Proceedings of the 1992 IEEE
International Conference on Robotics and Automation
Nice, France - May 1992

Integrated Planning and Execution Control of Autonomous Robot Actions

Raja Chatila Rachid Alami Bernard Degallaix* Hervé Laruelle†

LAAS-CNRS
7, Ave. du Colonel Roche, 31077 Toulouse cedex – France

Abstract

This paper describes an implemented integrated system allowing a mobile robot to plan its actions, taking into account temporal constraints, and to control their execution in real time. The general architecture has three levels and the approach is related to hierarchical planning: the plan produced by the temporal planner is further refined at the control level that in turn supervises its execution by a functional level. The framework of the french Mars Rover project VAP is used as an illustration of the various aspects discussed in the paper.

1 Introduction

Intervention robots are machines that have to perform non-repetitive and time-constrained tasks in ill-known environments which are often remote or of difficult or dangerous access, with specific constraints on communication (delays, bandwidth). In this context, classical teleoperation as well as telerobotics-like [21] approaches with a human operator in the control loop are not adequate [9].

Some authors argue that planning is not useful, and that reactivity is the necessary and sufficient ingredient of robot intelligence [4]. Furthermore, *collective intelligence* would emerge as a result of the simple interactions of reactive robots which have complete autonomy but limited behavior [5, 7]. Being not able of *predicting* their actions and their outcome, wich is one aspect of planning, such robots are more data than goal driven, and if ever able to accomplish a given task, they would certainly lack efficiency.

The challenge is to design an autonomous robot endowed with the capacities of planning its own actions in order to accomplish specified tasks, and having a reactive behavior with respect to is environment. Such a robot would be both goal and data driven. The work presented in this paper is a contribution to this objective.

Plans are produced by a temporal planner at the higher level. They are then refined into actions according to specified execution modalities and to the actual execution context [2, 18]. The robot control structure includes the systems necessary for task interpretation and autonomous execution. The system is aimed to be generic and applicable to several domain instances of intervention robots: planetary rovers [6], underwater vehicles [12], disaster reaction [18], etc.

The paper is organized as follows: section 2 presents the functional architecture of the overall system; section 3 focuses on planning and section 4 plan execution supervision. Finally, section 5 presents the task refinement and execution levels.

In order to illustrate the approach we will consider a Mars Rover as an example of intervention robots troughout this paper. This case study is motivated by the french national project VAP[1] [6].

2 System Architecture

The global system architecture is organized into three levels (figure 1). A higher temporal planning level - with its own supervision. The planning time is unbounded.

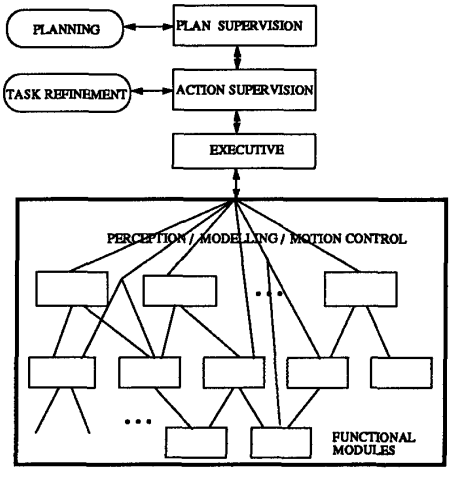

Figure 1: Global architecture

The second "refinement" level receives tasks that it transforms into sequences of actions using its own interpretation and planning capacities, and supervises

*Supported by IFREMER
†Supported by a "CIFRE" with MATRA

[1] VAP: Autonomous Planetary Vehicle, a project of the french national space agency CNES.

0-8186-2720-4/92 $3.00 ©1992 IEEE

the execution of these actions while being reactive to asynchronous events and environment conditions. This level comprises a **task supervisor** that is in charge of receiving the plan from the higher level and interacting with the second component, the **task refinement planner** which refines tasks into actions, taking into account the execution modalities specified with the plan, and the actual situation of the robot. The response time of this level is bounded as the refinement planning is in fact very much context-driven selection of actions in a precompiled structure.

