A Distributed Control System for the CMU Rover

Alberto Elfes Sarosh N. Talukdar

The Robotics Institute
Carnegie-Mellon University
Pittsburgh. PA 15213
USA

Abstract

This paper describes a distributed software control structure developed for the CMU Rover, an advanced mobile robot equipped with a variety of sensors. Expert modules are used to control the operation of the sensors and actuators, interpret sensory and feedback data, build an internal model of the robot's environment, devise strategies to accomplish proposed tasks and execute these strategies. Each expert module is composed of a master process and a slave process, where the master process controls the scheduling and working of the slave process. Communication among expert modules occurs asynchronously over a blackboard structure. Information specific to the execution of a given task is provided through a control plan. The system is distributed over a network of processors each running time operating system kernels local to each processor and an interprocess message communication mechanism ensure transparency of the underlying network structure. The various parts of the system are presented in this paper and future work to be performed is mentioned

1 Introduction

This paper is a progress report on the CMU Rover Project. The CMU Rover [1,2] is an advanced mobile robot, equipped with several different sensors, being developed at the CMU Robotics Institute.

Research in the area of mobile robots can be divided into two broad categories. One major line *of* investigation studies the problems of balance and locomotion [3]. Another line, exemplified by projects such as the Stanford Cart [4], Hilaire, VESA and others [5]. as well as the CMU Rover, concentrates on the problems of developing autonomous or semi-autonomous vehicles which use different kinds of sensors to obtain data about the real world.

From the point of view of application, the latter kind of research paves the way for the development of intelligent autonomous vehicles which could be used for space or sea exploration, or for work in hazardous environments, such as undersea mining and reactor maintenance [6,7]. Additionally, research in mobile robot systems is an important challenge in Artificial Intelligence research [2]. The need to cope with dynamically changing, unpredictable, real-world environments in which processing has to be done in real-time, can lead to the development of more robust and general AI tools.

In this paper, we will give a brief overview of the robot's hardware. describe a Distributed Control System designed for the Rover, and present one control configuration, which is being developed for obstacle-avoidance tasks.

This research *is* being supported by the Office of Naval Research under Contract NCC01481KC8C8 The first author is supported in pan by the Conselho Nacional dc Desenvolvimento Gentifico c Tecnológico - CNPq, Brazil, under Grant 200 986-80: in part by the Instituto Tcnológico dc Aeronáutica ■ ITA. Brazil; and in pan by The Robotics Institute. Carnegie-Mellon University.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies.

2 Hardware Structure

The CMU Rover Project is a continuation of research begun with the Stanford Cart [2.4], a minimal computer controlled mobile camera platform. I'hc Rover [1] is intended to support a variety of AI research in the areas of perception (sensory data processing and understanding), control, real-world modelling, problem-solving, planning and related issues. Real-For this reason, the system is being designed along the following guidelines:

- mechanical, sensor and controller flexibility;
- enough onboard processing capabilities to enable it to function autonomously, but with connections to a remote mainframe system for heavy processing, such as is needed in vision or in long-term planning;
- several different sensor systems which substitute and complement one another.

The Rover is cylindrical, approximately 1 meter tall and 50 cm in diameter, and is equipped with three individually stecrablc wheel assemblies, Each has two wheels, and is powered by two brushlcss samarium-cobalt motors. The whole system is powered by six scaled lead-acid batteries. Hach motor is controlled by a *Motor Processor* (a dedicated MC6805). The *Conductor* (an MC68000) orchestrates the individual Motor Processors to follow a given path. The *Simulator* (another MC68000) uses feedback information from the motors to maintain a dead-reckoning estimate of the robot's position.

