Ten Autonomous Mobile Robots (and even more) in a Route Network Like Environment*

L. Aguilar, R. Alami, S. Fleury, M. Herrb, F. Ingrand, F. Robert
LAAS / CNRS,
7, Avenue du Colonel Roche - 31077 Toulouse - France

Abstract: This paper presents an implemented system which allows to run a fleet of autonomous mobile robots in a route network with a very limited centralized activity. The robots are endowed with all the necessary ingredients for planning and executing navigation missions in a multi-robot context.

Multi-robot cooperation is based on a generic paradigm called *Plan-Merging Paradigm*, where robots incrementally merge their plans into a set of already coordinated plans.

The robot architecture is derived from the generic architecture developed at LAAS. A 3D graphic environment system allows to display a complete system composed of a dozen (or more) robots, each running on an independent workstation. An example is presented together with numerical results on the system behavior.

1 Introduction

We describe, in this paper, a system which allows to run a fleet of autonomous mobile robots in a route network with a very limited centralized activity. The robots are endowed with all the necessary ingredients for planning and executing navigation and load handling missions - expressed at a very high level of abstraction - as well as for multi-robot cooperation.

In order to implement a robust and efficient multi-robot cooperation capability, we use a generic paradigm called *Plan-Merging Paradigm*, where robots incrementally merge their plans into a set of already coordinated plans. This is done through exchange of information about their current state and their future actions.

The robot architecture is derived from the generic architecture developed at LAAS. The software tools we use allow us to run the robot software under VxWorks on real robots (from the Hilare family) as well as on Unix workstations emulating low level robot primitives.

A test and evaluation environment has been developed which includes a 3D graphic system to display a complete fleet of robots (a dozen or even more) in a structured environment. This testbed provides mechanisms such that each robot is fully functional and can be ran on an independent Unix workstation.

Typical runs on this testbed are presented together with numerical results on the system behavior.

*This paper will appear in the proceedings of IROS 95.

2 Related Work

Various methods have been proposed which deal with multi-robot systems. Besides, the term "cooperation" has been used in several contexts with different meanings. We will not consider here cooperation schemes at servo level nor contributions dealing with "intelligent groups" of simple robots.

A thorough analysis of the literature is still to be made; we simply mention here some representative contributions which involve an effective cooperation (at plan or program level) between several robots.

Several approaches have been proposed, such as generation of trajectories without collision (e.g. [7, 21]), traffic rules [11, 14], dynamic adaptation of trajectories [10], negotiation for dynamic task allocation [15, 4], and synchronization by programming [17, 23, 22].

Concerning more particularly multi-robot motion planning methods, numerous contributions have been made which are generally based on a central planner, specially designed to cope with the intrinsic complexity of the problem [20, 8, 18]. While these methods are not complete or cannot be used for a "large" number of robots (more than 3), recent techniques based on randomized search in the Global Configuration Space [19], allow most often to obtain a solution in a reasonable time, even though, in the worst case, they fall into the unavoidable problem complexity.

Most of these contributions are based on a predefined set of situations or on task specific planners. We claim that our Plan-Merging Paradigm is a generic framework which can be applied in different contexts, using different planners (action planners as well as motion planners). It has some clean properties (and clear limitations) which should allow, depending on the application context, to provide a coherent behavior of the global system without having to encode explicitly all situations that may be encountered. Another advantage of our method is that it allows, most of the time, to solve a conflict without using a full multi-robot planner and even without stopping the execution of the other robots.

3 A fleet of Autonomous Mobile Robots

The problem consists in devising a system which allows to run a large number of autonomous mobile robots in a route network composed of lanes, crossings and open areas. Typical applications of this problem are cargo transfer as dealt with in the MARTHA project[1] which requires the development of a large fleet of autonomous mobile robots for the transportation of containers in harbors, airports and railway station environments.

3.1 Autonomous Robots need Decentralized Cooperation

The system is composed of a Central Station and a set of autonomous mobiles robots.

The current state of the art allows a substantial level of autonomy for a unique mobile robot in a structured environment: autonomous motion planning, localization based on the perception of known features in the environment, obstacle avoidance, and so on.

However, if there are several robots, and if all necessary synchronizations are performed at a central system, this will result in a very high activity at the central system and a drastic limitation of the range of the plans allocated to the robots.

The *Plan-Merging Paradigm* we propose is well suited to such applications as it allows to run a great number of robots, locally dealing with conflicts while maintaining a global coherence of the system. Indeed, it limits the role of the central system to the assignment of tasks and routes to the robots (without specifying any trajectory or any synchronization between robots) taking only into account global traffic constraints.

