Control architecture for autonomous mobile robots with automated planning techniques

Ezequiel Quintero Ángel García Daniel Borrajo Fernando Fernández

Departamento de Informática. Universidad Carlos III de Madrid

Av. de la Universidad, 30. Leganés (Madrid). Spain

equinter@inf.uc3m.es

April 2010

Abstract

This paper describes the development of an architecture for controlling autonomous mobile robots. The proposed architecture is composed of a set of modules that integrates deliberation with a standard planner, execution, monitoring and replanning. In this paper, we explain the different components of the proposed architecture; we describe the International Planning Competition (IPC) domain *Rovers*, which was used on the experimentation; and we present results from experimens that were conducted with the real robot Pioneer P3DX.

1 Introduction

One of the main goals of robotics is to achieve full autonomy. That is why most of the robot control applications involve task execution management. Hence, in control architectures for intelligent agents such as autonomous mobile robots, it is particularly desirable the use of a deliberation component, based on AI techniques, i.e. automated planning (AP) [9]. Autonomous robots are used in dynamic environments which entail sensor noise, in addition to the uncertainty of task execution on the real world and the difficulty of environment modeling. This justifies the use of deliberative architectures that include an intermediate planning step between sensing and acting. This planning step is based on an environment model - dynamically created from the sensors information -.

Despite the remarkable research activity on AP [9], most of the research is done on theoretical domains of great scientific interest, that are implemented in real life only in the solution of problems within specific contracts with private or public organizations. This is due to the complexity and cost involved on reproducing the problems of scientific interest in reality. However, the field of AP is beginning to receive much attention from various production sectors such as logistics [6], spatial planning [5, 20], critical and control [16] decision systems [4], and even military operations and evacuation [7]. Although robots were designed to operate in the real world, so far most of the planning effort has been made in unreal, deterministic and complete observable environments.

In this paper, we present an architecture for autonomous mobile robots control that follows a sensing-planning-executing cycle. Integrating a deliberation module, based on planning techniques, that gives us domain and problem independence and allows us to use any deliberative planner. The proposed scheme also allows replanning, which means we can generate new plans to reach the final goals even if the initial long-term plan fails. The proposed architecture was tested with a real robot (Pioneer P3DX) on an International Planning Competition (IPC) domain (Rovers) [13]. The domain is inspired on the Mars exploration rover missions, and allows to represent a set of mobile robots that can traverse between

waypoints on a planet, collecting samples and sending data to a lander. Problems involve task and path planning.

The rest of the paper is organized as follows: in section 2, the state of the art is presented. In section 3, the proposed architecture is described. In section 4, the Rovers domain implementation is explained. In section 5, we discuss experiments. And the conclusions are presented in section 6.

2 Related Work

One of the first approaches of AP in robotics was suggested in [3], for the monitored execution of plans using the classic planner STRIPS [8].

In [18] the authors introduced an architecture based on reactive action packages (RAPs), where a plan consists of RAP-defined tasks. In [19] a three-tier control architecture, using RAPs too, is described; the design is similar to the one we propose, but it differs from ours in that, among other things, it focuses on the combination of reactive behaviors. In our case, we focus on domain independent deliberative planning, using the PDDL standard.

Another example of planning techniques applied to robotics, is the use of hierarchical planners in [2]. Also, the authors of [14] proposed an architecture that combines the use of a hierarchical task planner with a path planner. The disadvantage of using this type of planners is that a custom hierarchical task network (HTN) has to be defined for each domain, which augments the definition of primitive actions with ways of combining those actions to achieve specific goals (tasks).

