Mission Planning and Execution Control for Intervention Robots *

Raja Chatila François Félix Ingrand
Rachid Alami
Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS
LAAS - CNRS
7, Avenue du Colonel Roche 31077 Toulouse Cedex - France
e-mail: {raja, felix, rachid}@laas.fr

Abstract: We describe in this paper a complete architecture for action planning and execution for intervention robots. The architecture is based on a decomposition of the robotic system into a Ground Station that embeds the functions for mission planning and teleprogramming, and an on-board system on the remote robot. This last system is decomposed into two main levels: a "decisional level" that interprets the mission according to the actual execution context and controls its execution by a "functional level" embedding the necessary processings for action and perception. The decisional level makes use of PRS, a procedural reasoning system that will be presented.

1 Introduction

Applications such as planet or submarine exploration require a specific class of robots we call "Intervention robots". Such robots have to perform non-repetitive and time-constrained tasks in ill-known environments, with specific constraints on communication (delays, limited bandwidth). In this context, classical teleoperation as well as telerobotics-like [13] approaches with a human operator in the control loop are not adequate [5] because the environment is not known well enough beforehand to simulate or model it, and it may be too dynamic.

The global functional architecture we propose is composed of an Operator Station and an on-board Robot Control System (Figure 1). The Operator Station includes the necessary functions to allow a human operator to build an *executable mission*, i.e. a mission that can be interpreted by the Robot Control System, and to supervise its execution.

The process of building an executable mission is decomposed into two phases which correspond to two different levels of abstractions and to different planning techniques:

1. a phase called "mission planning" which produces a "mission plan", i.e. a set of (partially) ordered steps that will allow the robot to achieve a given goal.

2. a phase called "teleprogramming" that consists in refining the steps in the mission in terms of tasks that can be interpreted and then executed by the robot.

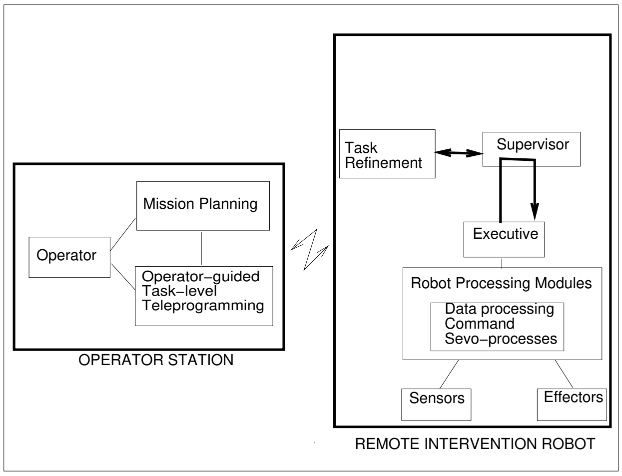

Figure 1: Architecture for Remote Intervention Robots

Mission planning must be performed at the Operator Station because the determination of the goal itself is based on the human interpretation of the working environment. Depending on the nature of the mission and its difficulty, and the amount of information available at planning time, an executable mission can range from an elementary step including every detail in the robot actions, to a complete sequence of steps.

Teleprogramming is also a planning phase as it is based on a projection into the future. However, this planning phase relies upon specialized planners (e.g., a geometric motion planner, a manipulation action planner) that are able to take into account, in an explicit way, the interactions between the robot and its environment. The key aspect for an intervention robot is that this programming phase must be performed using partial and inaccurate information about the robot world, and about the consequences of the robot's actions. This means that the resulting program must rely on sensor-based actions to allow the robot to constantly adapt its execution (e.g. feature tracking) and take appropriate actions when it detects any discrep-

*This paper has appeared in the "Undersea Robotics and Intelligent Control" workshop proceeding, March 2-3, 1995, Lisbon, Portugal.

ancy between between the planned and the actual state of the world.

On-board the remote robot itself, on-line planning and reasoning is necessary to adapt efficiently to the actual execution context because of the sparsity or uncertainty of knowledge. This is for example the case when the environment is gradually discovered by the robot.