3.2 Mission specification

We now present the Environment Model used for mission elaboration and motion planning and the role of the Central Station in the mission elaboration.

The Environment Model: In order to allow efficient and incremental plan merging, we have decomposed the route network into smaller entities called "cells" or "spatial resources" which will be used as a basis for dealing with local conflicts.

Basically, the robots navigate through an oriented graph of cells. Lanes and crossings are composed of a set of connected cells, while areas consist of only one cell.

Thus, the environment model, which is provided to each robot, mainly contains topological and geometric information (Figures 1 and 2):

• A network describing the connections of areas and crossings by oriented lanes. This is the only information used by the Central Station to elaborate routes for robots.

• A lower level topological description (cell level). The graph of cells is oriented, in order to provide a nominal (but not exclusive) direction for lanes and crossings use.

• A geometrical description of cells (polygonal regions).

• Additional information concerning landmarks (for re-localization), station descriptions for docking and load handling actions.

The role of the Central Station: The central station is in charge of producing high level plans to load/unload ships, trains or planes. A plan is produced taking into account the route network topology as well as the availability of such or such robot. However, it does not further specify the sequence of robots going through a crossing (this decision is left to the robot locally concerned), nor does it require the robot to remain on the specified lanes (in case it needs to move away from an unexpected obstacle).

[1]MARTHA: European ESPRIT Project No 6668. "Multiple Autonomous Robots for Transport and Handling Applications"

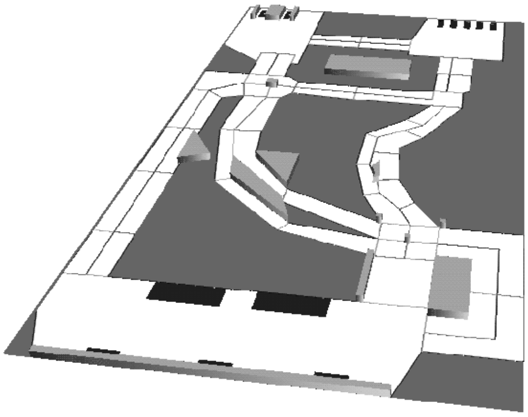

Figure 1: An environment (real size: 550 m x 250 m)

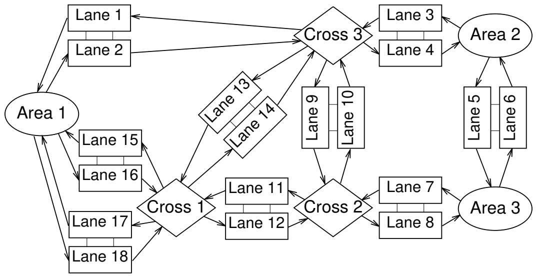

Figure 2: The associated topological graph

3.3 A Plan-Merging Protocol for Multi-Robot Navigation

The cooperation scheme we use is based on a general paradigm, called *Plan-Merging Paradigm* [2], where robots incrementally merge their plans into a set of already coordinated plans. This is done through exchange of information about their current state and their future actions.

For the case of a number of mobile robots in a route network environment, we have devised a specific *Plan-Merging Protocol* (PMO) based on spatial resource allocation (see [3]). It is an instance of the general protocol described in [2], but in this context, *Plan-Merging Operation* is done for a limited list of required resources: a set of cells which will be traversed during the plan to merge. The robot broadcasts the set or required cells, receives back the set of coordination plans from other robots which have already planned to use some of the mentioned cells, and then tries to perfrom a plan insertion which ensures that the union of the considered plans is a directed acyclic graph.

Due to space limitations, we will not describe in more detail this protocol. A full description may be found in [2].

One of the most interesting attributes of this protocol is that it allows several PMOs to be performed simultaneously if they involve disjunctive resource sets. This is particularly useful when there are several local conflicts at the same time.

Plan-Merging for cell occupation: In most situations, robot navigation and the associated Plan-merging procedure are performed by trying to maintain each cell of the environment occupied by at most one robot. This allows the robots to plan their trajectories independently, to compute the set of cells they will cross and to perform Plan-Merging at cell allocation level.

In order not to constrain unnecessarily the other robots, the allocation strategy makes a robot allocate one cell ahead when it moves along lanes, while it allocates all the cells necessary to traverse and leave crossings.