As stated, AP techniques have also been applied to real planetary exploration, with the Rover robots that are currently on Mars. This problem is particularly interesting because it is a case in which having a planning system that provides autonomy is essential. In this case, AP techniques are not used on board the rover, but on ground. The two-layer architecture introduced in [20], has been used in real Rovers and it is closer to our work than other approaches, in the sense that is oriented to interoperability of robot-control software, integrating a decision layer and taking into account high-level autonomy. The authors contributed with a framework for robotic components integration, simplifying the development and reuse of robotics algorithms. But that contribution focused on defining abstracted generic components that have to be adapted to different robot platforms, whereas our approach facilitates interoperability through the use of a stable robot control platform. Moreover, the mentioned research introduced an object oriented infrastructure, primarily for the use of the specific robot platform Rovers; while our approach arises from a more general perspective. Closely related with the former, T-REX [1] is an on-board system for planning and execution applied to the control of an autonomous underwater vehicle in real oceanographic scientific missions. T-REX uses an specific language (NDDL) to describe the domain, and it is based on the notion of partition; planning is done at different abstraction levels by different hierarchical modules, each of one embedding a temporal constraints satisfaction-based planner. Instead, we propose to use a standard planner that reads a PDDL (Planning Domain Definition Language) domain description and is able to control the robot on-board. Thus, we can benefit from improvements on these planners, though we left aside for now the temporal component, which is not crucial yet for the work we are carrying out.

A comparison of behavior-based and planning approaches of miniature mobile robotic agents control is presented in [10]. Recent developments in the plan-based control of autonomous robots are summarized in [11], where the author discusses plan-based high-level and Structured Reactive Controllers (SRCs), among others.

In [17] a control architecture that allows cooperation for autonomous robots, is proposed. The approach combines opportunistic planning and reactive techniques in the low level behaviors. The architecture focused on defining higher level actions as behaviors them-

selves, which was close to the current standard way of defining high level actions in PDDL. The advantage of our approach again is that we can benefit from any PDDL planner.

3 Architecture

In this section, the proposed architecture that integrates planning, execution, monitoring and re-planning, is described. This is the first step of a long term goal which consists on developing a more complex architecture, refining the different modules that are going to be introduced in this section; and adding new ones, such as several machine learning modules, that would merge learning reactive behaviours, learning control and domain knowledge, for the deliberative planner.

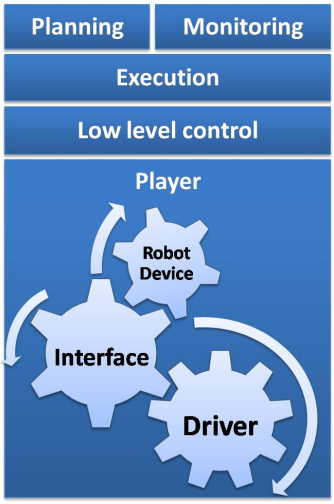

Figure 1: Architecture overview.

After a study, the chosen control platform for commanding the robot was Player [15]. Player is a (TCP) network server that provides an interface for robot device (sensors/actuators) control, designed to be language and platform independent. The Player project includes simulation environments (2D, Player/Stage; and 3D, Player/Gazebo) and other useful tools such as the monitoring application *playerv*. This platform provides official support for several languages and has non-official libraries for many others, so this allows our architecture to be language independent.

3.1 Low Level Control

The low level control layer implements the basic skills. This module receives low level action requests and sends the appropriate commands to the robot actuators, handling the corresponding communication with the Player control platform server. In other words, this module provides a set of basic skills that compose a low-level control server interface that is going to be used by the execution module (3.2).

Because this module has been implemented for mobile robot bases in 2D, it is reusable (with minor changes) for any planar mobile robot and has been tested with the real robot Pioneer P3DX. For this reason, the proposed architecture is robotic-device independent.

3.2 Execution

This module receives a Rovers domain PDDL problem, with the initial state and a set of goals. It first executes the deliberative planner to obtain a high-level solution (3.4). Then, it executes the high-level actions of the generated plan and receives the resulting state of the world after the execution of each action. The execution module communicates with the low-level-control server interface, requesting the execution of the low-level actions that compose each high-level action; and coordinates with the monitoring module (3.3).