We propose an architecture that answers the requirements for robot autonomy composed of a *functional level* embedding the robot sensing and acting capacities, including closed-loop processes, and a *decisional level* based on a layered plan-based framework that integrates interaction between deliberation and action [4].

We shall focus on this aspect of embedded control and supervision, and describe the use of PRS (Procedural Reasoning System, first developed at SRI [6]) as a tool for implementing the interaction between deliberation and reaction. PRS, and its implementation C-PRS that we use, provides tools and mechanisms to represent and execute plans, scripts and procedures, i.e., conditional sequences of actions which can be run to achieve given goals or to react to particular situations.

2 Deliberation and Action in the Robot System

In the robot architecture, the *Supervisor* interacts with the other layers and with the refinement planner. The other layers are viewed as a set of processes which exchange signals with the supervisor. These processes correspond to the actions of the agent as well as events associated with environment changes independent from robot actions.

The refinement planner is given a description of the state of the world and a goal resulting from the teleprogramming phase (e.g., to reach a given location with some constraints on the motion); it produces a plan (e.g., the sequence of motions and perception actions to take). One criterion that should be considered when speaking about planning is the "quality" of the produced plan which is related to the cost effectiveness of achievement of a given task or objective (time, energy,...), and to the robustness of the plan, i.e., its ability to cope with non-nominal situations. This last aspect is one of the motivations of our approach: besides providing a plan, the planner should also provide a set of execution "modalities". expressed in terms of:

- constraints or directions to be used for execution. These directions may be considered as meta-knowledge for the supervision of execution.
- description of situations to monitor and the appropriate reactions to their occurrence in order to prepare a more effective robot behavior to some possible events; such reactions are immediate reflexes, "local" correcting actions (without questioning the plan), or requests for replanning.

The activity of the supervisor consists in monitoring the plan execution by performing situation detection and assessment and by taking appropriate decisions in real time, i.e., within time bounds compatible with the rate of the signals produced by the processes and their possible consequences.

The responsibility of "closing the loop" at the level of plan execution control is entirely devoted to the supervisor. In order to achieve it, the supervisor makes use only of deliberation algorithms which are guaranteed to be time-bounded and compatible with the dynamics of the controlled system.

This execution control is done through the use of the plan and its execution modalities, as well a set of *situation-driven procedures* embedded in the supervisor and independent of the plan. These procedures are predefined at design phase. They can take into account the current goal and plan when they are executed, by recognizing specific goal or plan patterns.

The execution processes are represented by *finite state automata* (FSA). In the FSA we use, the set of allowed external signals correspond to all the actions that can be taken by the supervisor. Similarly, the set of possible internal signals correspond to all environment changes that could be perceived by the supervisor. The execution processes are embedded in robot modules controlled by an "Executive" implemented as a compiled 0^+ rule based-system, that produce a bounded-depth decision tree.

It is important to note that the supervisor is not just an interpreter that would execute a "reactive plan" composed of the plan and modalities produced by the planner. Indeed, the supervisor actually makes evaluations and takes decisions on the way the actions should be executed. Furthermore, the supervisor may decide that a replanning of a task is necessary, and in this sense it also controls the planner considered as a resource, and it may ask for a new mission plan or decision from the control station.

3 Procedural Reasoning for Supervision

Procedural reasoning is a suitable framework for implementing the supervisor part in the robot architecture. Before discussing how it is used we first present a brief description of its main features.

3.1 The Procedural Reasoning System

PRS is composed of a set of tools and methods to represent and execute plans and procedures. These plans or procedures are conditional sequences of actions and goals which can be run or posted to achieve given goals or to react to particular situations. Procedural reasoning differs from other commonly used knowledge representations (rules, frames, ...) as it preserves the control information (i.e. the sequence of actions and tests) embedded in procedures or plans.