When reasoning about cells is not sufficient While, most of the time, the robots may restrict their cooperation to cells allocation, there are situations where this is not enough. This happens when they have to cross non-structured areas (§3.2) or when an unexpected obstacle, encountered in a lane or in a crossing, forces a set of robots to maneuver simultaneously in a set of cells. In such situations, a more detailed cooperation (using the same protocol but a different planner: the motion planner) takes place allowing robots to coordinate their actions at trajectory level. Thus, we have a hierarchy of PMOs

• first, at the cell level, based on resource (cells) allocation

• then, depending on the context, at trajectory level: motion planning in a set of common cells determined by the first level

This hierarchy authorizes a "light" cooperation, when possible, and a more detailed one, when necessary.

4 The Robot Control System

The architecture of the Robot Control System (RCS) is directly derived from the generic control architecture for autonomous robots developed at LAAS [5, 1, 9]. It is composed of a Decisional Level and a Functional Level.

4.1 The Decisional Level

The Decisional Level (also called the Robot Supervisor (RS)) consists of three layers, corresponding to a hierarchical decomposition of planning and control activities (Figure 3): the Mission layer, the Coordination layer and the Execution layer.

The first two layers are themselves composed of a planning and a a supervision process. All processes (five) run in parallel and satisfy different response time constraints [1, 13]

1. The Mission layer: From time to time, the Central Station sends a new mission to the robot (Figure 4 illustrates a typical mission). The Mission layer deals with mission refinement and control. Mission refinement is performed through the use and context dependent instantiation of a library of "predefined plans" or "plans skeletons". A mission is first refined as if

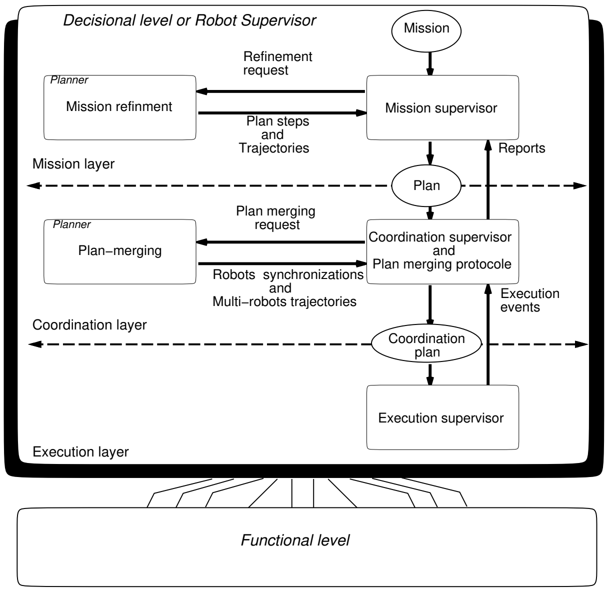

Figure 3: The robot supervisor architecture

the robot was alone. The resulting plan is a sequence of actions (including planned trajectories) annotated with cell entry and exit monitoring operations which will be used to maintain the robot execution state and to synchronize its actions with other robots. The Mission Supervisor is in charge of controlling the execution of such plans. If a plan fails to achieve a particular goal, alternative plans are refined and attempted.

```
(mission (.
    (action 1 (goto (station 1)) (using (lane 10)))
    (action 2 (dock))
    (action 3 (put-down))
    (action 4 (un-dock))
    (action 5 (goto (station 3)) (using (lane 12) (lane 8)))
    (action 6 (dock))
    (action 7 (pick-up (container 5)))
    (action 8 (un-dock))
    (action 5 (goto (end-lane 0)) (using (lane 9) (lane 0))) .))
```

Figure 4: A mission example

2. The Coordination layer: produces and controls "coordination plans". It performs Plan-merging operations and manages the interactions with the other robots (exchanging coordination plans and events). Indeed, the plans produced by the Mission layer are incrementally validated in a multi-robot context by the Coordination layer through the use of a Plan-merging protocol. The result is a "coordination plan" which specifies all trajectories and actions to be executed, together with all events to be monitored and sent to another robot or to be awaited from another robot (Figure 5).

Note that an execution failure reported by a robot from which an event is awaited will cause a new Plan-merging operation to be performed.

3. The Execution layer is in charge of the in-

```
(coordination-plan (.
        (exec-plan 1 (report (begin-action 1)))
        (exec-plan 2 (wait-exec-event r3 9))
        (exec-plan 4 (monitor (entry (cell 4))))
        (exec-plan 8 (exec-traj 0))
        (exec-plan 3 (wait-exec-event r7 48))
        (exec-plan 5 (monitor (entry (cell 5))))
        (exec-plan 6 (monitor (exit (cell 14))
                       (signal-exec-event r4 17)))
        (exec-plan 7 (monitor (exit (cell 4))))
        (exec-plan 9 (exec-traj 1))
        (exec-plan 10 (exec-traj 2)) .))
```

Figure 5: Example of coordination plan

terpretation and the execution of coordination plan. As a result, it is responsible of most interactions with the functional level. Besides all actions and monitors included in the plan, it also monitors and reacts to a number of critical events, such as unexpected obstacles in its path, or its own status (battery or fuel level), failure reports from the different modules, as well messages sent by the other robots (infos about synchro events or plan failures).

