Supervision and Control of the AMR Intervention Robot

Renaud Laurette
Arnaud de Saint Vincent

Matra-Espace
31 rue des cosmonautes
31077 Toulouse Cedex - France

Rachid Alami
Raja Chatila
Victor Pérébaskine

LAAS/CNRS
7 avenue du colonel Roche
31077 Toulouse Cedex - France

I. INTRODUCTION

The Eureka project AMR (Advanced Mobile Robots) aims at prototyping two intervention robots, for public safety applications in hostile environments. One of them is devoted to outdoors operations, while the second one is for indoors interventions. The project, started in 1987, involves companies and research laboratories from France, Italy and Spain [1].

One of the characteristics of the AMR missions is that the radio link with the mobile unit cannot be maintained during the whole intervention. Hence, providing the robot with the on-board capabilities to achieve an autonomous behavior is a major issue of the project. These capabilities include autonomous task management, environment modeling, motion planning, auto-localization, obstacle avoidance, etc.. [2]. Whenever the link is available, the same capabilities will allow the operator to interact with the robot at a high level of abstraction that will secure the use of the mobile unit (quicker reactions, minimum risk of human mistake) and improve the mission efficiency.

Hence, two main operating modes are foreseen : a teleoperated mode in which the operator keeps a close control on the robot at action or effector level, and an autonomous mode, in which the operator gives a goal to the robot, which in turn sequences and monitors by itself its activities according to predefined strategies.

Transforming the goals into simpler tasks, choosing the most adequate way to perform the tasks, controlling their execution and handling external events and internal failures, is under the responsability of the robot's Supervision Unit, described in this paper. After introducing the general architecture of the Unit, we focus on task execution, and then on the architectural impacts on the design of the robot's sub-systems.

Manuscript received April 3rd, 1991

The Supervisor was designed in cooperation between Matra Espace and Laas. It is inspired from the work conducted at Laas in the more general frame of intervention robot control and teleprogramming [3][4][5], and also from the TCA architecture developed at CMU [6].

II. THE SUPERVISION ARCHITECTURE

A. Overview

The operator communicates with the robot at three different levels, ranging from effector level (direct control of the robot's effectors) to action level (local movement or manipulation tasks for which the robot coordinates by itself its subsystems), and up to goal level (the robot finds a sequence of tasks to achieve the goal). These three levels correspond to three entities in the architecture of the Supervisor. Respectively : the *Executive*, the *Task Manager* and the *Mission Manager*. The architecture we have chosen (Fig. 1) is organized around the Executive, which receives orders from the operator, interfaces the robot's subsystems, maintains a representation of the on-going activities and of the allocated resources. An effector level order (a routine) is expanded and transmitted by the Executive to the concerned subsystems. An action level order (a task), is routed to the Task manager, which will refine it into routines, and sequence these routines back to the Executive. A goal level order (a mission) is routed to the Mission manager, that operates as the Task manager does, but at mission level. Conversely, completion statuses are transmitted the other way up from the subsystems to the operator. The rationale for this architecture is twofold : the availability of a unique central representation of the robot's activities and resources together with that of specialized entities to handle different abstraction levels.

B. The Executive Level

The Executive interfaces the robot's subsystems and is responsible for coordinating them so that, at any moment, the robots exhibits a coherent behavior. It receives orders, check their appropriateness, transforms and dispatches them, and monitors their execution. This coordination is made efficient by organizing the architecture of the subsystems into functional *modules*, as described further. The processing performed on each order by the Executive is organized according to the following steps :

1) Resource allocation: By consulting an internal model of actions, the Executive checks which are the resources involved in the execution of the received order, whether they are available, and allocates them.

2) Activity management: If the resources are successfully allocated, the robot is able to unterdake the activities corresponding to the received order, and the activity tree is updated. This tree describes the instantaneous hierarchy of the on-going activities, and mentions for each one the originator, the receiver and the status. If any activity is further generated, it will be linked to its mother activity by a mother-daughter link.

