series of auxiliary functional modules. One of these modules is responsible for monitoring and update the internal representation of the environment, and the others provide parameters that assist other modules in different control layers.

2.1.5. Deliberative Layer

The **deliberative layer** has the only function to perform trajectories planning, from the robot position to a final user defined position. The planner may be activated by the user (new destination position specification) or it can be interrupted by requests originated from the functional layer. Sometimes the functional layer can detect an impossibility of plan execution (route no longer available due to obstacles) and then request a new planning.

The planning task is processed in two phases. In the first one, using the topological information (connectivity graph), a pre-planning is executed and determines the sequence of topological regions that will be traversed in the final path. The Dijkstra algorithm [7] was used in this initial phase, and this information will be used to optimize the next phase of planning. In the second phase, the A* algorithm [8] is used to calculate the definitive trajectory based on a grid representation of the environment map. The grid representation based algorithms can be great CPU time-consuming tasks, depending on the grid size. In our system, this task is optimized using the pre-planning phase to reduce the size of the grid used by the path-planning algorithm.

The trajectory planning obtained using the A* algorithm produces a way composed by a sequence of cells that must be followed to reach the destination position. This sequence of cells is converted into a sequence of points into the polygonal representation using the center of cell as base. The final plan is one trajectory to be followed by the autonomous mobile robot, from its current position to the destination position, composed by a sequence of points in the polygonal layer representation. This plan is made available into the shared memory to be used by the functional layer.

3. SimRob3D Simulator

We implemented a robot simulator that includes all necessary resources to realize experiments with autonomous mobile robots placed in a dynamic environment. This implementation was created in order to validate our proposed control system. This simulator was called SimRob3D (Mobile Robots Simulator based on Three-dimensional Environments) [10]. The main characteristic of this simulator resides in the fact that it implements our proposed control architecture using a three-dimensional environment for the navigation of the simulated mobile robots. The environment structure and objects can be designed with three-dimensional modeling software existing in the market (AutoCad, 3D Studio, among others), once we adopted in our simulator a common standard 3D data file format, the ".3DS". This file format allows us to specify different elements composing the robot environment (walls, objects, light, textures), resulting in an environment with a high level of realism if compared with other environments used in some 2D based simulators.

The simulator allows the user to place and configure obstacles, also allowing moving them in real time during simulation. The obstacles can also be pre-programmed to move in cyclical trajectories. The simulator possess several sensorial and kinematics models (e.g. encoder, sonar, infrared and laser sensors; Ackerman and differential kinematics), allowing the user to configure different types of robots.

All the sensors and actuators interact directly with the three-dimensional environment, becoming the simulation more realistic. Sensor and actuators are also modeled in a realistic way, i.e. we included in their model a gaussian error in order to introduce input sensor errors (e.g. noise) and output control errors (e.g. wheels sliding). These models of sensors and actuators provided to us a more realistic and non-deterministic robot behavior.

Another important characteristic of our simulator is its modularity. The controller is programmed separately from other simulation functions, implemented as a dynamic library

(DLL). The controller is loaded at execution time, and it can be implemented using any language chosen by user. As the controller is completely separated from the robot and environment simulation, it has a well defined interface to communicate with the robot. The only information exchanged between robot controller and the other simulation modules are basically "*Get_Sensors*" and "*Send_Command*". So, we can easily replace the simulated robot by an actual robot. HyCAR controller was implemented in that way, as a separated module that communicates with other SimRob3d modules.

4. Simulation Results

We used SimRob3D to validate HyCAR robot control architecture in *localization* and *navigation* tasks. Several *localization experiments* in static and dynamic environments were carried out: using a perfect environment map and using a modified environment map (with some static and dynamic/mobile obstacles added, which are not present in the original map). We evaluated the capacity of our system to perform local and global localization, and also we test the ability of the system to re-localize the robot when an incorrect initial position was informed to the robot. These experiments were executed using the original Trinity environment map [9] (figure 2). We also used some variations of that environment, in order to evaluate each type of localization, where the robot available environment map was set to the original Trinity environment, but during simulation the robot interacted with a modified map with some removed objects and added obstacles.