4.2 The Functional Level

The functional level implements all the robot basic capabilities in sensing, acting and computing. These functionalities are grouped according to data or resource sharing, and integrated into *Modules*.

Beside real time capabilities to insure closed-loop control and reactivity, this level fulfills several conditions towards the decisional level: bounded response time to requests, observability and programmability.

All modules have the same structure (see [9]). A module is composed of a module controller and an execution level.

At the execution level, the implemented functions (i.e. embedded algorithms) are interruptible by the module controller. They can also abort by themselves if the execution cannot be achieved (internal failure, failure of a server at a lower level...). A specific report is then returned to the client.

A formal language is used to describe a module including its behavior, its interfaces and its connections with others modules. We have built an automatic modules compiler that uses a formal module description and user supplied functions to automatically produce the modules.

Figure 6 shows the functional level, for the presented application, including 7 modules, their client/-server relations and 3 exported data (posters). Each module will be described later on.

The Robot Supervisor is a client of all the modules of the functional level. It manages itself the poster named `ENVIRONMENT` which exports the topological and geometrical model of the environment (cf §3.2).

4.2.1 The Motion Planner Module

Each robot is equipped with an independent Motion Planner composed of a Topological Planner, a Geometrical Planner and Multi-robot Scheduler. It is

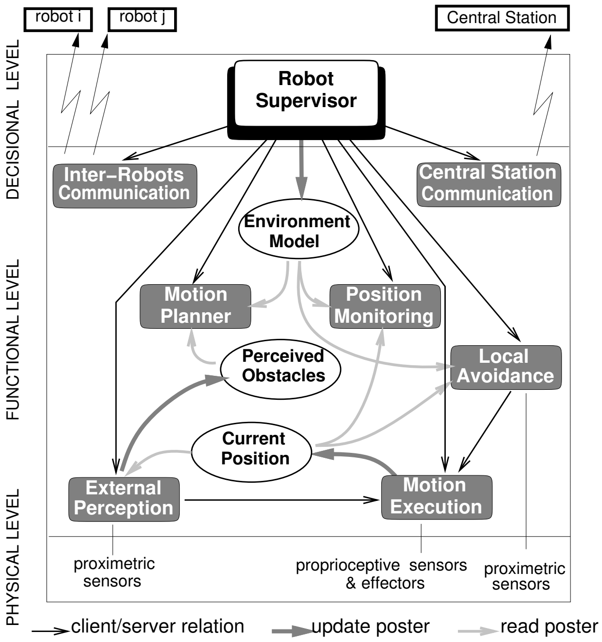

Figure 6: Architecture and interactions of the functional level

used in order to compute not only feasible trajectories but also synchronization events between different robot trajectories.

The Topological Planner: performs a search in the graph of cells in order to determine the set of cells to be used for a given motion task. The selection of cells may be done in two modes: a *Local obstacle Avoidance* mode that selects only the cells which correspond to the nominal way of traversing lanes or crossings, and an *Extended obstacle Avoidance* mode which is invoked, for example, when a major obstacle forces a robot to leave its current lane and to use cells belonging to a "parallel" lane.

The Geometrical Planner: computes a non-holonomic path between an initial position and a goal position. When used in a "multi-robot" mode, it produces a path which avoids the last positions of the other robots. It is based on techniques similar to those described in [6].

The Multi-robot Scheduler: determines, along each trajectory, the positions where a robot should update its set of occupied resources, the positions where it should signal a *trajectory synchronization event* to another robot, and the positions where it should stop and wait for synchronization. Figure 7 illustrates trajectory synchronizations: *Ri-Wj* stands for a position where robot *ri* should stop and wait that robot *rj* has passed position *Rj-Si*.