The sensors available include a TV camera, a set of sonar ranging devices and an array of proximity sensors. Each is controlled by a dedicated MC6805 (respectively the *Camera. Sonar* and *Proximity Processors).* The *Utility Processor* (also an MC6805) monitors interna conditions of the Rover, such as motor temperature or battery voltage. Additional processors (all MC68000s) arc the *Controller,* responsible for the sensor subsystem; the *Blackboard Processor,* where information relevant to several processes is shared; and the *Communications Processor,* which controls a remote link to a mainframe system (a VAX 11/780 with a high-speed digitizer and an array processor). I'hc processor network is shown in Fig. 1.

3 Software Structure

To permit high-level control, ease of programming and use, and flexibility to perform different kinds of experiments with the Rover, an integrated Planning and Control System is being designed, composed of the following modules (Fig. 2):

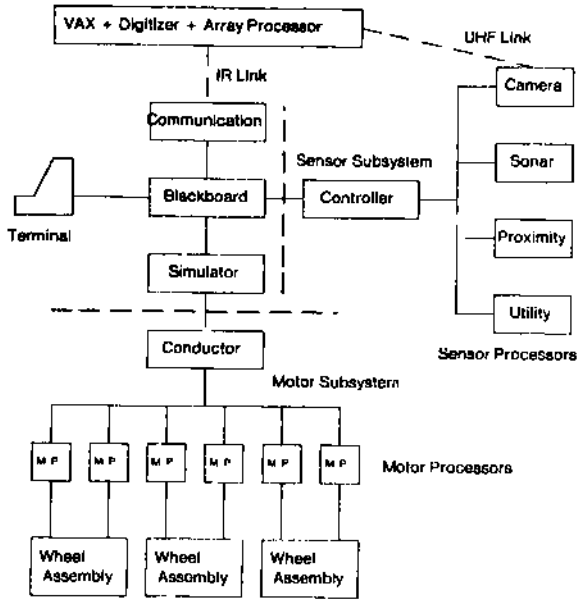

Figure 1: Architecture of the CMU Rover

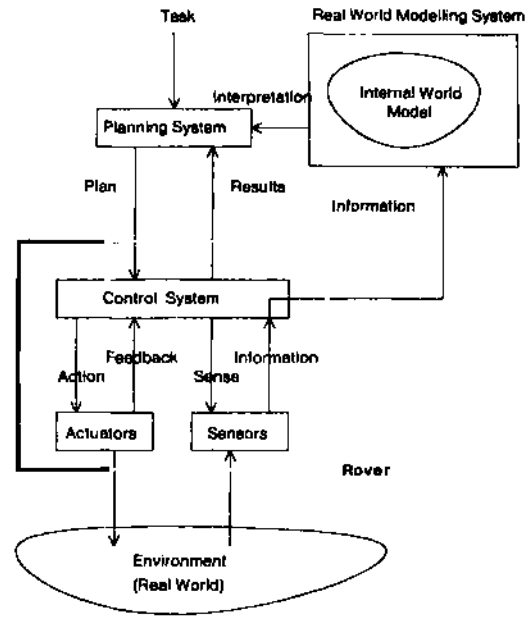

Figure 2: General Structure of the Planning and Control System

- a *Real-World Modelling System,* which interprets and integrates the qualitatively different, partial and sometimes conflicting sensory information, and constructs an Internal Model of the real world environment in which thee Rover is operating;
- a high-level *Planning and Simulation System,* which uses general problem-solving techniques, as well as specific information obtained from the Internal World Model, to develop and monitor the execution of plans to solve a given task;
- a *Distributed Control System,* which is responsible for the execution of a *Control plan,* through parallel and coordinated control of the different sensors and actuators.

Of the three modules, the Distributed Control System was developed first, and is described in the following material.

3.1 The Distributed Control System

3.1.1 Design Considerations

The design of the Distributed Control System was guided by the following goals:

- effective use of the large parallel processing potential existing in the system;
- flexibility of interaction with the different sensors and actuators;
- distributed hierarchical decision-making capabilities;
- flexibility and case of expansion and change of the system;
- graceful integration of a task-specific Control Plan with the several processes which control and operate the Rover, and with sensory and feedback information.