A complete description of PRS is given in previous papers [10]. Nevertheless, we find it necessary to provide a brief description of its main components:

a database which contains facts representing the system view of the world and which is constantly and automatically updated as new events appear,

a library of procedures (or scripts), each describing a particular sequence of actions and tests that may be performed to achieve given goals or to react to certain situations,

an intention (or task) graph which is a dynamic set of intentions/tasks currently executing (Fig-

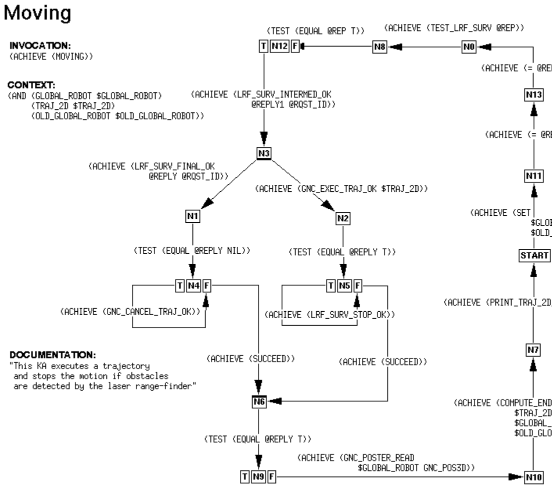

Figure 2: A KA with multiple threads

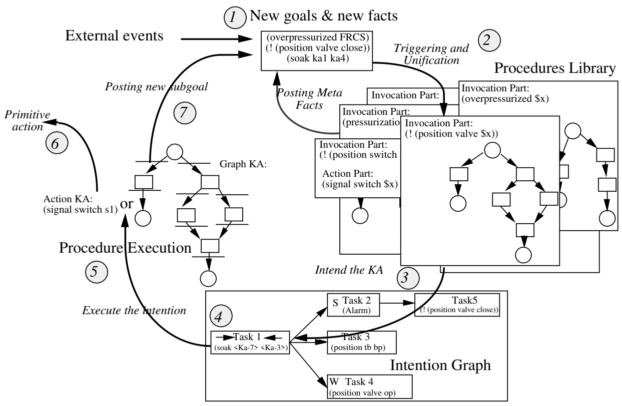

Figure 3: PRS Interpreter

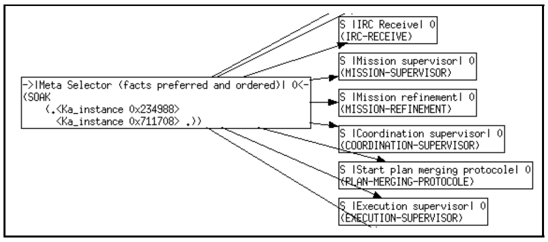

Figure 4: Partial Intention Graph Snapshot

ure 4 shows an example of an intention graph snapshot (from a multi robot experiment performed at LAAS)). Intentions (or tasks) are dynamic structures which execute the "intended procedures", they keep track of the state of execution of these intended procedure, and of the state of their posted subgoals.

There exist various implementations of PRS: SRI PRS [10], ADS PRS, UM-PRS [12]. The one we use, and present in this paper is called C-PRS [1] and is an implementation of PRS in C, under Unix.

3.1.1 KAs, Scripts and Procedures

Knowledge about how to accomplish given goals or to react to certain situations is represented in PRS by declarative procedures historically called *Knowledge Areas* (KAs). (See Figure 2). Each KA consists of a *body*, which describes the steps of the procedure/plan[1], an *invocation condition*, which specifies the goal the KA may fulfill or the events to which it reacts, and a *context* describing under which situations the KA is applicable. Together, the invocation condition and body of a KA express a declarative fact about the results and utility of performing certain sequences of actions under certain conditions [6]. Other piece of information are stored in KA such as facts to conclude or retract upon successful execution or properties which hold user-defined property/value pairs (See Figure 2).

In PRS, *goals* are descriptions of a desired state associated to a behavior to reach/test this state. For example, the goal to position `robot-1` in sea-area-47 is written `(ACHIEVE (position robot-1 sea-area-47))`. The goal to test (without modifying the environment) if the robot `robot-1` is in sea-area-47 is represented `(TEST (position ...))`. The goal to passively wait until the robot `robot-1` gets in sea-area-47 is represented `(WAIT (...))`. The goal to check that the robot `robot-1` stays in sea-area-47 while performing other actions is represented `(PRESERVE (...))`. Similarly, the goal to maintain the robot `robot-1` in sea-area-47 is represented `(MAINTAIN (...))`. For example, the goal to follow a submarine cable while maintaining a safe distance in between another inspection robot could be written: `(& (ACHIEVE (follow cable)) (MAINTAIN (safe-distance @@myself robot-2)))`.