The functional "execution" level is composed of a set of modules embedding the functions for sensing and acting, including various specific processings necessary for perception and motion control [2, 20, 10]. This level is managed and controlled by a central **Executive** in order to execute the actions requested by the task supervisor. The response time of these modules that implement polynomial time algorithms is bounded.

A module embeds primitive robot functions which share common data or resources [1, 8]. An internal control process called the "module manager" is responsible for receiving requests to perform these functions from the robot controller, and for otherwise managing the module. Each function being well defined, its activation or termination must respect certain conditions that the module manager verifies.

Modules interact by message passing or by reading data exported by other modules, and by putting their own processing results into exported data structures (EDS). At a given time, a module can be executing several functions. All of the functions of a given module are pre-defined at the system design stage.

In the context of a Mars rover, the planner could be on Earth, while the plan supervisor in charge of controlling its execution and the two other levels are on-board the robot.

3 Planning

3.1 Temporal Planning

The planning level is based on general action planning techniques, including temporal reasoning, since it is necessary to take into account time constraints in robots that have to act in the real world.

The explicit representation of time allows for an operator representation that is richer than STRIPS operators or their usual extensions as summarized in [16]. It is possible, for instance, to specify information concerning the duration of operators, the relative time when the postconditions of an operator become true, the conditions which must remain true during action execution, joint effects of operators with other operators executed in parallel, and the like. This has been partly addressed in temporal planners like DEVISER [22] and in event based planners like GEMPLAN [17] or, more recently, CHICA [19]. GEMPLAN also makes a difference between temporal and causal orderings of operators. FORBIN [11], as a time map based planner, involves most of these issues.

We have developed a temporal planning system called *IxTeT* (Indexed Time Table) [13, 15] which can reason on symbolic and numeric temporal relations between time instants. It produces a set of partially ordered tasks with temporal constraints.

IxTeT's representation is based on reified logic. It relies on a 2-dimensional array (called Indexed Time Table) with rows corresponding to logical assertions and columns to time points (instants). Cells in the table are temporal qualifications of the assertions. Temporal constraints between instants are represented by a time lattice. A temporal relation manager maintains the lattice and propagates temporal constraints [14].

The descriptions of the world, the goals and the decomposition operators are given using symbolic and/or numeric temporal relations between time points (instants) or a set of temporal relations between intervals (*start, meet, finish, during, overlap, before, same* and the inverse relations) [3] which can be transformed into relations between instants.

Ixtet involves two levels of description for decomposition operators: the *Task* and the *Procedure*.

Tasks in IxTeT correspond to the lowest level of description. The 'effects' of a task are assertions which change as a direct result of performing that task. Hence, a task is described through its 'effects', temporally linked to the interval of execution of the task. Minimal and maximal duration of a task may also be specified.

For example, in the task description given in figure 2, *(Data-Available ?data ?site ?target)* becomes true at the end of the task while *(Sending-Data ?target)* becomes and remains true during task execution.

```
(ixtet:Task TASK-SEND-DATA
        :args (?data ?site ?target)
        :effects ( (:meets Data-Available ?data ?site ?target)
                   (:equal Sending-Data ?target))
        :duration-min (length ?data)
        :duration-max (length ?data))
```

Figure 2: A task description

Procedures are the decomposition operators. They are defined in terms of a context (a set of conditions needed in order to apply the procedure), a set of temporally constrained tasks required to achieve a particular goal, together with additional effects which are due to the combined execution of the tasks involved by the procedure.

In the example given in figure 3, in order to apply the procedure *SEND-PANORAMA*, the robot will have to wait until *(Sunlight)* becomes true before executing *TASK-GET-PANORAMA(?site)*, and to wait for a visibility window with the target (Earth or Orbiter) and the rover, with a sufficient duration for performing *TASK-SEND-DATA(panorama, ?site, ?target)*.