3.1.2 Overview

With the previously mentioned functions and design considerations in mind, the Distributed Control System was designed and a first version was implemented. The system consists of an expandable community of *Expert Modules,* which communicate asynchronously among themselves over a *Blackboard* and perform a given task under the direction of a *Control Plan. Each* Expert Module is dedicated to a particular type of subtask, such as the monitoring of sensors or actuators, problem-solving, or system management. The Control Plan breaks the overall task set to the Rover into a set of subtasks and a set of constraints (the order in which these subtasks must be executed). One Expert Module, the *Supervisor,* dynamically extracts scheduling information on the subtasks from the Control Plan. The Blackboard is a data structure [8] where information relevant to the different Expert Modules is posted, This includes information on the robot's status, relevant sensory and feedback data, scheduling information abstracted from the Control Plan and the degree of progress made towards completing this plan.

The Expert Modules reside in the various processors that make up the network. Local to each processor we have a small real-time operating system kernel that handles physical I/O, schedules processes and manages interprocess and interprocessor communication. A general message-based communication mechanism is used for local as well as remote interprocess communication.

A high level diagram of the Distributed Control System is shown is Fig. 3.

3.1.3 Details

Each expert module is composed of a pair of closely coupled processes: a *"master"* process and a *"slave"* process. The master process keeps track of relevant information on the Blackboard, changing when necessary the status *(run/stop/abort/continue) of* its associated slave process. The master also retrieves needed data from the Blackboard, hands it as input to the slave module, and posts relevant information generated by the slave process on the blackboard. The slave process is the one actually responsible for expert work, such as monitoring sensors,

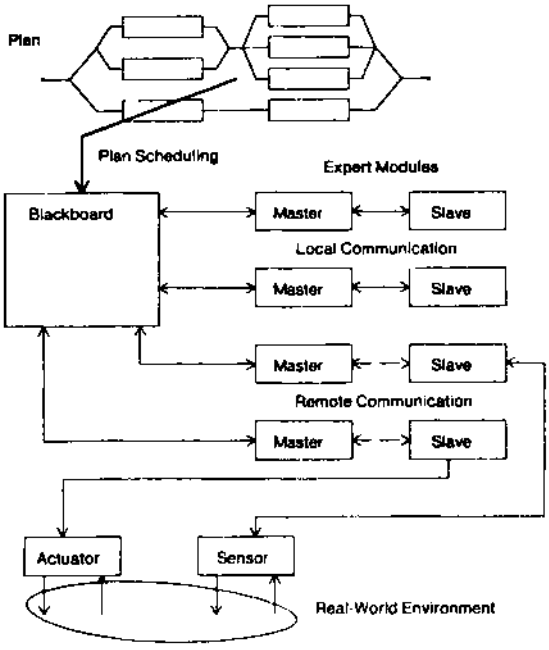

Figure 3: Structure of the Distributed Control System

```
Rover:[Set-Goal(Final-Position);
       <Go:[See-Obstacles;
            <Calculate-Path(New-Position);
             ON (Present-Position ≠ New-Position)
                DO <Move(New-Position);
                    ON (Touch)
                        DO [Stop(Move);
                            Abort(Go);
                            Run(Go)];
                    ON (Wheel-Slipping)
                        DO [Stop(Move);
                            Vel:=Vel/2;
                            Continue(Move)]
                   >
            >
        ]
        ON (Present-Position = Final-Position)
           DO [Stop-Rover]
      >
    ].
```

Figure 4: Example of a Simple Plan: Moving to a Goal and Avoiding Obstacles

handling events, controlling actuators, interpreting sensory information, doing problem-solving, etc. In the typical case, the master process resides in the Blackboard processor (for access efficiency reasons), and the slave process actually resides in a different processor of the network. Communication between them is maintained through the use of a symbolic message passing mechanism, similar to the system proposed in [9]. A message is composed of a set of «namc>, <valuc>) pairs, and has a priority associated to it. Each of the Expert Modules, working asynchronously, maintains an input processing stream and an output processing stream.