3) Monitoring: One possible type of order is the monitoring of a logical condition. In such a case, at this step, the Executive stores the condition to be monitored, decomposes it into elementary conditions to be monitored at subsystem level, and stores a reflex action to be executed when the condition is verified.

4) Activation: At last, the order is routed to its receiver(s). Note that in the case of behaviour level orders, a single received order may map onto several activities at subsystem level. Therefore, several messages may be output by this level.

Conversely, the Executive receives status messages from the subsystems, evaluates the monitored conditions, updates the activity tree, frees the resources and reports to the originator of the activity. The Executive also handles an *internal status* which is updated by subsystem messages. This architecture allows for a great flexibility : The originator and the receiver may be any entity connected to the Executive. However, in the AMR context, only the operator, the Mission manager and the Task manager are foreseen as originators.

C. The Task Level

When an order is recognized as a task in the Executive model of actions, it is sent to the Task manager. A *task* corresponds to a possibly complex sequence of local actions (short range movement, data acquisition, ..) described by predefined *procedures* [7]. In many cases, several alternative procedures will correspond to a single task. For instance, a movement to a close goal may be performed either by modeling the environment and planning a trajectory, or by servoing the motion on a nearby wall, or by tracking a salient visual feature. Hence, a *Selector* will first choose the most appropriate procedure according to the robot's internal status (resource availability) and to environmental parameters. Some other tasks, such as effector level tasks, map onto a single procedure, and no selection need to be performed.

Using procedures instead of planning the actions on-line was chosen at task level in our application for time efficiency and because the number of alternative strategies for a given task did not call for a planner. However, should the need appear, the architecture allows to substitute the Selector for a Planner without any further change.

The procedure contains the description of the actions to be undertaken to achieve the task, as well as the reactions to perform in response to asynchronous events. Actions and reactions are described in a unified formalism called the *script* of the procedure. A script contains the usual constructs of high level languages : testing, branching, iterating, nesting, as well as synchronization mechanisms for parallel execution. Moreover, it handles a special construct to deal with asynchronous events : the surveillance. A surveillance consists in a logical condition to be monitored on internal or sensed data, a reflex action to be performed immediatly when the condition fires, and a control action, expressed as a subscript. The reflex action guaranties the safety of the robot, while the control action indicates how to cope with the event. The sequencing of the script is performed by the *Interpreter*, which sends actions and surveillances to the Executive.

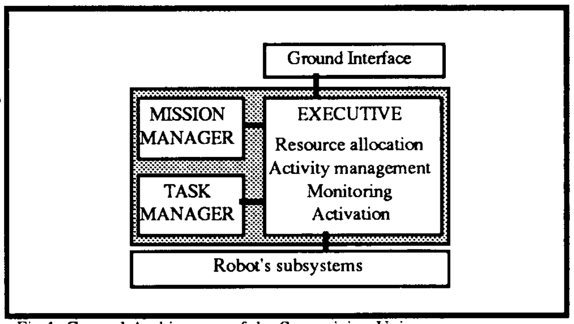

Fig.1. General Architecture of the Supervision Unit

D. The Mission Level

The Mission manager deals with a high-level description of the robot's activities, and of the environment. At mission level, tasks may be represented in terms of state operators. Thus, mission refinement is performed using general action planning techniques, while task refinement is performed using a procedure based approach. In the AMR context, mission refinement only concerns long range movement orders (e.g. reach distant place X, moving through different topological "places"). The mission interpretation will determine sub-goals to be reached by the robot, and the corresponding movement tasks. The subgoal search is made using an a-priori approximative topological description of the environment provided by the user. This model is enhanced by landmarks (objects that the robot may recognize autonomously) and by qualitative information (terrain roughness, risk, availability of communications, ...) When the movement tasks are planned, the mission level sequences them to the Executive, and not directly to the Task manager in order to keep the consistency in activity representation.