Planning in IxTeT proceeds from an initial context called 'initial situation' containing an initial state, ex-

```
(ixtet:Procedure SEND-PANORAMA
   :args (?site ?target)
   :goal (Data-Available panorama ?site ?target)
   :context (   (:during s1 (:on (Sunlight)))
                (:during s1 (:on (VAP-Site ?site)))
                (:during s2 (:on (Visibility ?target))))
   :steps       (s1 (TASK-GET-PANORAMA ?site)
                 s2 (TASK-SEND-DATA panorama ?site ?target))
   :such-that ((:before s1 s2))
   :effects ())
```

Figure 3: A Procedure description

pected events and goals which are linked (directly or indirectly) to an initial time-point by numerical or symbolic temporal relations. The occurence dates may be specified more or less precisely (ranging from an exact absolute or relative occurence date, to a numerical interval, to a simple symbolic ordering).

The planner performs goal decomposition by heuristically selecting procedures and inserting them in the time lattice together with constraints addition and propagation.

3.2 Example

Here we describe an example based on the VAP mars Rover mission scenario. If on-board computing capacities are limited, planning may take place on Earth, and the plan be sent to the rover for refinement and execution.

The mission of VAP consists in sending to the orbiter a panoramic view of a zone called *site-2* and in collecting geological samples from *site-3*.

The initial time-point is noted 1. At this time the rover is localized in a geographic zone called *site-1*. Besides, there are three expected events that are known to occur in the future: *(Sunlight)* will become true at instant 2, a visibility window will start with the orbiter at instant 18 and with Earth at instant 17. This initial situation is represented by the time lattice in figure 4.

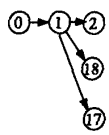

Figure 4: The initial situation

The procedure *Vap-Navigate(?site1, ?site2)* can only be applied during day-time (sunlight). Besides, *Collect-samples(?site)* can be applied only after the panoramic view of *?site* was transmitted.

Figure 5 illustrates the plan as produced by IxTeT. The upper part of the table shows the validity in time of the predicates describing the world state and the effects of the robot's actions (values before instant 1 are not relevant). The lower part shows the the tasks

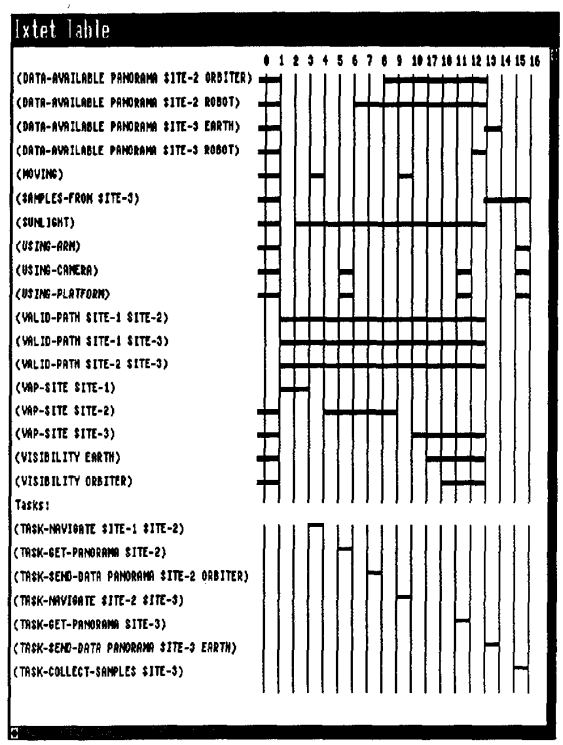

Figure 5: The indexed time table. The upper parts shows predicate validity in time and the lower part the tasks in the plan.

composing the plan and their beginning and end instants. Note that the identifiers of the time instants are purely arbitrary and are not ordered in figure 5. The temporal relations between the instants (partial order) are given in figure 6.

We see that in order to achieve the mission goal, the rover must navigate from *site-1* to *site-2*, acquire a panoramic view, and then send it to the orbiter, navigate from *site-2* to *site-3*, acquire a new panoramic view which is sent to Earth, and finally collect samples. Note also that the rover will wait until *Sunlight* (instant 2) becomes true before moving and that it will wait for visibility with the orbiter or Earth in order to communicate with them. Furthermore, the tasks *Send-data(Panorama, site-2, orbiter)* and *Navigate(site-2, site-3)* may be executed in parallel (instants 7 and 9).