The information posted on the Blackboard has a structure in some aspects similar to the structure of messages, being also composed of (<namc>. <valuc>) pairs. For efficiency and logical structuring reasons, the Blackboard is subdivided into different areas, with several levels of abstraction possible within each. In each area related types of information are posted, e.g.. data relating to "movement", "proximity sensors", etc. Actual access to the Blackboard is done only by the *Blackboard Monitor,* to insure the integrity of the posted data. A *Blackboard Scheduler* schedules the master processes to interact with the Blackboard, according to their own priorities and the priorities of data and events being recorded there.

Integrated into the overall control structure of the Rover is the *Control Plan.* A simplified example is shown in Fig.4. The syntax is partially based on [10]. Processes can be specified to execute in parallel (those within < > brackets) or sequentially (those within [ ] brackets). Responses to events arc defined by the use of "ON <cvcnt> IX) <action>" rules. From the Plan, information about parallel and sequential execution of processes, as well as reactions to events, is abstracted, and posted dynamically on the blackboard as the Plan is executed.

Transparency of the physical structure of the network itself is obtained through the underlying support software residing in each processor. Local to each processor we have a real-time operating system kernel , which handles interprocess and intcrprocessor *communication,* takes care of the internal *scheduling* of the resident processes, and handles low-level I/O. *I/O handlers* interface with the actuators and sensors in the system; other routines are responsible for the *translation* of specific low-level actuator and sensor commands to control signals (logical -> physical translation) and of feedback and sensory data to higher abstraction level descriptions (physical -> logical translation). The *mailing* routine constructs messages to be sent to other processors, and handles incoming messages. Messages have *headers* with routing information (source and destination) and *information contents*. The specialized modules residing in each processor can either perform only local functions (in which case they don't interact with the blackboard), or can belong to the set of master-and-slave expert modules. The scheduling of local (in-processor) modules for execution/suspcnsion/abortion/continuation is done on the basis of: a)thc inherent priority of the module; b)the priority of internal events, generated either by software or by hardware; c)thc priority of related incoming messages. The resident kernel also takes care of flow control, buffering, message fragmentation, broadcasting,etc.

3.2 Discussion

The system presented reflects the structure of a community of *cooperating experts*. These experts communicate asynchronously over the processor network, generating and absorbing streams of data. Concomitantly, the system embodies a *hierarchical model* of distributed computation, where the arrangement of the processors can be seen as a tree. I the reflects the fact that the decision-making model is partially hierarchical: control decisions arc. whenever possible, made locally in the processor which is confronted with a problem. Otherwise, the problem or data is broadcast recursively to the next higher level of decision (processors on the path up the tree). Another result from this model is that commands and data can exist within the system at several levels of abstraction. At each level only the necessary degree of detail is present Higher levels are able to deal with the same information in a more abstract form and do not become cluttered with unnecessary details.

The system described is loosely coupled , since the rate of communication between machines, spcciall) from the motor and sensor subsystem levels on upwards, is relatively small. Iliis results from the use of asynchronous processes and the blackboard. This approach can lead 10 higher performance and better adaptability to dynamically changing conditions [11].

The blackboard mechanism has been used by several researchers, e.g. in speech understanding [8]. image understanding [12], and tracking of objects [13]. In our case, due to the multiprocessing network and the need to dynamically respond to the changing conditions of the real-world environment, the role of the "condition pan" of the Hcarsay-II Knowledge Sources [8] was expanded to a more general master/slave relation. In this way, unnecessary or even dangerous actions can be discontinued when necessary. Furthermore, a separate control plan, integrated into the overall structure, and which Hearsay-II lacked, was considered essential in our case. Other researchers have also fell the need to use separate focusing and goal-proposing mechanisms [12,13].