III. ROBOT SUBSYSTEMS

In order to improve the efficiency of the control structure and robot operation, we defined the basic functions in a systematic and formal way so that they can be controlled according to their specific features, while being easy to combine, modify or redesign. We therefore defined in [8] and extended in [9] the notion of *robot module* and introduced *primitive function types*.

A module embeds a set of primitive robot functions which share common data or resources. An internal control task called the "module manager" is responsible for receiving requests to perform these functions from the Executive, and for otherwise managing the module.It also manages the specific activation and termination conditions of each type of module. A module may read data exported by other modules, and may output its own processing results to Exported Data Structures (EDS). At a given time, a module can be executing several functions. All of the functions of each module are pre-defined at the robot design stage. The robot primitive functions fall into four different *types* according to their functioning mode (Fig. 2).

1) Servers: This type of function is executed upon request. The result of the processing is put into an EDS to be accessed by the requesting module.

2) Filters: Such functions are started by a request and then run continuously at a given rate (some filters may run automatically as soon as the robot is switched on). Their results are output at the given rate in an EDS.

3) Servo-Processes: These functions implement a closed-loop between a perception function (related to processing sensory data) and an "action" function (related to the robot effectors). The servo-process has known parameters that must be respected when connecting the input and output modules. For example, a wall following servo-process function might be defined to control the robot's motion, using information provided in an EDS associated with a filter which interprets data from the ultrasonic sensors, and that is read by the adequate locomotion module function.

4) Monitors: These functions are used to detect a given situation, expressed by a logical condition, and to react by generating an event for the supervisor. The logical condition is evaluated regularly, at a predefined rate.

Both filters and servo-processes may be stopped, but cannot stop by themselves, unless they detect an abnormal condition ; in such a case, an appropriate event is sent to the supervisor. An example of a routine involving the interaction of several modules is wall following (Fig. 3).

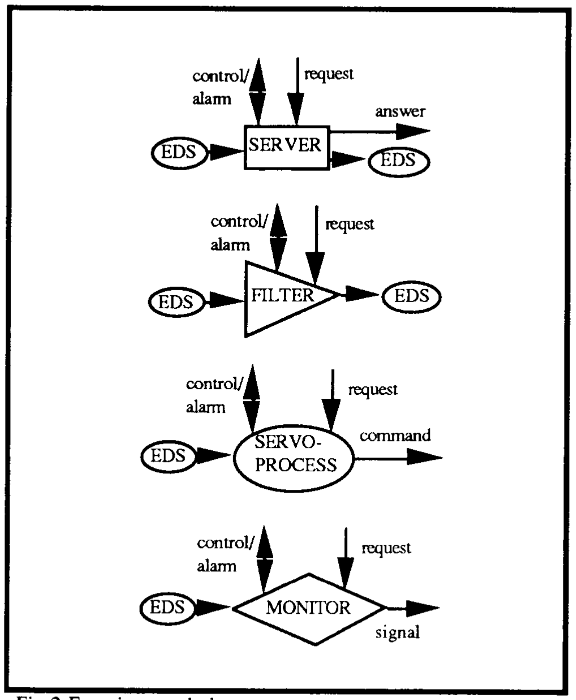

Fig.2 Function symbology.

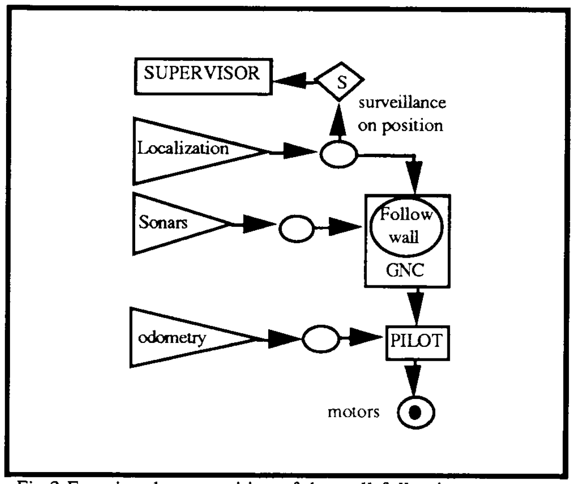

Fig.3 Function decomposition of the wall-following process.