As an example, for a problem where there are two waypoints and we aim to take an image of an objective, visible from one of them, a high level plan would be:

```
1 (NAVIGATE P3DX WP0 WP1)
2 (CALIBRATE P3DX LOGITECHSPH OBJ1 WP1)
3 (TAKE_IMAGE P3DX WP1 OBJ1 LOGITECHSPH HIGH_RES)
4 (NAVIGATE P3DX WP1 WP0)
5 (COMMUNICATE_IMAGE_DATA P3DX GRAL OBJ1 HIGH_RES WP0 WP1)
```

The set of basic behaviors corresponding to each of the high-level actions, shown in the previous plan, would be as follows:

1. Navigate: `determine_direction(wp0,w1)`, `turn_to_direction`, `move_forward`.
2. Calibrate: `find_max_blob_in180deg(red)`, `reach_blob`, `bump_center_bumper`.
3. Take image: `capture_image`.
4. Navigate: `determine_direction(wp1,w0)`, `turn_to_direction`, `move_forward`.
5. Communicate image date: `send_email`.

3.3 Monitoring

This module carries out the supervision of the plan execution and the achievement of goals, by receiving information from the execution module. It is in charge of translating the low level (sensor readings) state to high level (domain predicates), for the execution module. Aside from checking whether the goals have been fulfilled, it also determines if replanning is required on each execution step.

The current re-planing policy verifies if the preconditions of the next action to be executed are satisfied. For example, in the action `navigate` (shown below), where the rover `?x` navigates from `?y` waypoint to `?z`, it verifies the following three preconditions (`:precondition`) to determine if we need to replan or not: check that the rover is available (not performing another task), that it can go to the destination waypoint and that this waypoint is visible from the current one.

```
(:action navigate
 :parameters (?x - rover ?y - waypoint ?z - waypoint)
 :precondition (and (can_traverse ?x ?y ?z) (at ?x ?y)
                    (available ?x) (visible ?y ?z)
                )
 :effect (and (not (at ?x ?y)) (at ?x ?z))
)
```

3.4 Planning

This module is in charge of executing the planner and generating the sequence of high level actions that are going to be executed. The planner used is SAYPHI [21]; however, given that we are using the standard PDDL domain description from the IPC, any planner could have been used instead. The planner receives the Rovers PDDL domain and a problem specified in the same format, and returns the corresponding sequence of high level actions that have to be executed in order to achieve the goals.

4 Rovers Domain Implementation

The rovers domain includes the following objects: rovers, its stores, cameras, waypoints, soil/rock sample waypoints, objectives, and a lander-waypoint where the lander spacecfraft is located. Each rover is at a given location, and may carry a sample of a given waypoint or be empty. Taked samples may or may not have been communicated to the lander. Rovers can traverse the path between two connected waypoints; load soil/rock samples of a waypoint; take images of an objective with the calibrated camera; transmit data for a sample or image; and empty the store. The objectives are to communicate a set of sample/image data to the lander.

For example, the PDDL formalization of the problem `wafexample`, which solution is shown in 3.2, is as follows:

```
1  (define (problem wafexample) (:domain Rover)
2  (:objects
3          general - Lander
4          colour high_res low_res - Mode
5          p3dx - Rover
6          p3dxstore - Store
7          wp0 wp1 - Waypoint
8          logitechsph - Camera
9          obj1 - Objective
10 )
11 (:init
12         (channel_free general)
13         (available p3dx)
14         (at p3dx wp0)
15         (store_of p3dxstore p3dx)
16         (empty p3dxstore)
17         (equipped_for_imaging p3dx)
18         (on_board logitechsph p3dx)
19         (supports logitechsph high_res)
20         (calibration_target logitechsph obj0)
21         (visible wp0 wp1)
22         (visible wp1 wp0)
23         (can_traverse p3dx wp0 wp1)
24         (can_traverse p3dx wp1 wp0)
25         (at_lander general wp1)
26         (visible_from obj1 wp1)
27 )
28 (:goal (communicated_image_data obj1 high_res))
29 )
```

In line 1, the `Rover` domain is specified. Between lines 2 and 10, the objects are listed; in

this problem: the rover (`p3dx`) and its store (`p3dxstore`), the camera (`logitechsph`), two waypoints (`wp0` and `wp1`) and one objective (`obj1`). From line 11 to 27, the initial state is described. And in line 28 the goal (communicating an image of the only objective) is specified.