4.2.2 The Motion Execution Module

This module has a permanent activity which consists in **position computation** from proprioceptive sensors (odometers, gyro) and in the feedback control on the robot position (x,y). The current position is

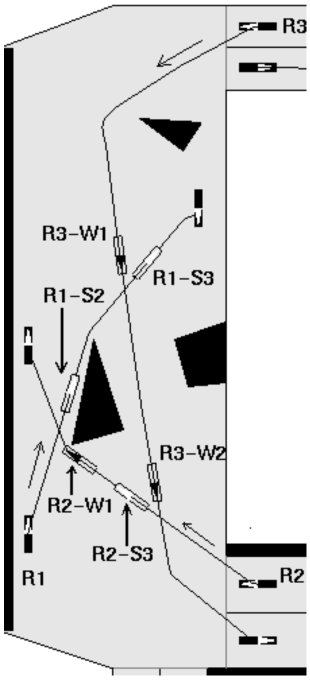

Figure 7: The result of a PMO at trajectory level (in an area)

exported in the `POSITION` poster Besides, it executes trajectories composed of a sequence of segments and arcs of circle, and a polygonal bounding area in which the robot should remain (cell, lane). The trajectories are smoothed with clothoids.

It is also able, for multi-robot coordination purposes, to monitor the curvilinear abscissa while executing a motion (cf §4.2.1).

4.2.3 The Local Obstacle Avoidance Module

The Local Avoidance module monitors obstacles with range sensors (ultra-sonic sonars or laser range finder) and filters the trajectories before transmitting them to the Motion Execution module.

According to the urgency, the robot can be stopped or the trajectory can be slightly modified in order to avoid an obstacle. However, the trajectory should remain in a bounding area specified by the Robot Supervisor. If the robot is blocked, the motion requests are ended. The Robot Supervisor will then produce a new planned trajectory taking into account the new obstacles.

4.2.4 The External Perception Module

The role of this module is twofold[16]: (1) updating the robot position using exteroceptive data (range sensors) and performing landmarks based re-localization; and (2) building and maintaining a local obstacles map which may be provided, upon request, to the Motion Planner, through the `PERCEIVED OBSTACLES` poster.

4.2.5 The Position Monitoring Module

This module allows to maintain the set of resources (cells, lanes, areas) occupied by the robot and to monitor the entry and the exit of these resources. This information is necessary for the Plan-Merging activity performed by the Robot Supervisor,

4.2.6 The External Communication Modules

Tne communications with the Central Station and with the others robots are achieved by two distinct modules called Inter-Robot Communication (IRC) and Central Station Communication (CSC).

A message between robots can be dedicated to one specific robot or broadcasted to all robots in its vicinity (the IRC is assumed to have a limited range).

5 Implementation of a Realistic Testbed

We have developed a complete robot control system which includes all the features described above.

The robot supervisor is coded using a procedural reasoning system: C-PRS [12, 13]. The target implementation runs on a multi-processor VME rack, under the VxWorks real-time system.

For testing and demonstration purposes we have built an emulation of the communication and multi-tasking primitives of the real-time system, that allows us to run all elements of a robot (the supervisor and all functional modules) on a Unix workstation as a set of Unix processes.

The motion execution is simulated at the lowest level by sending the elementary motion controls into the perception sub-system. Radio communications between robots and with the central station are simulated on the Ethernet network (Figure 8).

A 3-D graphic server has been built to visualize the motions and the load operations performed by the robots in their environment (Figures 1,9-12). It receives positions updates from the motion execution modules of all the robots, each running on its own workstation.

Experiments have been run successfully on a dozen of workstations (each workstation running a complete robot simulator, Figure 8). Some results are presented in the next sections.

We have also tested the implementation on two real mobile robots, and a third robot is under construction to run real experiments in a real environment.

6 Results

We shall now illustrate the plan-merging paradigm and its capabilities with some sequences from our experimentation with simulated robots in a route network with open areas. The first example presents a PMO at a crossing, the second, a PMO in an open area.

An example of PMOs at a crossing

Note that as presented in section 3.3, PMO performed at crossing allocates all the cells necessary to traverse and leave the crossing.

Crossing, Step 1 (Figure 9):