IV. TASK EXECUTION

We present here the task execution mechanisms, and we illustrate them through the task "Go to landmark" *(goto-ldmk)*. The goal of this task is to take the robot from its current location to the vicinity of a landmark previously defined in the topological model of the environment. This landmark may be located anywhere in the current topological place. Hence, the task involves autonomous navigation among possible obstacles, landmark recognition, and landmark approach.

A. Knowledge representation

To execute such a task, the supervision subsystem relies on different types of knowledge. The *operational knowledge* is a static representation of tasks . It describes how tasks map into procedures, and which steps the procedure consists of. It also contains information for fault diagnosis and error recovery (Table I). This knowledge is given to the robot before the mission starts. Apart from this static information, the robot maintains dynamic data, organised in three main models. The *internal state* describes the state of the different sub-systems. Part of this data is reported periodically to the Supervisor, while the other part is available from each sub-system upon request. The *external state* gather information relative to the robot's situation with respect to the topological model, such as the current topological place, and also data collected by or derived from the sensors' outputs: the robot's position and attitude, environmental conditions such as the temperature or the radiation level. The *robot activity* encompasses the description of all the activities currently undertaken by the robot (the current task, the selected procedure, the action inside the procedures, the active surveillances, and so on).

TABLE I. TYPICAL DATA STRUCTURES CONTAINED IN THE OPERATIONAL KNOWLEDGE BASE.

Task	Procedure
post-conditions	preconditions
list of pairs - procedures - cost elements	script
cost function	

B. The selection mechanism

When a task is sent to the Task Manager, a Selector is called (Fig. 4). This module aims at choosing the better suited procedure to achieve the task's goal. In the *goto-ldmk* example, several procedures may be chosen from (Table II). One involves an iterative process including environment modeling, subgoal and path generation and execution. Another one is eligible only if the terrain is smooth and if the movement may be servoed to a nearby wall through ultra-sonic sensors. The former procedure involves the most efficient capabilities of the robot, and has the highest probability of success. An approach could be to always use this procedure to perform a *goto-ldmk*. On the other hand, if the preconditions of the latter procedure are satisfied, the former takes much more time, for it needs to construct the environment model and to generate a trajectory. Hence, the goal of the Selector is to determine which procedure is likely to be the most effective in the current context.

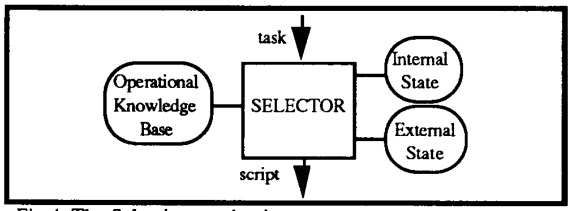

Fig.4. The Selection mechanism

TABLE II. PROCEDURES FOR THE *GOTO-LDMK* TASK

task	goto-ldmk (the-ldmk)
postcond.	(robot inside approach-zone of the-ldmk)
procedures	follow-corridor-to-ldmk ; cost=2 plan-and-follow-path-to-ldmk ; cost=4 follow-wall-to-ldmk ; cost=3 follow-simple-path-to-ldmk ; cost=1
procedure	follow-corridor-to-ldmk (the-ldmk)
precond	(robot inside corridor ?x) and (the-ldmk inside corridor ?x) and (distance between robot and the-ldmk > dmin) and (laser range finder available)
description	Use laser sensor to servoe the robot's movement to the irection of the corridor
procedure	plan-and-follow-path (the-ldmk)
precond	(laser range finder available)
description	Use the laser to build a map of the environment, and the path generator to compute a path to a subgoal. Iterates till the goal is reached.
procedure	follow-wall-to-ldmk (the-ldmk)
precond.	(laser range finder available) and (the-ldmk close to wall ?x) and (distance between robot and the-ldmk > dmin) and (us sensors available) and (terrain smooth)
description	Use the ultrasonic sensors to servoe the robot's movement to the direction of a nearby wall.
procedure	follow-simple-path-to-ldmk (the-ldmk)
precond	(laser range finder available) and (distance between robot and the-ldmk < dmax)
description	Generates a simple path (straight moves and rotations) to go to the landmark, assuming that the terrain is smooth. Performs local obstacle avoidance base on the LRF. Short range moves only.