3.1.2 Meta-level KAs

The set of KAs in a PRS application system not only consists of procedural knowledge about a specific domain (See Figure 2), but also includes *meta-level* KAs – that is, KAs able to manipulate applicable KAs, goals, and intentions of PRS itself. The use of meta-level KAs ranges from methods for choosing among multiple applicable KAs, to insure mutual exclusion on critical resources, or to compute the amount of additional reasoning that can be undertaken, given the real-time constraints of the problem domain. To achieve such objectives, these meta-level KAs make use of information about KAs, goals, facts that is contained in the system database or in the *properties* slot of the KA. For example, a meta KA could insure that any procedure invoked because of an external event, such as an external fact, will be intended[2].

3.1.3 The Interpreter

The PRS kernel interacts with its environment both through its database, which acquires new beliefs in

[1] Some KAs, called action KAs, just have an external function call as a body.

[2] This type of KA is usually required in a monitoring and control application where external events are considered more urgent than internal goals.

response to changes in the environment, and through the actions it performs as it carries out its intentions.

As shown in Figure 3, an interpreter manipulates these components. It receives new events (both from outside and from asserted facts) and internal goals (1), selects an appropriate KA based on these new events, goals, and system beliefs (2), places the selected KA on the intention graph (3), chooses a current intention/task among the roots of the graph (4) and finally executes *one step* of the active KA in the selected intention (5). This can result in a primitive action (6), or the establishment of a new goal (7).

An important part of the main loop is the one which finds applicable KAs and selects those which will be intended (2). Basically, this part is composed of one meta-level reasoning loop inside the main loop. The purpose of this inner loop is to determine the successive sets of applicable KAs, in the light of the concluded beliefs on the previous set of applicable KAs. This inner loop stops whenever no applicable KA is found. This means that there exist no more criteria (i.e., an applicable meta procedure) to select among the applicable procedures.

3.2 Implementation of a Supervisor using C-PRS

There are a number of reasons why the PRS approach appears to be well suited for the implementation of a supervisor which satisfies the requirements of the above mentioned architecture. We shall now examine them. Some of these reasons relate to the original capabilities of the PRS and some relate to new features implemented in C-PRS, the specific version we use.

3.2.1 Partial Plan/Script Representation

In PRS, each KA is self-contained, as it describes in which condition it is applicable and the goals it achieves. It usually contains in its "body" tests which condition the proper posting of its subgoals while leaving to the interpreter (and the meta-level KAs) the choice of the adequate procedure to try to satisfy each posted subgoal.

This is particularly well adapted to context based task refinement and to a large class of robot tasks which can be viewed as incremental. The same task corresponding to the achievement of a given goal has to be pursued for a given period of time while its conditions change due to its own execution state or to changes in the environment state or the robot state. A typical task of this type is navigation is a partially known and/or dynamic environment.

For example, in the the experiments we have performed on mobile robot at LAAS, a number of tests and actions must be performed before the robot begins to plan its motion. Nevertheless, the choice of the motion planner used is left to the interpreter and possibly to meta-level KAs which will decide which method is the best in the current situation.

3.2.2 Event and Goal Driven Behavior

KAs can be triggered upon occurrence of events or posting of goals. This is a key feature for implementing a periodic monitoring through a set of situation driven procedures while refining and execution a plan as provided by the planner.

A convenient way to interface the supervisor and the planner is the PRS data base. This will allow for example to express "execution modalities" as facts which will modify, inhibit or awaken some of the situation driven procedures installed in the supervisor.

One can also make use of other mechanisms for implementing more specific monitoring embedded in KA descriptions such guarded task execution.