This snapshot has been edited to exhibit the routes

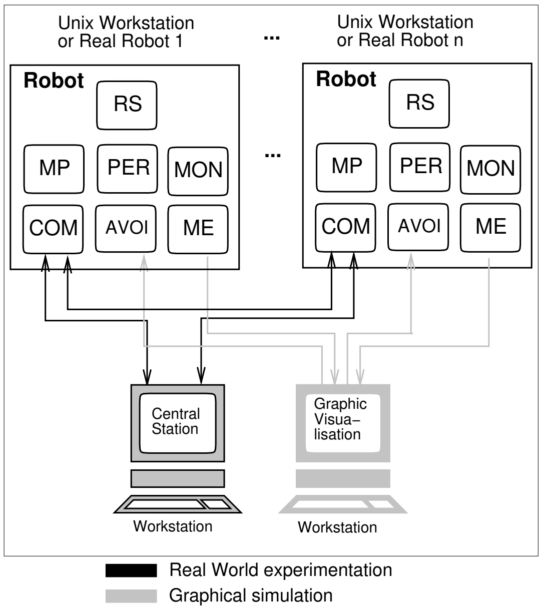

Figure 8: n Unix workstations implementing n robots

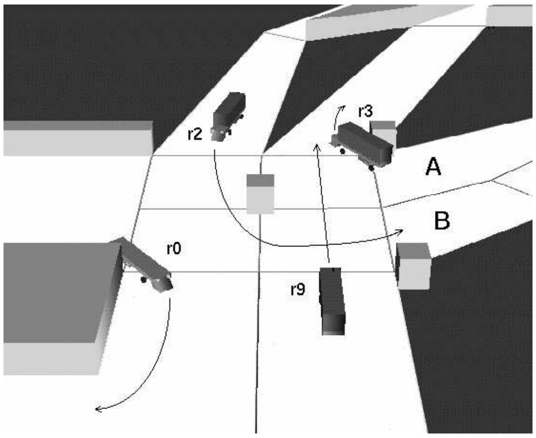

Figure 9: Crossing (Step 1)

that the robots must follow.
• Robot $r3$, coming from position A, and $r0$ have disjunctive list of resources. Therefore, they can perform PMOs in parallel and traverse the crossing without any synchronization.
• Robot $r9$ follows $r3$, but its PMO fails (because $r3$ has not yet planned an action to free the cell that $r9$ must allocate to exit the crossing). As a result $r9$ must wait a *planning event* from $r3$ (i.e. a new PMO).
• Robot $r2$, which wants to go to position B, finds itself in a situation where it can merge its plan with the other robots plans.

Crossing, Step 2 (Figure 10):
• Robot $r2$ traverses the crossing, after an execution synchronization with $r0$ (it must wait until $r0$ leaves the lower left cell of the crossing before entering it).
• Robot $r9$ has received the awaited *planning event* from $r3$ (which has now planned an action to exit its

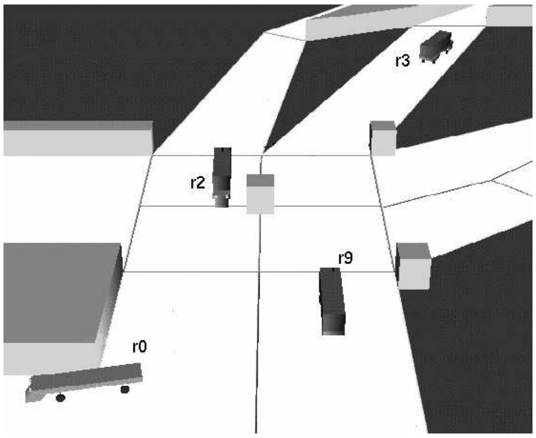

Figure 10: Crossing (Step 2)

current cell) and its PMO succeeds, but it must synchronize with $r2$ and $r3$.

Crossing, Step 3:
• Robot $r2$ frees the crossing cells and sends the corresponding *execution event* to $r9$.
• Robot $r9$ can now traverse the crossing.

An example of PMOs in an Open Area

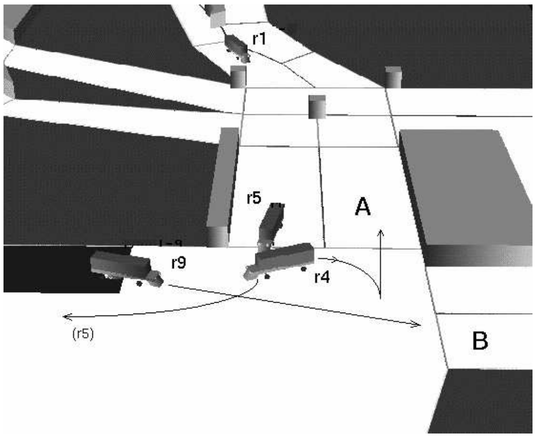

Figure 11: Open Area (Step 1)

As mentioned earlier, open areas are not exclusive resources. In our example, $r9$ goes to position B, $r4$ moves backward from a station in the area to position A and $r5$ enters the area to dock at a station.

Open Area, Step 1 (Figure 11):
The three robots have perform theirs PMOs, first at "cell level", then at "trajectory level", in the following order: $r4$, $r9$ and $r5$. $r9$ and $r5$ (which has entered the area) are both waiting until $r4$ passes a computed curvilinear abscissa. Each of them waits at a particular position (*R9-W4* and *R5-W4*) that $r4$ passes respectively *R4-S9* and *R4-S5*.