First, the post-conditions of the task are evaluated. They encode the goal the task achieves. If they are satisfied, the Selector, and then the Task Manager, return a successful completion status. Otherwise, the Selector proceeds to choose the best procedure to perform the task. A list of candidate procedures is attached to each task in the operational knowledge base. A candidate procedure is eligible if its pre-conditions are satisfied. The final choice between all eligible procedures is made by evaluating a cost function attached to the task.

C. Interpretation and Execution

Once the procedure has been selected, its script is interpreted and executed by the *Interpreter*. The formalism of the script allows for both the actions and the reactive behaviour of the robot to be described. An action may be either a subtask or a *routine*. A substask will be recursively expanded into a (sub)script, using the selection mechanism described above. A routine corresponds to an actual robot action. This action is further decomposed and monitored by the *Executive* which is in charge of coordinating the subsystem activities to perform the action. The reactive behaviour is described by the surveillances.

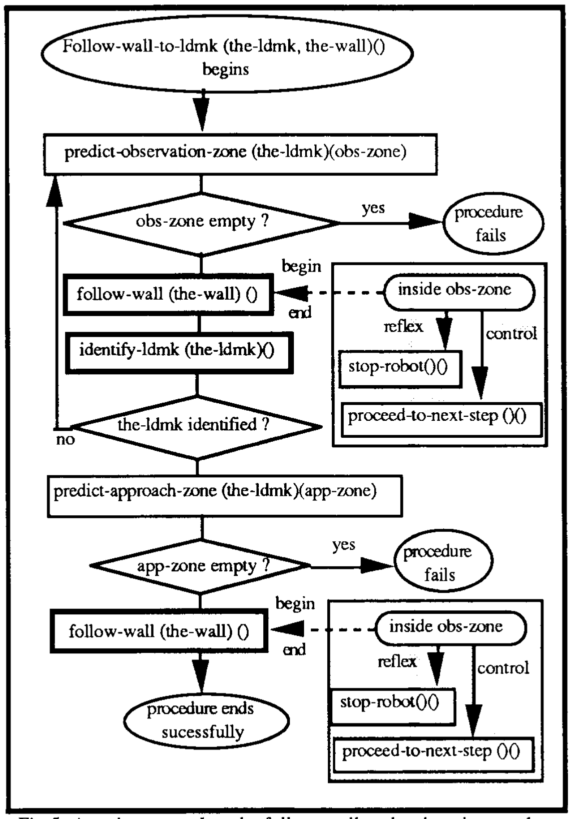
Follow-wall-to-ldmk (the-ldmk, the-wall)() begins
predict-observation-zone (the-ldmk)(obs-zone)
obs-zone empty ?
yes
procedure fails
begin
follow-wall (the-wall) ()
end
inside obs-zone
reflex
control
stop-robot()()
identify-ldmk (the-ldmk)()
proceed-to-next-step ()()
the-ldmk identified ?
no
predict-approach-zone (the-ldmk)(app-zone)
app-zone empty ?
yes
procedure fails
begin
follow-wall (the-wall) ()
end
inside obs-zone
reflex
control
stop-robot()()
procedure ends sucessfully
proceed-to-next-step ()()

Fig.5. A script exemple : *the follow-wall-to-landmark* procedure

Assuming that the procedure *follow-wall-to-ldmk* has been selected, the script would look like in fig. 5. In the main script, we see that both routines (zone predictions) and substasks (*Follow-wall, identify-landmark*) appear. Since wall-following is a mode that does not end in itself, surveillances (in the grey boxes) are used to proceed to the next action when the robot reaches the target zones. Different procedures (not presented here) may be used to perform *identify-landmark*, depending whether the robot needs to move, whether laser or video need to be used, etc. On the other hand, *Follow-wall* is a specific task which maps onto a single procedure (not presented here).