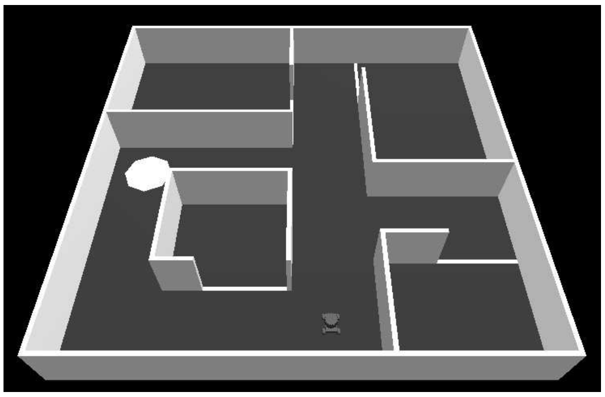


Figure 2 - Three-dimensional environment used in localization and navigation experiments based upon Trinity Fire-Fighting Contest environment map [9].

Table 1 shows the simulation results, using a 6x8cm Ackerman based robot, with the localizer module configured to use: 1000 particles sampling, 16 sonar sensors (30 degree wide range field, 100.0 to 1.0 cm distance sensibility, 5% of average noise) and 2 actuators - wheels and steering (10% average error on encoders, 1% average error on command response). Experiments were made with robot following a pre-defined trajectory, with a total of 2700 simulation cycles. Each cycle corresponds to a set of sensorial readings received from the robot and used by the localizer module to estimate the robot position. Table 1 values are the average obtained from10 different simulations results.

Table 1 - Results obtained in localization experiments.

Experiment	**Number of Localization Cycles (Correct Position Found)**	**Number of Localization Cycles**
Local Localization	0	2700
Global Localization	674.4	2025.5
Re-localization	681.8	2018.2

Navigation experiments were carried out in static and dynamic environments with some static or mobile obstacles added on it. The main objective was to evaluate the capacity of the control system to conduct the robot in these environments, following the plans supplied by the deliberative layer. The robot was placed in an initial position and the user specified several target destinations.

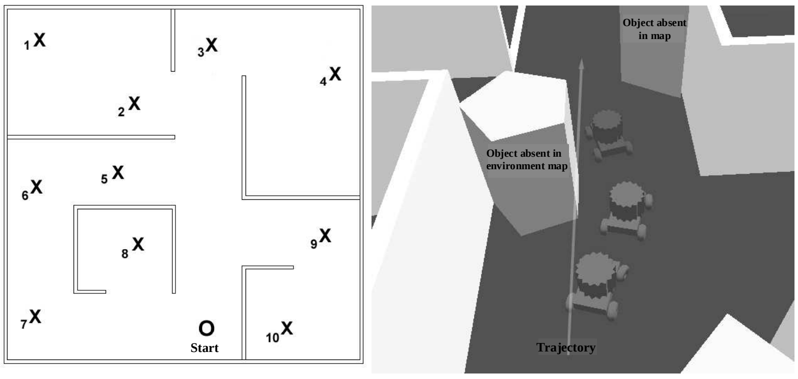
(a)  (b)

Figure 3 - Trinity environment map with destination points used in navigation experiments (a). Robot movement sequence executed to avoid an unexpected obstacle (b).

The results obtained in navigation experiments had shown that, in all simulations (10 simulations for each experiment: from the initial position, move to destination points 1-10; see Fig. 3(a), the control system was able to guide the mobile robot and achieve destination position, deviating from all unmapped obstacles (static or mobile), as can be seen in Fig. 3(b). The localizer module was able to keep a correct robot position during the entire trajectory, with an average certainty of 98% and a dispersion of 0.73cm with 91% particles grouped. Calculating the difference between the controller estimated position and the simulated robot position (actual position), we obtained a positioning error of 2.16cm in average computed over all simulation execution, in a 6.15m^2 environment.

We also carried out some experiments to determine if the use of a hybrid environment map representation can improve the planning algorithm execution. We used the same navigation tasks as described in Fig. 3(a). The planning execution time was measured for A* algorithm with no pre-planning and after that using our pre-planning algorithm based on topological information. The obtained results showed an improvement of global planning time execution with average gain of 68.5%.