4.1 Domain Mapping

In order to map high-level actions and states into sensoring data and robot low-level behaviours and actuators, we used the following representation mapping:

- Navigate, is mapped to the standard turn and move behaviors, orienting the robot in the direction of the destination waypoint and moving it forward. First, we determine the final orientation of the robot with respect to the destination waypoint. Then, the robots turns into that direction, and moves forward.
- Calibration and rock/soil sample: the calibration and sample actions were represented by a blob tracking action joined with a bumping action. i.e. taking a soil sample is represented as: locating a yellow blob, reaching it and finally bumping that zone with the center front bumper of the robot.
- Take image: a real image is taken with the camera to represent this action.
- The rest of high level actions were not represented as any real physical action. The drop action was not included, because our robot does not have an arm that can take real samples, so there is nothing to drop. The communicate (soil, image and rock) data is intended to be represented as sending an email.

Also, in order to use the Rovers domain, the waypoints were mapped to a grid. The mapping was done as follows: waypoints were defined as an (x, y) pair; the distance between waypoints, on the grid, is d; to differentiate between waypoints, a bounding box of d^2 was established for them; and to simplify, the bounding boxes were adjacents to each other. As stated, the actions `calibrated`, `sample_soil` and `sample_rock` were represented as a joint action of blob-tracking and bumping. Specifically, the blob-tracking was implemented as a simple blob-loop, implemented with the camera panning and a blob detection proxy (provided by Player).

5 Experiments

A Pioneer P3DX, equipped with sonar, bumpers and a motor-base, was used along with a Logitech Sphere cam (with PTZ capabilities). The control software was running in a PC connected via USB to the camera (usb-usb) and the robot (serial-usb). The implemented low-level control module (3.1) provides an interface that allows controlling the following sensors: sonar, motor base, camera (PTZ and blob detection) and bumpers. The grid to which the waypoints were mapped was drawn on the floor, only for external monitoring during the experiments. The robot did not use this drawn grid to orient itself. In Figure 2 the robot is shown on the test environment.

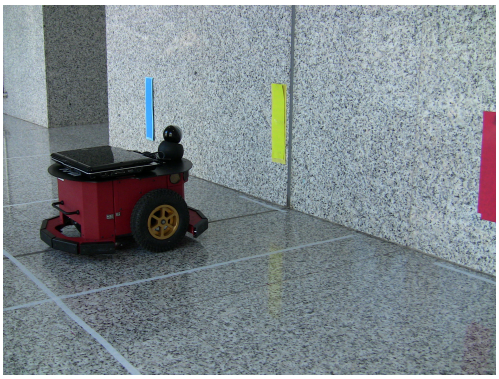

Figure 2: P3DX robot (our rover), on the test environment (our mars)

The objectives of the experiments were: testing the replanning capabilities; comparing the noise introduced by the robot speed versus the certainty generated by slowing it down; and determining the best behavior according

to some metrics, such as number of actions of the plan, execution time, planning time, number of replanning steps, number of failures, ...

We tested the architecture performance with two different problems on the same map (see figure 3). The first one consisted on nine goals, involving thirteen waypoints (six describing the map and ten the exterior) and one rover (the P3DX robot). In figure 3 the rock/soil samples placement and the waypoints from which the objectives were visible, are shown (for the first problem). On that graphic representation, the thick lines between waypoints represent it can not traverse them. The gray waypoints are the external area, which is represented in order to consider the case of the rover going out of the map; for this reason rovers can be *at* them (failure in execution) but not go *into* them; and the shown colors are the ones that were assigned to the blob-tracking behavior.

The second problem, was a simplier one, consisting on communicating one sample of rock data, one sample of soil data and one image data.

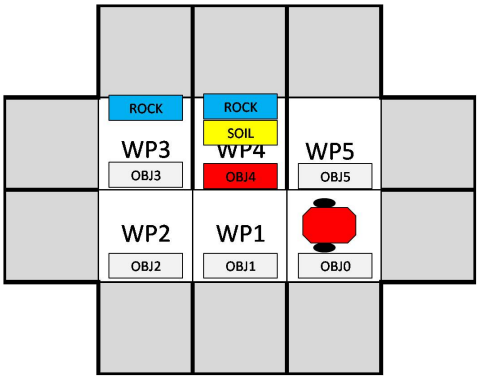

Figure 3: Graphical representation of the first problem.