When a new step is started, the Executive first receives the surveillances whose scope start with the step, and then, the routine to be executed at that step. The routine will be expanded into primitive orders for the concerned subsystem(s).

When the routine ends, the associated surveillances are destroyed. Then we proceed to the next step. When a monitored condition fires, the Executive immediatly executes the reflex action. The Interpreter is then notified, and the control action, expressed as a script, is undertaken (Fig. 6). Should the surveillance be issued by another entity than the Interpreter, the notification would be sent to the originator.

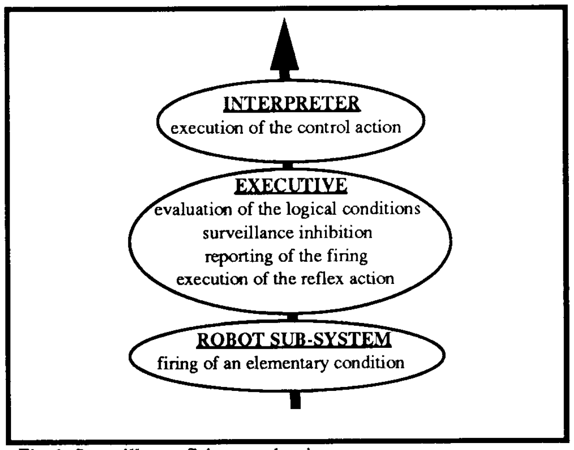
INTERPRETER
execution of the control action
EXECUTIVE
evaluation of the logical conditions
surveillance inhibition
reporting of the firing
execution of the reflex action
ROBOT SUB-SYSTEM
firing of an elementary condition

Fig.6. Surveillance firing mechanism

D. Failure recovery

The robot activity is organized hierarchically : tasks are expanded into procedures that may contain subtasks. When a failure occurs, the Interpreter calls a Failure Recovery Unit, which will first determine the type of failure and the hierarchical level at which it should be treated (Table III). The strategy of the Recovery Unit is to modify the current activity as little as possible. It will first attempt to recover at the level where the failure appeared. For example, if a tracking routine fails, it will try to find a predefined recovery procedure, such as "move the sensor to acquire the target again", to overcome the problem locally. If no solution is found at one level, the failure is propagated to the upper hierarchical level. At the higher level, the operator is prompted for a solution. Should communications be unavailable, the robot attempts to recover them, either by going to a reference place, or by searching in the inmediate neighborhood. No further detail is given on this recovery aspect which is currently under investigation.

TABLE III. FAILURE CLASSIFICATION

Detected as an event coming from the executive	Routine Failure
	Constraint violation
Detected by the Interpreter	Unsatisfied precondition
	Unsatisfied post-condition
	Explicit procedure failure

V. CONCLUSION

We have presented here the major concepts of the AMR Supervision. The architecture is summarized in fig. 7. Breadboards of the Task and Executive level are being developped and will be experimented in 1991, both on HILARE II platform (at Laas) and on a specific AMR platform (at Matra). This demonstration will conclude the phase two of the project. Further work during phase three include the Mission level and Failure recovery implementation. By the end of phase three, two demonstrators (indoor and outdoor) will be built using the concepts presented here.