4 Plan Supervision

4.1 Introduction

A Plan, as produced by IxTeT, is a set of partially ordered tasks, together with temporal constraints such as minimal and maximal expected durations, and synchronisation with expected external events or absolute dates.

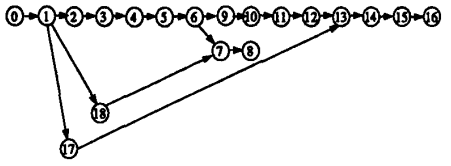

Figure 6: The lattice of time instants.

The plan supervisor is in charge of monitoring the execution of the tasks involved in the plan. It has therefore to verify that the tasks produce indeed the expected effects, and must react in case of discrepancy. Execution will be globally managed by the robot supervisor, that schedules the tasks, sending them to the action supervisor that controls their execution, possibly after refinement.

4.2 Plan Execution Monitoring

Instants in the time lattice correspond to different event types: beginning or end of task execution, intermediate events produced by tasks during their execution, expected external events (which occur independently from robot actions). Besides, numerical bounds for dates and durations may be attached to some time-points or intervals.

The plan execution control process interacts with a clock by requiring messages to be sent at given absolute dates, and with the robot action supervisor by requiring task execution or cancellation (fig 7).

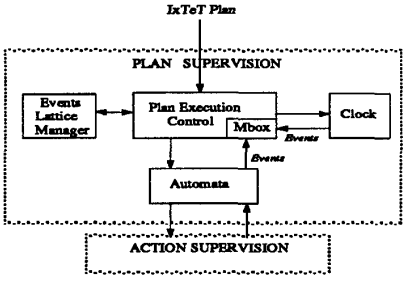

Figure 7: The Plan supervisor

Messages received from the clock authorize the system to state the occurence of dated expected events and to monitor the tasks minimal and maximal expected durations.

Messages received from the robot action supervisor are 'filtered' by the associated automata (see below §4.3) and transformed into events which correspond either to instants planned in the time lattice (in case of nominal execution) or to unexpected instants otherwise.

The plan supervisor starts from a given instant in the time lattice and 'executes' the time lattice by performing the following actions:

- At any moment, it considers only the time instants whose predecessors have been processed and whose planned occurence date is compatible with the current time.
- It requires the execution of a new task when it reaches the instant which corresponds to its beginning.
- While processing incoming events, the supervisor verifies that they correspond to planned instants and that they satisfy the planned ordering and numerical time constraints. If it is the case, the absolute date corresponding to the occurence of the events is considered, inducing a progressive 'linearisation' of the time lattice (see figure 8).
- In the current implementation, the only 'standard' reaction performed by the plan supervisor in case of non-nominal situations is to stop the current tasks, and update the world state according to events incoming from task execution, until all tasks are stopped. This is performed by inserting new time instants in the time lattice corresponding to the transitions as defined by the automata.

Figure 8 shows the plan at two different stages of interpretation. The first time lattice represents a situation wherein the rover is navigating from *site-2* to *site-3* (indeed, instant 9 is in the past while instant 10 is still in the future). It has previously acquired a panoramic view of *site-2* (between instants 5 and 6), however it has to wait for instant 18 (visibility with the orbiter) in order to send the data (between instants 7 and 8).

In the second lattice, instant 18 has occured; the rover is performing two tasks in parallel (navigating and sending data to the orbiter).

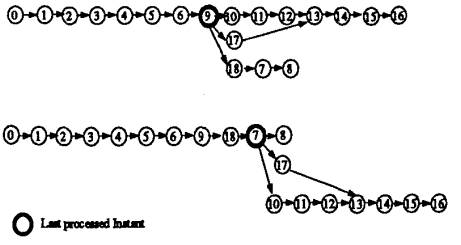

Figure 8: Plan execution

4.3 Using Automata To Monitor Task Execution

In order to allow the plan supervisor to monitor task execution and to act on them while they are executed, each task is modelled by a finite state automaton (FSA).