The KA in Figure 2 illustrates one of these mechanisms. Out from node `N3`, two threads are started, one to execute a trajectory, another one to set a monitoring. If the trajectory executes properly, it then stops the monitoring task which will return without modifying the `@REPLY` variable. Otherwise, if the monitoring detects an obstacle, it returns and the `@REPLY` variable is set to nil, which leads this thread to cancel the trajectory execution.

Other types of monitoring and supervision can be implemented using the `WAIT`, `PRESERVE` and `MAINTAIN` operators.

3.2.3 Reasoning on robot capabilities

The meta level reasoning available under PRS provides a mechanism to control the PRS main loop. Currently, meta level reasoning is mainly used in the KA selection part of the PRS main loop[3].

The meta level reasoning can be used to endow the supervisor with reasoning on robot capabilities e.g., to ensure mutual exclusion of the execution of incompatible procedures, or to implement a preference on the method used to achieve a particular goal when multiple alternatives are given, or to implement some event or procedure based priority mechanism.

3.2.4 Time bounded reaction

As discussed above, one mandatory feature of the supervisor is its ability to react with a guaranteed time time bound compatible with the dynamics of the controlled system.

The algorithms and the main loop used in C-PRS are such that, under some reasonable assumptions, the C-PRS main loop can guarantee an upper bound on reaction time (see [9]). This upper bound is a function of the longest action of the system and of the maximum frequency of event arrival. From this, the user can derive or implement other complex and advanced temporal properties, such as priority mechanisms, deadlines, and so on.

For example, using meta level KAs, it is easy to implement a mechanism which can guarantee that a KA with a particular property will be intended as root of the intention graph (therefore executed before any other KA).

[3] However, it can easily be extended to other parts of the PRS interpreter (for example to react to intention graph changes or to goal failure).

3.2.5 Other useful C-PRS features

C-PRS provides a set of features which allow its effective use in autonomous robot applications. Besides complementary constructions which facilitate programming (e.g. Multi-threading, Elaborated programming environment), C-PRS provides mechanisms to allow its integration with other systems.

A communication library (called mp-prs) has been developed which provides simple but efficient communications over Internet sockets between C-PRS processes and other processes. It is also possible to link user C functions to the C-PRS kernel which will be invoked through an action KA or when evaluating a PRS function or predicate.

Another key aspect is the availability of C-PRS on Unix workstations but also 680X0 or Sparc board under VxWorks enables an easy migration of the application from the workstation to the robot's on-board CPUs.

3.3 Future Developments

There are a number of developments which we think would improve the overall capabilities of C-PRS to handle supervision and control of mobile robots.

An important issue which remains open in the current version of C-PRS is the ability to execute a subset of the loaded KAs with a guaranteed bound on their execution time. This could be achieved using compilation techniques similar to the ones used in KHEOPS [8], a 0^+ rule based-system, that produce a bounded-depth decision tree or using situated automata such as in Rex/Gapps [11].

Another point which appears critical in the two mentioned applications is the ability to handle errors at the "procedure" level. We could implement in C-PRS some kind of error handling mechanism on procedures. Each procedure would then have a number of error handlers which trigger under specified conditions or with particular signals. These handlers could be implemented using the internal mechanisms currently used by the `PRESERVE` and the `MAINTAIN` operators.

Last, we think that the notion of activity, which corresponds to tasks under execution, although more or less present in the intention/task concept, must be further developed to be easily handled by the user [4]. The activity tree is an important representation level in mobile robot control as it allows to send events or signals to activities and propagate them to its children. This notion would improve the control mechanism because it represents more accurately the status of the robot execution tasks.

4 Related Work

The earliest work on procedural reasoning is the study performed by Georgeff *et al* at SRI and described in [7]. One of the major criticisms one can make to this study is that it never reached a point where a real robot ran under the control of SRI PRS. For various reasons, but mainly performances, the procedures were ran with a robot simulator. Moreover, the version of SRI PRS used at that time lacked many of the functionalities which now make an implementation such as C-PRS better suited for this type of application.

More recently, other research laboratories have found interest in using the PRS approach for mobile robot applications. In [12], the authors describe an implementation of procedural reasoning (called UM-PRS) to control an outdoor environment vehicle.

