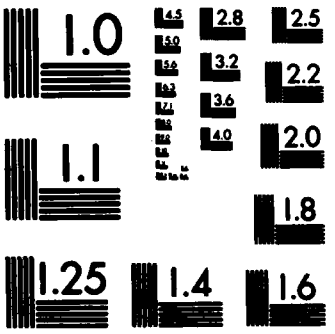
AD-A133 201 A DISTRIBUTED CONTROL SYSTEM FOR THE CMU ROVER(U) 1/1
CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST
A ELFES ET AL. 27 JAN 83 N00014-81-K-0503

UNCLASSIFIED F/G 6/4 NL



# A Distributed Control System for the CMU Rover

Alberto Elfes and Sarosh N. Talukdar

The Robotics Institute

Carnegie-Mellon University

Pittsburgh, PA 15213

27 January 1983



This research is being supported by the Office of Naval Research under Contract N00014-81-K-0503. The first author is supported in part by the Conselho Nacional de Desenvolvimento Cientifico e Tecnologico - CNPq, Brazil, under Grant 200.986-80, in part by the Instituto Tecnologico de Aeronautica - ITA, Brazil, and in part by The Robotics Institute, Carnegie-Mellon University.

# A Distributed Control System for the CMU Rover


## Abstract

*This paper describes a distributed 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 incoming 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 with the aid of a blackboard. Information specific to the execution of a given task is provided through a control plan. The system is distributed over a network of processors. 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 [1,2]. The CMU Rover 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 categories. The first works on the problems of balance and locomotion [4]. The second category, in which this project belongs, concentrates on the use of sensors to obtain data about the robot's surroundings, on the integration of data coming from different sensors, and on the development of strategies for planning the execution of tasks, with the goal of developing autonomous or semi-autonomous vehicles [3,5]. The Rover Project continues research begun with the Stanford Cart [1,2,3], a minimal, computer controlled, mobile camera platform.

From the point of view of application, this 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, in the long range, Moravec [2] argues that research in mobile robot systems is an important challenge in AI research. 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 the distributed control system designed for the Rover, and present a set of expert modules developed for obstacle-avoidance tasks.

## 2 Hardware Structure

The CMU Rover 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. For this reason, the Rover has been designed with enough onboard processing capabilities to enable it to function autonomously. However, it is also connected to a mainframe system, used for heavy processing, to permit higher overall performance and faster response. The robot also has several different sensor systems, which substitute and complement each other.

The Rover is cylindrical, approximately 1 meter tall and 50 cm in diameter, and is equipped with three individually steerable wheel assemblies. Each has two wheels, and is powered by two brushless motors. Each motor is controlled by a dedicated MC6805 *Motor Processor*. The *Conductor* (a MC68000) orchestrates the individual Motor Processors to follow a certain path. The *Simulator* (another MC68000) uses feedback information from the motors to maintain a dead-reckoning best estimate of the robot's position.

The sensors available include a TV camera, an array of proximity sensors and a set of sonar ranging devices. Each is controlled by a dedicated MC6805 (respectively the *Camera*, *Sonar* and *Proximity Processors*). The *Utility Processor* (another MC6805) monitors internal conditions of the Rover, such as motor temperature or battery voltage. Additional processors (all MC68000s) are 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) (Fig. 1).

# 3 Software Structure

## 3.1 Function

The Distributed Control System provides an environment in which the following functions are accomplished:

- Parallel and coordinated control of the different sensors and actuators, and handling of detected events and emergencies.
- Construction of an internal model of the real-world environment in which the Rover is operating. This model is constructed using the data provided by the sensors. This information varies qualitatively from one sensor type to another, and may be incomplete or even conflicting.
- Planning or acquisition (from the user) of strategies for achieving the goals set to the Rover by the user, and monitoring of their execution.

A high level diagram of these activities is shown in Fig. 2.

## 3.2 Design Considerations

The design of the Distributed Control System was guided by the desire to make effective use of the large potential for parallel processing existing in the robot. It should also integrate gracefully the activities mentioned in section 3.1. Finally, the control system should be flexible and easy to change and expand, since modules will have to be added, tested and changed frequently as the system evolves and the Rover is used to handle different kinds of tasks.