The FSA associated to a task not only models the nominal task execution (as defined in the task description manipulated by the planner) but also non-nominal situations (exceptions), as well as the different actions which should be taken then by the plan supervisor.

In the finite state automata we use, state transitions can be caused by 'internal events' (i.e. events generated by the task execution process and transmitted to the plan interpreter) or by 'external events' (i.e. events due to an action of the plan supervisor on the task while it is executed).

Typical internal events are: RUN (beginning of execution), RETURN (nominal end of execution), STOP_ON_FAILURE (end due to a failure), STOPPED (end of execution caused by an external STOP request). A Typical external event is STOP (corresponding to a stop request issued by the plan interpreter). Other external events are related to the task and force a state transition in the execution.

For each task, an automaton class is provided. An example is given in figure 9. Note that each automaton state change produces transitions in the world state as it is maintained by IxTeT. The example in figure 9 shows a decription of automaton associated to the task *TASK-NAVIGATE*. Whenever a task execution is started by the plan supervisor, an automaton of the associated class is instanciated and initialized. It will then represent the execution of the task as it is viewed by the plan supervisor.

```
(ixtet:Task TASK-NAVIGATE
    :args (?aitel ?aite2)
    :effects ((:is-set VAP-Site ?aitel)
              (:meets VAP-Site ?aite2)
              (:equal Moving))
    :duration-min (min-length ?aitel ?aite2)
    :duration-max (max-length ?aitel ?aite2))

(DEFINE-AUTOMATE-CLASS TASK-NAVIGATE (?aitel ?aite2)
 (:STATE begin
  :INTERNAL-EVENTS ((:run
                     :NEXT-STATE :executing
                     :TRANSITIONS ((:ON MOVING)
                                   (:OFF VAP-SITE ?aitel))))
  :EXTERNAL-EVENTS ((:stop
                     :NEXT-STATE :stopping
                     :TRANSITIONS ())))
 (:STATE executing
  :INTERNAL-EVENTS ((:return
                     :NEXT-STATE :end
                     :TRANSITIONS ((:OFF MOVING)
                                   (:ON VAP-SITE ?aite2)))
                    (:execution_failure
                     :NEXT-STATE :failure
                     :TRANSITIONS ((:ON NAVIGATION-FAILURE))))
  :EXTERNAL-EVENTS ((:stop
                     :NEXT-STATE :stopping
                     :TRANSITIONS ())))
 (:STATE failure
  :INTERNAL-EVENTS ((:stop_on_failure
                     :NEXT-STATE :end
                     :TRANSITIONS ((:OFF MOVING)
                                   (:ON VAP-SITE CURRENT)
                                   (:ON VALID-PATH CURRENT ?aitel)
                                   (:OFF VALID-PATH ?aitel ?aite2))))
  :EXTERNAL-EVENTS ((:stop
                     :NEXT-STATE :stopping
                     :TRANSITIONS ())))
...)
```

Figure 9: Automaton for TASK-NAVIGATE

5 Refinement and Execution Control

5.1 Task Refinement

Task Refinement transforms a task into specific actions that are adapted to the actual context. As an example, a motion task may be executed in different ways:

- a displacement using only dead-reckoning systems to guide the movement.
- a closed-loop motion using a perceptual feature of the environment (landmark tracking, edge following of a large object, rim following, etc.).

These two modes correspond to the execution of different *scripts* (see [18] for details), associated with the task "move". Script selection is based on testing conditions using the acquired data. Scripts have variables as arguments, that are instanciated at execution time. The execution of a script is similar to the execution of a *program*.

5.2 Task Execution

Action execution are represented by *activities*. The execution of a script corresponds to a global activity. An activity is thus equivalent to the execution of a program, and is analogous to the notion of process in an computer operating system. A simple activity is the execution of a function by a module. An activity may cause the emission of requests to other modules, starting *children activities*. The module in the parent activity is then a client of the module in the child activity. An activity may be the parent of many children, but may be the child of only one other activity. A set of rules and mechanisms were developed to create and manage activities. Two basic mechanisms are activity creation and message transmission between two activities.