A well known architectural paradigm for robot control is NASREM [2]. However, this is not actually a system, but rather a set of guidelines and mechanisms for implementing hierarchical control systems.

A global architecture, TCA, for autonomous robot control was also proposed by R. Simmons [14]. The architecture features several properties that we find in the one presented above, and is in the same "school of thought" of integrating deliberation and reactivity, as opposed to behavior-based approaches which are rather event-driven [3]. TCA is based on a central control that handles several classes of messages exchanged with specific modules. It manipulates a task tree similar in a way to the intention graph in PRS. TCA also includes explicit temporal constraints on tasks. One important difference with the PRS approach is that the execution of a KA is questioned at each step, with respect to the content of the database. This is a powerful feature for easy and incremental programming and for adapting the execution to the context.

5 Conclusion

We presented in this paper a generic architecture for intervention robots. Mission planning and teleprogramming takes place on a Ground station, and autonomous execution, including mission refinement, is carried out by the remote robot. The robot control system is composed of a a decisional level and a functional level based on a distinction between decision based on global and abstract representations and computations on low level numerical representations. The decisional level is composed of a planner-supervisor pair to ensure a deliberative and reactive behavior. The supervisor makes use of procedural representations for plan execution and goal refinement, and pursues goal-directed tasks while being responsive to changing patterns of events in bounded time. The use of PRS to implement the supervisory part was detailed and its critical features presented. Work on refining this architecture and improving some features in PRS is on-going to better suit control and supervision of autonomous intervention mobile robots.

References

[1] ACS Technologies. *C-PRS Development Environment Manual.* ACS Technologies, 5, Place du Village d'Entreprises, BP 556, 31674 LABEGE Cedex, France, 1992-1994.

[2] J. S. Albus, H. G. McCain, and R. Lumia. NASA/NSB Standard Reference Model for Telerobot Control System Architecture (NASREM). Technical Report NSB Technical Note 1235, National Bureau of Standards, 1987.

[3] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, April 1986.

[4] R. Chatila, R. Alami, B. Degallaix, and H. Laruelle. Integrated planning and execution control of autonomous robot actions. In *IEEE International Conference on Robotics and Automation, Nice, (France)*, 1992.

[5] R. Chatila, M. Devy, and M. Herrb. Perception System Architecture for Environment Modelling and Motion Control of a Mobile Robot. In *IARP 2nd Workshop on Multi-Sensor Fusion and Environment Modelling, Oxford (U.K.)*, September 1991.

[6] M. P. Georgeff and A. L. Lansky. Procedural Knowledge. *Proceedings of the IEEE Special Issue on Knowledge Representation*, 74:1383–1398, 1986.

[7] M. P. Georgeff, A. L. Lansky, and M. Schoppers. Reasoning and planning in dynamic domains: an experiment with a mobile robot. Technical note 380, AI Center, SRI International, Menlo Park, California (USA), 1987.

[8] M. Ghallab and H. Philippe. A compiler for real-time Knowledge-based Systems. In *International Workshop on Artificial Intelligence for Industrial Applications*, Hitachi City, Japan, May 1988. IEEE.

[9] F. F. Ingrand and V. Coutance. Real-Time Reasoning using Procedural Reasoning. Technical Report 93-104, LAAS/CNRS, Toulouse, France, 1993.

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

[11] L. P. Kaelbling. Goals as Parallel Program Specifications. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 60–65, Saint Paul, Minnesota (USA), 1988.

[12] J. Lee, M. J. Huber, E. H. Durfee, and P. G. Kenny. UM-PRS: An Implementation of the Procedural Reasoning System for Multirobot Applications. In *Proceedings of the Conference on Intelligent Robotics in Field, Factory, Service, and Space*, pages 842–849, Houston, Texas (USA), March 1994.

[13] T. Sheridan. Telerobotics. In *IEEE International Conference on Robotics and Automation, Scottsdale, (USA)*, 1989.

[14] Reid G. Simmons. Structured control for autonomous robots. *IEEE Transactions on Robotics and Automation*, 10(1), Feb. 1994.