5.1 Results

The first problem was tested with and without obstacles. And in each case, the robot was tested with two different speeds. So, two types of tests were conducted: the *slow* experiments, that were done with a forward-speed of 0.2m/s and a turn speed of 0.2rads/s; and the *fast* experiments, that were performed with a forward-speed of 0.4m/s and a turn speed of 0.35rads/s. In the case of the obstacles experiments, a second remote controled robot was crossed in the middle of the rover trajectory five times, during the execution of different navigate actions. The moving obstacle momentarily crossed the path of the rover to simulate the case of another rover exploring, in order to test the replanning capabilities.

Table 1 summarizes the experiments results for the first problem. The following metrics (measure column) were established: total executed actions; number of replanning steps performed; total execution time, in minutes; and total planning and replanning time, in seconds.

	No Obstacles		Obstacles	
Measure	Slow	Fast	Slow	Fast
Initial actions	59			
Total actions	59	59	71	74
Replanning	0	3	20	20
Total ex time	17m	14.7m	15m	16.3m
Planning time	0.12s	0.7s	0.12s	1s
Replanning time	0s	0.18s	2s	2.3s

Table 1: Summary of experiments results for the first problem. Presence vs. absence of obstacles, with different speeds.

Without obstacles, in both speed configurations, the total high-level executed actions is the same as the original plan. Given that higher speeds introduce more easily failures in execution, in the fast executions, replanning was needed, while in the slow configuration it was not needed. Despite the fact that replanning was necessary because of the noise acumulation, caused by the turn and forward speedup, the time of execution was still reduced.

With obstacles, in both speed configurations, the total high-level executed actions (71 in the slow configuration and 74 on the fast one) is significantly increased with respect to the number of initial planned actions (59). The number of replanning steps needed were the same on both configurations. Thus, the

speedup did not affect significantly the total execution time. But, in the case of obstacles, we did not improve with the increase of speed.

Unlike the first problem, the second one was executed only without obstacles because the objective was to observe the behavior of the architecture in smaller problems. The table 2 summarizes the experiments results for the executions of the second problem. The same metrics were used.

Measure	Slow	Fast
Initial actions	26	
Total actions	26	22
Replanning	0	5
Total ex time	7.8m	4.8m
Planning time	0.06s	0.05s
Replanning time	0s	0.38s

Table 2: Summary of experiments results for the second problem. Simple problem with different speeds.

The solution of this problem consisted on 26 actions (initial plan actions), which is less than a half of the previous problem solution. The difference between the total executed actions (second raw) is caused because in some of the *fast* executions, the accumulation of noise in the odometry made the robot accidentally go into waypoints that shortened the path (i.e. moving forward two way points at a time). As it is a small problem, only 5 replanning steps were performed on the *fast* configuration; and no replanning was done on the *slow* one. On this problem the total execution time was reduced to half with the *fast* configuration; so the increase of speed was totally worth it, taking into account the insignificant amount of time consumed by the replanning steps that the speedup noise caused.

In both problems the resulting replanning time was insignificant with respect to the tasks execution time, so there would be no need to include plan adaptation strategies at this point.

6 Conclusions

In this paper we have presented an architecture for autonomous mobile robots control, that integrates automated planning (AP) techniques, execution, monitoring and replanning.

As the chosen robotic-control platform (Player) provides support for many programming languages, the proposed architecture is language independent (adapting the low level control component).

Having implemented the low level control module with Player, also makes the architecture platform-independent, becasue the proposed control code is valid for any planar mobile robot bases in 2D (with minor changes).

The main advantage of applying AP techniques is that we can use a standard planner that reads a PDDL (Planning Domain Definition Language) domain description. This allows us to use the architecture in different domains.

With this work we have contributed to a long-term goal by laying the basis for developing a more ambitious architecture that hopefully will contribute on achiving full autonomy on mobile robotics.

7 Acknowledgements

This work has been partially supported by the Spanish MICINN under projects TIN2008-06701-C03-03, TRA-2009-008 and the regional projects CCG08-UC3M/TIC-4141.