At a given moment, the set of activities represent the functions being executed in the robot system. The activity structure is a tree with a parent-child relationship. The tree evolves while the robot is executing. An activity communicates with its (single) parent activity via "up-signals" and with its (eventual) child activities via "down-signals". The activity hierarchy is not predefined and depends on the current task.

Regardless of the specific processing it performs, an activity must be able to react to signals sent by its parent or its children. In particular, it must be able to react to asynchronous signals within a pre-defined bounded time delay.

An activity is also represented by a finite automaton. Its state changes are dependent on external or internal signals. Control flow between a mother and child activities is implemented as typed messages that cause a state change. Specifc mechanisms permit the propagation of a state change along the activity tree.

Important features of the notion of activity are:

- Activities provide for the management of a hierarchy of actions without imposing a fixed number of layers.
- All activities, regardless of level, may be treated in the same way.
- No assumptions are made about the nature of the inter-connections between activities: the activity

hierarchy is *not* pre-defined. We might require that a "high level" activity starts and manages a "low level" activity at any level. This knowledge must however exist in the modules.

- All mechanisms for managing activities (starting, terminating, etc.) and the communication between them do not depend upon any programming language constructs; in this sense, they resemble part of an operating system.
- The concept of activity permits reactivity at all levels: at any moment, each activity in the tree structure is able to respond to asynchronous events.

Figure 10 represents an example of activity tree at one stage of the execution of a *go-to(location)* task. The "root" box represents the mother activity of this task, within the executive. The "monitor-11" box is the mother activity of a main child-activity (monitor-12) and of an associated monitoring activity (timer-200: a time-out for this task). When the corresponding event occurs, the main activity is stopped. "Monitor-12" is a monitoring activity also: the traveled distance should not exceed a given maximum (here 60 meters). Sequence in the "sequence-6" box means that the following children are to be executed in sequence (this defines a sub-activity block). "go-to-xy" and "exec-traj" are the two sequential activities. The first is the "go to" coordinate location, and the second is the trajectory execution phase within it. Other possible phases are "build-model" (environment modelling) and "build-traj" (trajectory planning). The last box "move-5" is the primitive action move being executed.

Figure 11 shows the execution of the *go-to(location)* task in a laboratory experiment, where the robot discovers its environment and builds a model of it while it navigates to reach the assigned location.

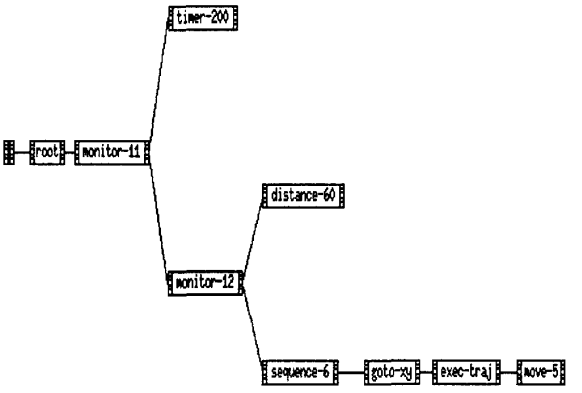

Figure 10: Activity tree during execution of a GOTO task

6 Conclusion

We presented in this paper a global approach to task planning and execution for intervention robots. Such robots are characterized by the fact that they have to be able to autonomously execute their actions in a partially and poorly known environment in order to accomplish missions and tasks specified and programmed by a human user. The robot have to interpret the tasks according to the context and its evolution, and to achieve autonomous execution. The main interest of the plan execution supervision as it is proposed in this paper is to maintain and update a complete history of the plan execution not only in nominal cases but also when a failure occurs. This is performed using automata which model tasks execution and their interaction with the plan supervisor. However other representations may be used. Indeed, we are working on an extension using on a compiled rule-based system which should allow not only to model task execution but also combined effects due to the simultaneous execution of several tasks as well as domain dependent knowledge allowing to infer new transitions starting from a set of observed events. While it is still necessary to further deepen some aspects, the experimental results show that this approach is sound and applicable.