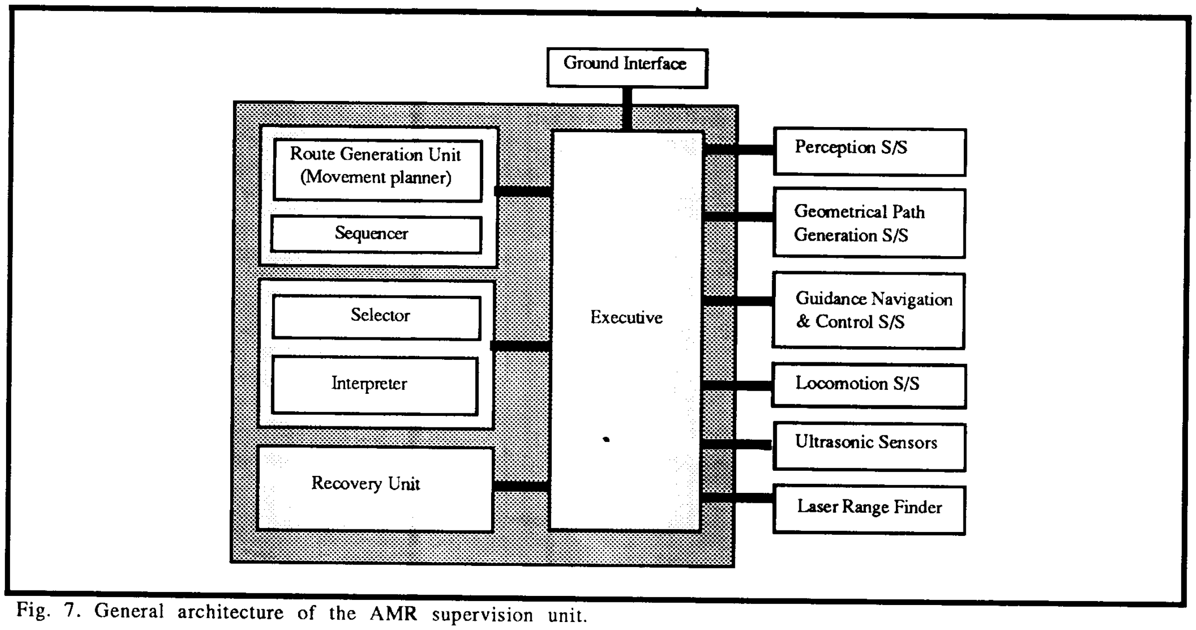

Fig. 7. General architecture of the AMR supervision unit.

REFERENCES

[1] M.Deplanté, D.Mornas, J.L.Ollier. "Le programme AMR de robots mobiles avancés pour la sécurité civile." *Séminaire EC2, Les Robots mobiles*, La Défense, France, 1989.

[2] A. de Saint Vincent."Téléopération au niveau tâche et intelligence artificielle embarquée pour le robot mobile AMR." *Séminaire EC2, Les Robots mobiles*, La Défense, France, 1989.

[3] G.Giralt, R.Alami, R.Chatila : "Autonomy versus teleoperation for intervention robots ? A case for task-level teleprogramming" - in *Proc. 2nd Conf. on Intelligent Autonomous Systems* - Amsterdam , Dec 1989

[4] F. Noreils, R. Chatila. "Control of mobile robot actions." *IEEE Int'l Conf. on Robotics and Automation*, pp 701-712, Scottsdale, Arizona, 1989

[5] R.Alami, R.Chatila, P.Freedman. "Task-level teleprogramming for Intervention robots." *1st IARP Workshop on Mobile Robots for Subsea environment.*

[6] L.Lin, R. Simmons, C. Fedor. "Experience with a task control architecture for mobile robots." in *Technical Report CMU-RI-TR-90-04*, The Robotics Institute, Carnegie Mellon University, 1990.

[7] M. Georgeff : "Reasonning about Procedural Knowledge", in *AAAI*, 1985

[8] R. Alami, R. Chatila, M. Devy and M. Vaisset. "System architecture and processes for robot control." Technical Report (unpublished), LAAS-CNRS, 1990

[9] R. Chatila, R. Feraz De Camargo. "Open architecture design and intertask / intermodule communication for an autonomous mobile robot." *IEEE Int'l Workshop on Intelligent Robots and Systems*, Tsuchiura, Japan, July 1990