Open Area, Step 2 (Figure 12):
Robot $r5$ goes from *R5-W4* to *R5-W9* and waits for an *execution event* from $r9$ (which will be sent as soon

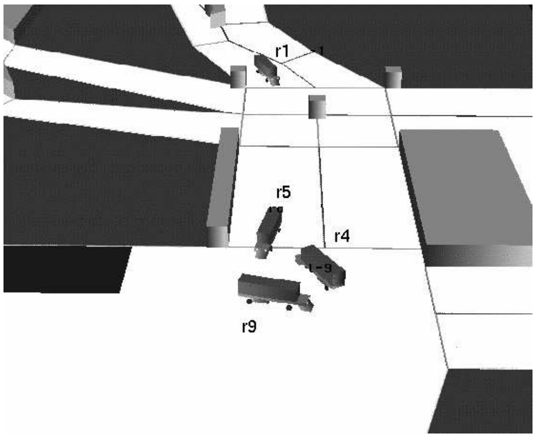

Figure 12: Open Area (Step 2)

as $r9$ passes *R9-W5*).

Open Area, Step 3:
$r9$ has sent the execution even and all the robot can now continue their mission.

The two examples, presented above, exhibit the following properties of the PMO:

• Planning and execution is done in parallel.

• Several robots may use the crossing simultaneously (r0 and r3).

• The example exhibits the two types of synchronization: *execution events* ($r4$, $r5$ and $r9$ in the area example), and *planning events* ($r9$ and $r3$ in the crossing example).

• Each robot produces and merges its plans iteratively, and the global plan for the use of the crossing is incrementally built through several PMOs performed by various robots.

• It is not a first arrived first served execution. In the crossing example $r9$ arrived second, was blocked by $r3$, but did not block the crossing, letting $r2$ enter the crossing before itself.

6.1 A Complete Mission

The simulated robots were able to achieve navigation missions, performing hundreds of PMOs and solving local conflicts. Motion planning and PMOs were sufficiently efficient to allow the robots to elaborate and merge their plans without stopping unless necessary.

From a set of experiments ran on the environment presented on Figure 1 where ten emulated robots execute each a representative mission (such as the one presented in section 4), we collected the following data: 379 messages were exchanged:

155	broadcasts for plan merging
56	plans exchanged
65	messages for execution synchronization.

During all these missions several types of conflicts have been encountered and solved:

5	conflicts for simultaneous PMO for common resources
11	conflicts for PMO queue management
16	messages to release the PMO token ($16 = 11 + 5$)
36	messages to update the deadlock detection graph.

Depending on the length and difficulty of their missions, individual robots have produced from 20 to 90 messages. The average message length is 100 bytes (without any optimization). The experiments lasted around 15 minutes for 41 k-bytes exchanged. A very simple improvement (shortening the keywords included in the messages) would result in a 20 k-bytes exchanged for the whole mission. These results show that the bandwidth required by the PMO protocol is compatible with the low baud rate provided by the communication interface.

Moreover, the robots were able to run at an average speed of 3m/s without ever stopping, unless required by a synchronization.

7 Conclusion and future work

The system described in this paper presents many original contributions to the field of research on autonomous mobile robots. To our knowledge, it is the first time such a large fleet of autonomous robots is put together to execute high level missions given by a central station.

Our experimentation using large number of emulated robots and with real robots (from the Hilare family) has shown the feasibility and the embarkability of our solution.

The Plan-Merging Paradigm we propose has the following properties;

1. It "fills the gap" between centralized, very high level planning and distributed execution by a set of autonomous robots in a dynamic environment.

2. It makes possible for each robot to produce a coordination plan which is compatible with all plans executed by other robots.

3. No system is required to maintain the global state and the global plan permanently. Instead, each robot updates it from time to time by executing a PMO.

4. The PMO is safe, because it is robust to plan execution failures and allows to detect deadlocks.

The current implementation in simulation has shown that the protocol works and allows for far more than ten robots to cooperate. In fact, considering the locality of the conflict resolution, i.e. the ability of robots in a group to coordinate their plans and actions without disturbing the rest of the fleet, one can easily see that this protocol can scale to a much larger number of robots (hundreds). This protocol allowed us to make a large number of autonomous robots behave coherently and efficiently without creating a burden on the central system activity.

Our future work will concentrate on developing new cooperation schemes by embedding a multi-robot planning activity inside a PMO particularly in the case of motion planning.

References

[1] R. Alami, R. Chatila, B. Espiau. Designing an intellingent control architecture for autonomous robots. *ICAR'93*, Tokyo (Japan), Nov. 1993.