4 Implementation Status

The several physical subsystems of the Rover have been tested and are operational. Final assembly, integration and testing arc now being undertaken. Concomitantly, one specific control configuration, consisting of a set of Expert Modules and the underlying support software, is being developed. This set of modules includes tht Controller, Communication, Sonar, Proximity-Sensor, Camera, Simulation, Blackboard and Utility modules, which implement the corresponding functions mentioned in Section 2; the Movement module accomplishes a trajectory along a path provided by the Path-Planner, the Vision-Processing module extracts visual information about obstacles; the Real-World-Modeller uses information from the Obstacle-Detection module to construct an Internal Model of the environment; the Supervisor dynamically extracts the necessary information from the Control Plan, while the Defaulter fills in the details; the Guardian watches over certain key parameters in the Blackboard to prevent dangerous situations from happening, and the User Interaction module permits communication with a human user. Most of these modules have been implcmcnicd and tested in a simulated environment. The Distributed Control System has proven to be very flexible and adequate. It is now awaiting in; sito testing, running the Rover.

5 Conclusion

Although the development of such complex pieces of hardware and software is fraught with difficulty, there are considerable benefits. Among them, many insights were gained into the problems of control, planning, interaction with the real-world environment and use of a processor network.

The Distributed Control Structure described in this paper fulfills the initial design goals. Future work includes the integration of the information coming from the sensors into a more sophisticated Internal Model, the development of a more elaborate planning system, and research into the problems of automatic knowledge acquisition and learning.

6 Acknowledgments

The authors would like to thank Hans Moravcc, Marc Donner, Zary Segall, Dave McKcown and Peter Hibbard for their suggestions and for interesting discussions.

7 References

1. Moravcc, H.P. "The CMU Rover." Proceedings of the AAA1-82, August 1982.
2. Moravcc. H.P. "The Stanford Cart and The CMU Rover." Proceedings of the IEEE, to appear, 1983.
3. Raibert, M. H. and Sutherland. I. E. "Walking Machines." Scientific American, January 1983.
4. Moravcc. H.P. Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover, PhD dissertation, Stanford University, September 1980. Published as Robot Rover Visual Navigation by UMI Research Press, Ann Arbor, Michigan, 1981.
5. Julliere. M. and Marce, L. Contribution a l'Autonomie des Robots Mobiles Laboratoirc d'Applications des Techniques Electroniqucs Avancecs, Institut National des Sciences Appliquccs, Renncs, 1982.
6. NASA Machine Intelligence and Robotics: Report of the NASA Study Group. Final Report, N81-21769, March 1980.
7. Ayres, R.and Miller, S. The Impacts of Industrial Robots, CMU Robotics Institute TR-81-7, November 1981.
8. Erman, L.D. et al. "The HEARSAY-M speech-understanding system: Integrating knowledge to resolve uncertainty." Computing Surveys 12:2, June 1980.
9. Feldman, J. A. "High Level Programming for Distributed Computing." CACM 22:6, June 1979.
10. Donncr, M.D. "The Design of OWL: A language for Walking." Proceedings of 1983 Sigplan Symposium, ACM, June 1983.
11. Talukdar, S.N., Pyo, S.S. and Giras, T.G, "Asynchronous Procedures for Parallel Processing". Proceedings of Power Industry Computer Applications Conference, 1983. To appear in IEEE Trans, on PAS.
12. Hanson, A.R. and Riseman, E.M. "Visions: A computer system for interpreting scenes." In: Hanson, A.R. & Riscman, F.M. (cds.), Computer Vision Systems. Academic Press, NY, 1978.
13. Corkill, D.D., Lesser. V.R. and Hudlicka.F. "Unifying Data-Directed and Goal-Directed Control: An Example and Fxperiments." Proceedings of the AAAI-82, August 1982.