References

[1] R. Alami, R. Chatila, M. Devy, and M. Vaisset. System architecture and processes for robot control. Technical report, Laboratoire d'Automatique et d'Analyse des Systèmes (C.N.R.S.), Toulouse (France), June 1990.

[2] R. Alami, R. Chatila, and P. Freedman. Task level programming for intervention robots. In *IARP 1st Workshop on Mobile Robots for Subsea Environments, Monterey, California (USA)*, pages 119–136, October 1990.

[3] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[4] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, 1986.

[5] C. M. Angle and R. A. Brooks. Small planetary rovers. In *IEEE International Workshop on Intelligent Robots and Systems (IROS '90), Tsuchiura (Japan)*, pages 383–388, July 1990.

[6] L. Boissier and G. Giralt. Autonomous Planetary Rover (VAP): The robotics concepts. In *I.A.R.P. 91 on Robotics in Space, Pisa (Italy)*,1991.

[7] R. A. Brooks, P. Maes, and G. Moore. Lunar base construction robots. In *IEEE International Workshop on Intelligent Robots and Systems (IROS '90), Tsuchiura (Japan)*, pages 389–392, July 1990.

[8] R. Ferraz De Camargo, R. Chatila, and R. Alami. A distributed evolvable control architecture for mobile robots. In *'91 International Conference on Advanced Robotics (ICAR),Pisa (Italy)*, pages 1646–1649, 1991.

[9] R. Chatila, R. Alami, and G. Giralt. Task-level programmable intervention autonomous robots. In *Mechatronics and Robotics I, P. A. MacConaill, P. Drews and K.-H. Robrock Eds, IOS Press*, pages 77 – 87, 1991.

[10] R. Chatila and R. Ferraz De Camargo. Open architecture design and inter-task/intermodule communication for an autonomous mobile robot. In *IEEE International Workshop On Intelligent Robots and Systems, Tsuchiura, Japan*, July 1990.

[11] T. Dean, R.J. Firby, and D. Miller. Hierarchical Planning Involving Deadlines, Travel Time, and Resources. *Comput. Intell.*, 1988.

[12] L. Floury and R. Gable. The Wireline Reentry in Deep Ocean DSDP/ODP Holes. Technical Report DITI/ICA-91/172-LF/DB, IFREMER - Centre de Brest, July 1991.

[13] M. Ghallab, R. Alami, and R. Chatila. Dealing with Time in Planning and Execution Monitoring. In R. Bolles, editor, *Robotics Research: The Fourth International Symposium*. MIT Press, Mass., 1988.

[14] M. Ghallab and A. Mounir Alaoui. Managing Efficiently Temporal Relations Through Indexed Spanning Trees. In *11th International Joint Conference on Artificial Intelligence (IJCAI), Detroit, Michigan (USA)*, pages 1297–1303, 1989.

[15] M. Ghallab and A. Mounir Alaoui. Relations temporelles symboliques: représentations et algorithmes. *Revue d'Intelligence Artificielle*, 3(3), February 1990.

[16] J. Hendler, A. Tate, and M. Drummond. AI Planning: Systems and Techniques. *Artificial Intelligence Magazine*, 11(2):61–77, Summer 1990.

[17] A.L. Lansky. A Representation of Parallel Activity Based on Events, Structure, and Causality. In *Reasoning about Actions and Plans. Proc. of the 1986 Workshop, Timberline*, pages 123–159, 1987.

[18] R. Laurette, A. de Saint Vincent, R. Alami, R. Chatila, and V. Pérébaskine. Supervision and Control of the AMR Intervention Robot. In *'91 (ICAR),Pisa (Italy)*, pages 1057–1062, June 1991.

[19] L. Missiaen. *Localized Abductive Planning with the Event Calculus*. PhD thesis, Dept. of Comput. Science, KU Leuven (Belgium), 1991.

[20] F. R. Noreils, R. Chatila. Control of Mobile Robot Actions. In *IEEE Int. Conf. on Robotics and Automation*, 1989.

[21] T. Sheridan. Telerobotics. In *IEEE Int. Conf. on Robotics and Automation (Workshop on Integration of AI and Robotic Systems*, 1989.