[2] R. Alami, F. Robert, F. Ingrand, S. Suzuki. A paradigm for plan-merging and its use for multi-robot cooperation. *IEEE Int. Conf. on Systems, Man, and Cybernetics*, San Antonio (USA), 1994.

[3] R. Alami, F. Robert, F. Ingrand, S. Suzuki. Multi-robot cooperation through incremental plan-merging. *IEEE ICRA'95*, Nagoya (Japan), May 1995.

[4] H. Asama, K. Ozaki, et al. Negotiation between multiple mobile robots and an environment manager. *ICAR'91*, Pisa (Italy), June 1991.

[5] R. Chatila, R. Alami, B. Degallaix, H. Laruelle. Integrated planning and execution control of autonomous robot actions. *IEEE ICRA'92*, Nice (France), May 1992.

[6] J. C. Latombe. A Fast Path Planner for a Car-Like Indoor Mobile Robot. In *Ninth National Conf. on Artificial Intelligence*, 1991.

[7] H. Chu, H.A. EiMaraghy. Real-time multi-robot path planner based on a heuristic approach. *IEEE ICRA'92*, Nice (France), May 1992.

[8] M. Erdmann, T. Lozano-Perez. On multiple moving objects. *IEEE ICRA'86*, San Francisco (USA), April 1986.

[9] S. Fleury, M. Herrb, R. Chatila. Design of a modular architecture for autonomous robot. *IEEE ICRA'94*, San Diego (USA), 1994.

[10] T. Fraichard, C. Laugier. Path-velocity decomposition revisited and applied to dynamic trajectory planning. *IEEE Transactions on Robotics and Automation*, pages 40–45, 1993.

[11] D.D. Grossman. Traffic control of multiple robot vehicles. *IEEE Journal of Robotics and Automation*, 4(5):491–497, Oct. 1988.

[12] F. Ingrand, M. Georgeff, A. Rao. An Architecture for Real-Time Reasoning and System Control. *IEEE Expert, Knowledge-Based Diagnosis in Process Engineering*, 7(6):34–44, Dec. 1992.

[13] F. Ingrand, R. Chatila, R. Alami, F. Robert Embedded Control of Autonomous Robots using Procedural Reasoning. *ICAR 95*, Sant Feliu de Guixols, Spain, Oct. 95.

[14] S. Kato, S. Nishiyama, J. Takeno. Coordinating mobile robots by applying traffic rules. *IEEE IROS'92*, Raleigh (USA), July 1992.

[15] C. Le Pape. A combination of centralized and distributed methods for multi-agent planning and scheduling. *IEEE ICRA'90*, Cincinnati (USA), May 1990.

[16] P. Moutarlier, R. Chatila. Incremental free-space modelling from uncertain data by an autonomous mobile robot. *IEEE IROS'91*, Osaka (Japan), Nov. 1991.

[17] F.R. Noreils. Integrating multi-robot coodination in a mobile-robot control system. *IEEE IROS'90*, Tsuchiura (Japan), July 1990.

[18] P.A. O'Donnell, T. Lozano Perez. Deadlock-Free and Collision-Free Coordination of Two Robot Manipulators. *IEEE Transactions on Robotics and Automation*, 1989.

[19] P. Svestka, M.H. Overmars. Coordinated Motion Planning for Multiple Car-Like Robots using Probabilistic Roadmaps. *IEEE Transactions on Robotics and Automation*, 1995.

[20] Pierre Tournassoud. A strategy for obstacle avoidance and its application to multi-robot systems. *IEEE ICRA'86*, San Francisco (USA), April 1986.

[21] T.Tsubouchi, S.Arimoto. Behavior of a mobile robot navigated by an "iterated forecast and planning" scheme in the presence of multiple moving obstacles. *IEEE ICRA'94*, San Diego (USA), 1994.

[22] J. Wang. On sign-board based inter-robot communication in distributed robotic systems. *IEEE ICRA'94*, San Diego (USA), 1994.

[23] S. Yuta, S. Premvuti. Coordination autonomous and centralized decision making to achieve cooperative behaviors between multiple mobile robots. *IEEE IROS'92*, Raleigh (USA), July 1992.

Acknowledgments: This work was partially supported by the MARTHA (ESPRIT III) Project, the CNRS, the Région Midi-Pyrénées and the ECLA ROCOMI Project.

This project is the fruit of a very intensive collaboration between numerous researchers. We would like to acknowledge the help and the effective involvement of M. Khatib, H. Bullata, S. Suzuki, J. Perret, T. Siméon, B. Dacre-Wright, C. Dousson, M. Devy, P. Gaborit.