References

[1] C. McGann, F. Py, K. Rajan, J. P. Ryan and R. Henthorn. "Adaptive Control for Autonomous Underwater Vehicles". AAAI, 2008.

[2] B. Morisset and M. Ghallab. "Learning how to combine sensory-motor modalities for a robust behavior". Advances in Plan-Based Control of Robotic Agents, pp. 1-24, 2002.

[3] R. E. Fikes, "Monitored Execution of Robot Plans Produced by STRIPS".TFIP Congress, August 23-28, 1971.

[4] R. R. Penner and E. S. Steinmetz. "Automated support for human mixed initiative decision and control". 42nd IEEE Conf. on Decision and Control, pp 3549-3554, 2003.

[5] M. Rodríguez-Moreno, D. Borrajo, and D. Meziat. "An AI planning-based tool for scheduling satellite nominal operations". AI Magazine, 25(4):9-27, 2004.

[6] A. Tate, B. Drabble and J. Dalton. "O-Plan: A Knowledge-based Planner and Its Application to Logistics". University of Edinburgh, Artificial Intelligence Applications Institute, 1996.

[7] D. E. Wilkins and R. V. Desimone. "Intelligent scheduling, chapter Applying an AI plannet to military operations planning". Morgan Kaufmann, 1978.

[8] R. Fikes and N. Nilsson. "STRIPS: A new approach to the application of theorem proving to problem solving". Artificial Intelligence, 2 (3-4): 189-208, 1971.

[9] M. Ghallab, D. Nau, P. Traverso. "Automated planning: theory and practice". Morgan Kaufmann, May 2004.

[10] S. Slusny, R. Neruda, P. Vidnerova. "Comparison of behavior-based and planning techniques on the small robot maze exploration problem ". Neural Networks, Vol. 23, Iss. 4, 2010, pp. 560-567.

[11] M. Beetz. "Towards Comprehensive Computational Models for Plan-Based Control of Autonomous Robots". Mechanizing Mathematical Reasoning, 2005. pp. 514-527.

[12] C. McGann, F. Py, K. Rajan and A. Garcia. "Integrated Planning and Execution for Robotic Exploration". International Workshop on Hybrid Control of Autonomous Systems. July 2009.

[13] Internation Planning Competition (IPC-3). Hosted at the Artificial Intelligence Planning and Scheduling Conference, 2002. http://planning.cis.strath.ac.uk/competition/

[14] J. Guitton, J. Farges, and R. Chatila. "A planning architecture for mobile robotics". AIP Conf. Proc. 1019, 162 (2008).

[15] B. Gerkey, R. Vaughan and A. Howard. "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems". In Proc. of the 11th International Conference on Advanced Robotics (ICAR 2003), pp. 317-323, Coimbra, Portugal, June 2003.

[16] L. Castillo, J. Fdez.-Olivares, O. García-Pérez, F. Palao. "SIADEX. An integrated planning framework for crisis action planning". ICAPS 2005, Software Demonstrations Track.

[17] V. Matellán and D. Borrajo. "ABC^2 An Agenda Based Multi-Agent Model for Robots Control and Cooperation". Journal of Intelligent and Robotic Systems, n. 1, vol. 32, pp. 93-114. 2001.

[18] Firby, R.J. "Adaptive Execution in Complex Dynamic worlds". Ph.D. thesis, Yale University, 1989.

[19] R. Bonasso, D. Kortenkamp and D. Miller, M. Slack. "Experiences with an Architecture for Intelligent, Reactive Agents". Journal of Experimental and Theoretical AI, 1995, vol. 9, pp. 237-256.

[20] I.A. Nesnas, A. Wright, M. Bajracharya, R. Simmons, T. Estlin, Won Soo Kim, "CLARAty: An Architecture for Reusable Robotic Software". SPIE Aerosense Conf. 2003, 5083(1):253-264.

[21] T. de la Rosa, A. García and D. Borrajo. "Case-Based Recommendation for Node Ordering in Planning". Proc. of the 20th International FLAIRS Conference, 2007. AAAI Press.