These results demonstrate an efficient integration between the deliberative layer and the vital layer (reactive). The robot was capable to accomplish different navigation tasks even in presence of a modified environment with mobile obstacles. The localizer module also demonstrated the capacity to keep a correct robot position estimation even in a partially known environment (modified environment).

Our experiments demonstrated that our hybrid control system was able to keep an estimated correct position in all simulations configurations, with very good local localization ability. The experiments with global localization and re-localization obtained good resulted in most of tested environments, having some difficulties only in environments with a great number of modifications. In the navigation tasks, the control system was perfectly able to move the robot up to the specified objectives in all types of environment we used in our experiments. Some videos showing SimRob3D experiments can be viewed in the simulator site on the Internet: *http://ncg.unisinos.br/robotica/simulador/index_us.html*

5. Conclusion

The results had shown that HyCAR hybrid control system was perfectly able to control autonomous mobile robots. The system was able to execute navigation tasks in static and dynamic (modified) environments also including the presence of mobile obstacles. The mobile robot was able to follow global trajectories planned by the deliberative layer module, and using their reactive skills from the vital layer module, it was possible to navigate without colliding with any kind of obstacle or being imprisoned by local minima, even in a dynamic environment. Our experimental results showed that cooperation between the global planning algorithm and the local reactive algorithm (intermediated by the functional layer) provided a robust robot control and navigation system. The localization module also shows a capacity to determine and to keep a good estimation of the robot position in the environment.

The proposed architecture fulfills our main goals: (*i*) provides a global navigation system with a low computational cost; (*ii*) provides a robust local navigation system, preventing the robot to stay blocked in a local minima or to collide with unmapped obstacles. The localization module has a central role in this system, since this module provided indispensable conditions to correctly control the robot in order to achieve the proposed tasks. This demonstrates that it was possible through the combination of different methods to obtain a robust control system integrating and exploiting the best characteristics of each one of these methods.

The main contribution of this work was the proposal of a new hybrid control architecture for autonomous mobile robots that was validated through experiments. The HyCAR system proved to be robust and capable of operate in dynamic environments. The HyCAR architecture is well adapted to operate an autonomous mobile robot in changing environments that can include mobile obstacles, and also it is able to determine the robot position in these environments even when the environment did not reflect exactly the internal representation map. In the near future we are planning to integrate our control system into a Nomad 200 robot and then evaluate our control system in real situations.

References

[1] G. Dudek and M. Jenkin. **Computational Principles of Mobile Robotics**. Cambridge University Press, Cambridge, UK. 2000.

[2] E. Gat. **Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robots.** AAAI-92 Proceedings, AAAI Press, 1992.

[3] R. P. Bonasso., et al. **Experiences with an Architecture for Intelligent Reactive Agents.** Journal of Experimental and Theoretical AI, 9(2). 1997.

[4] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. **Monte Carlo localization: Efficient position estimation for mobile robots.** In Proc. of the National Conference on Artificial Intelligence (AAAI). 1999.

[5] Borenstein, J. and Koren, Y. **Real-time Obstacle Avoidance for Fast Mobile Robots.** IEEE Transactions on Systems, Man, and Cybernetics, Vol. 19, No. 5, pp. 1179-1187. 1989.

[6] D. Fox. Markov Localization: **A Probabilistic Framework for Mobile Robot Localization and Navigation.** Institute of Computer Science III, University of Bonn, Germany. Doctoral Thesis. 1998.

[7] T. Cormen, C. Leiserson, R. Rivest**. Introduction to Algorithms**. MIT Electrical Engineering and Computer Science Series. MIT Press. 1990.

[8] Nilsson N. J. **Principles of Artificial Intelligence**. Tioga Plublishing Company. 1980.

[9] Trinity College. **Fire-Fighting Home Robot Contest Website**. Ultima Atualização: Abr 2002. “http://www.trincoll.edu/events/robot/”.

[10] Heinen, Farlei. **Sistema de Controle Híbrido para Robôs Móveis Autônomos**. Master Thesis. UNISINOS University, PIPCA-Applied Computing Master Course. São Leopoldo-RS, Brazil. May 2002.

[11] Dudek, Gregory and Jenkin Mitchel**. Computational Principles of Mobile Robots**. Cambridge University Press. 2000.

[12] Murphy, Robin R. An Introduction to AI Robotics. MIT Press. 2000.