[22] S.A. Vere. Planning in time: Windows and durations for activities and goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(3), May 1983.

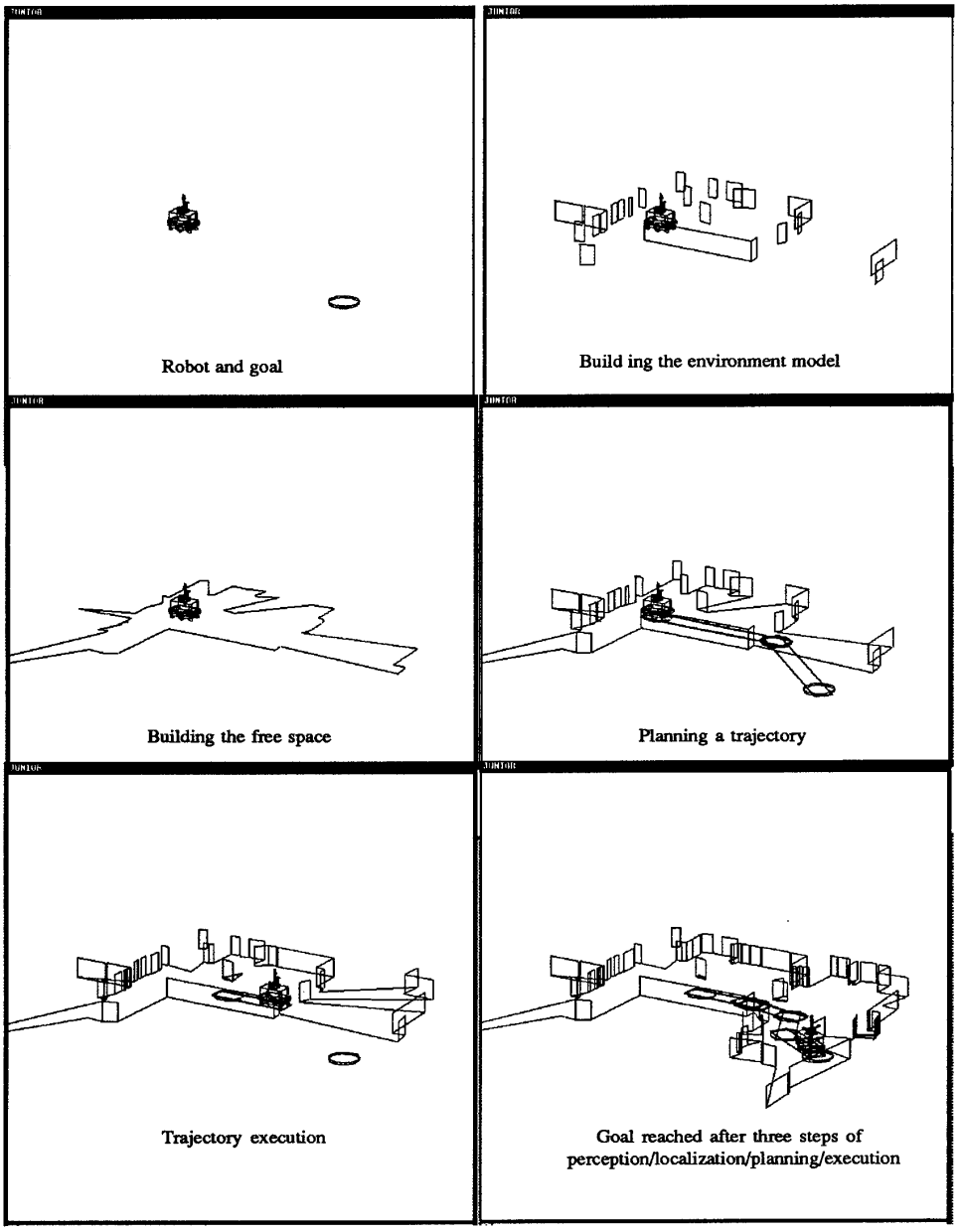

Figure 11: Experimental Execution of a *goto(location)* task.