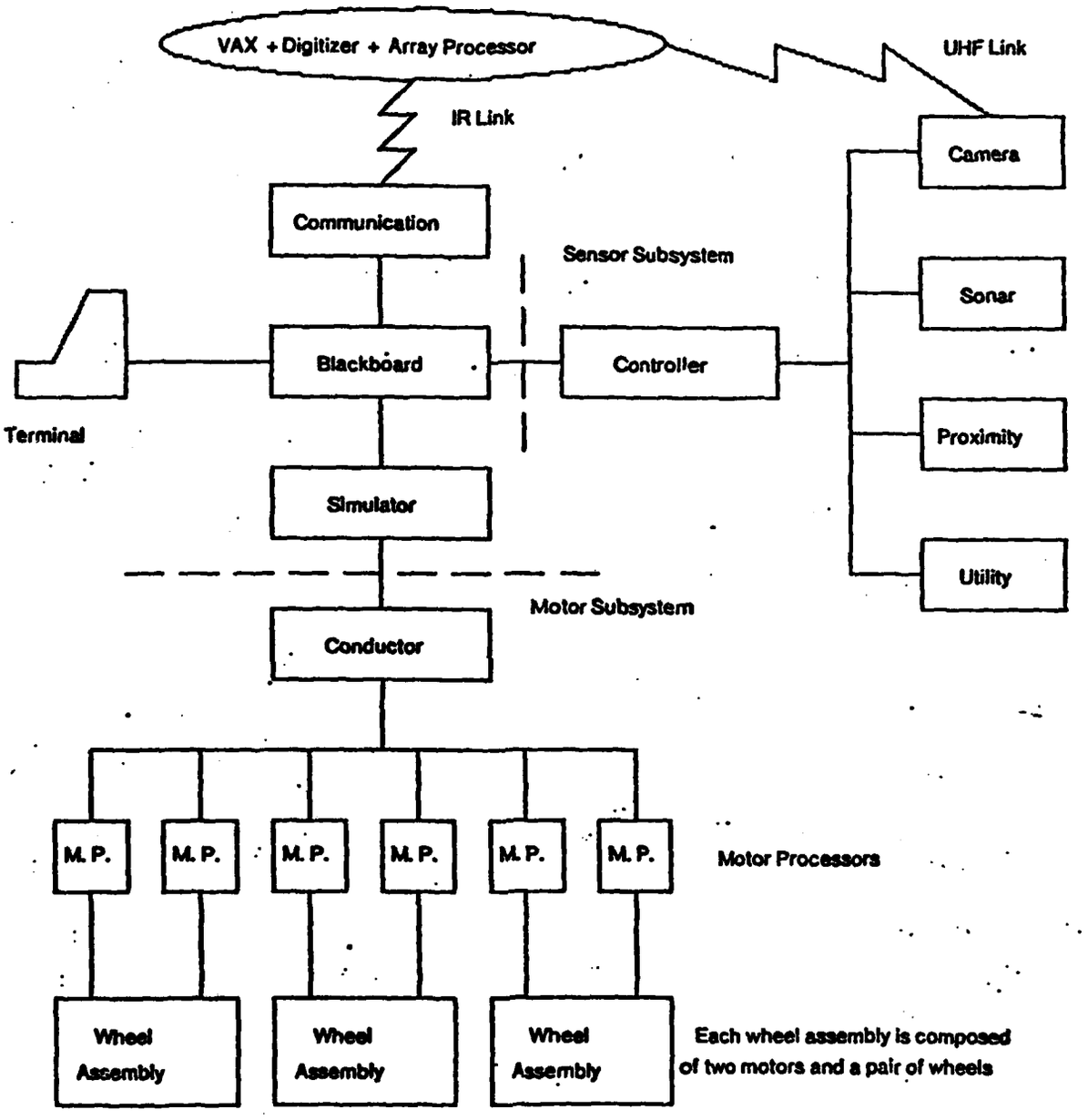



Figure 1: Architecture of the CMU Rover

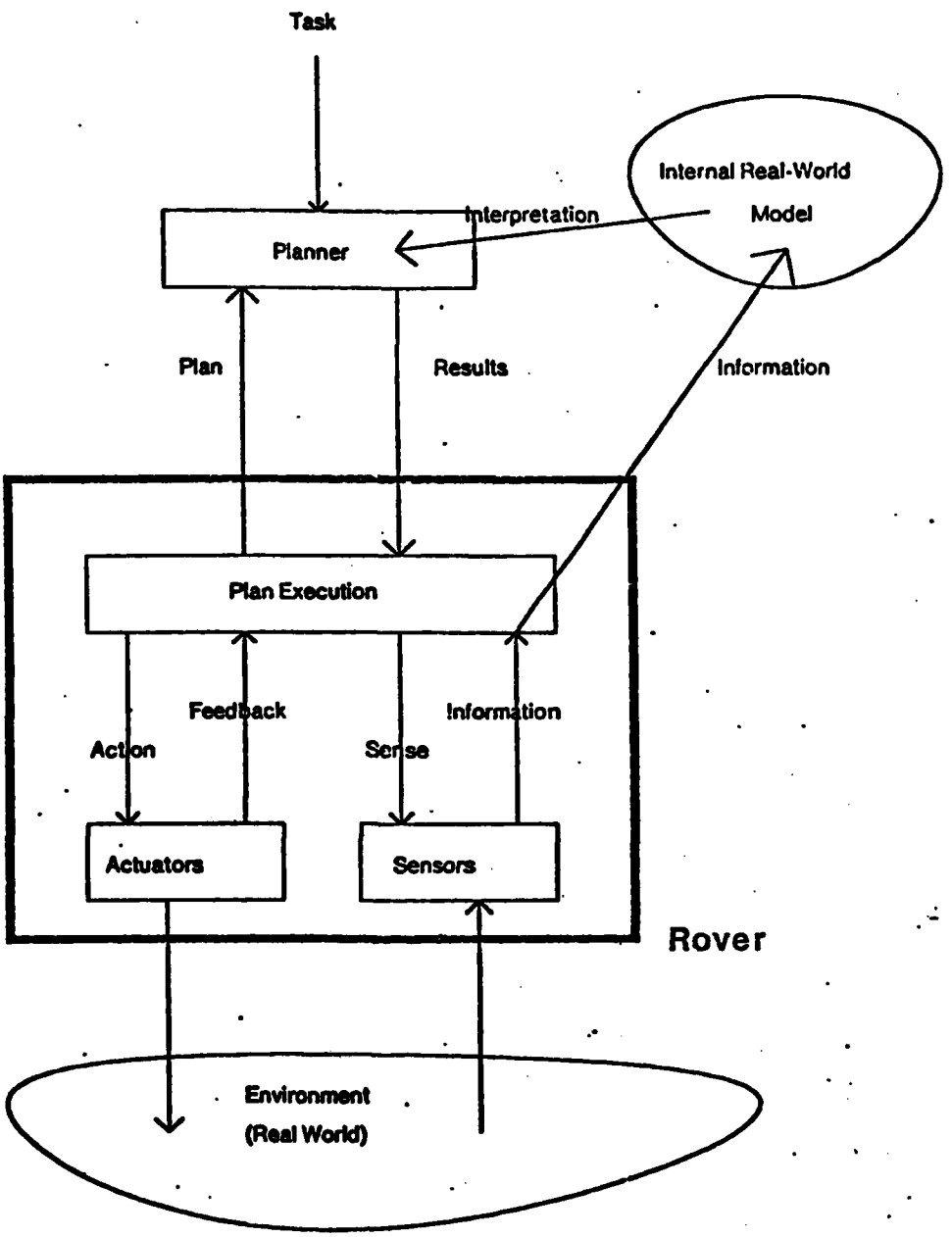



Figure 2: Main Activities of the Distributed Control System

## 3.3 A Distributed Control System

### 3.3.1 Overview

With the previously mentioned functions and design considerations in mind, a distributed control system was designed and implemented. The system consists of a *control plan* and an expandable community of *expert modules*. 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). Each expert module is dedicated to a particular type of subtask. One expert module, the *supervisor*, dynamically extracts scheduling information on the subtasks from the control plan. The expert modules communicate asynchronously among themselves over a *blackboard* [8]. This blackboard contains information on the robot's status, including the latest update of its model of the surrounding environment, information abstracted from the current version of the control plan and the degree of progress made towards completing this plan. A high level diagram of the Distributed Control System is shown is Fig. 3.

### 3.3.2 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 conjugated slave process. The master also retrieves necessary data from the blackboard, hands it as input to the slave module, and posts pertinent information generated by the slave process on the blackboard. The slave process is the one actually responsible for expert work, such as monitoring sensors, handling events, controlling actuators, interpreting feedback information, doing path planning, etc. In the typical case, the master process resides in the blackboard processor (for ease of access), 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 [9]. Each of the expert modules, since it works asynchronously, maintains normally an input processing stream and an output processing stream, as well as a special queue for urgent data (typically commands related to abortion/suspension of the process).

The information posted on the blackboard has a structure in some aspects similar to the structure of messages, being always composed of <name, value> pairs. For efficiency and logical structuring reasons, the blackboard is subdivided according to the general area of activity to which the information is related, e.g., "movement", "proximity sensors", etc. Actual access to the blackboard is done only by the *blackboard monitor*, to insure integrity of the posted data. A *blackboard scheduler* establishes the scheduling of the master processes, according to their own priorities and the priorities of the data and events being recorded on the blackboard.

Necessary to the overall control structure of the Rover is a *control plan*. A simplified example is shown in Fig.4. The notation is based on [13]. Processes can be specified to execute in parallel (those encompassed within < > brackets) or sequentially (those encompassed within [ ] brackets). Response to events is defined by the use of "ON <event> DO <action>" rules. From the plan, the necessary information for parallel and sequential execution of processes, as well as the reaction 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 interprocessor *communication*, takes care of the internal *scheduling* of the resident processes, and controls the low-level I/O handling. *I/O handlers* interface with the actuators and sensors in the system; other routines are responsible for the *translation* of individual low-level actuator and

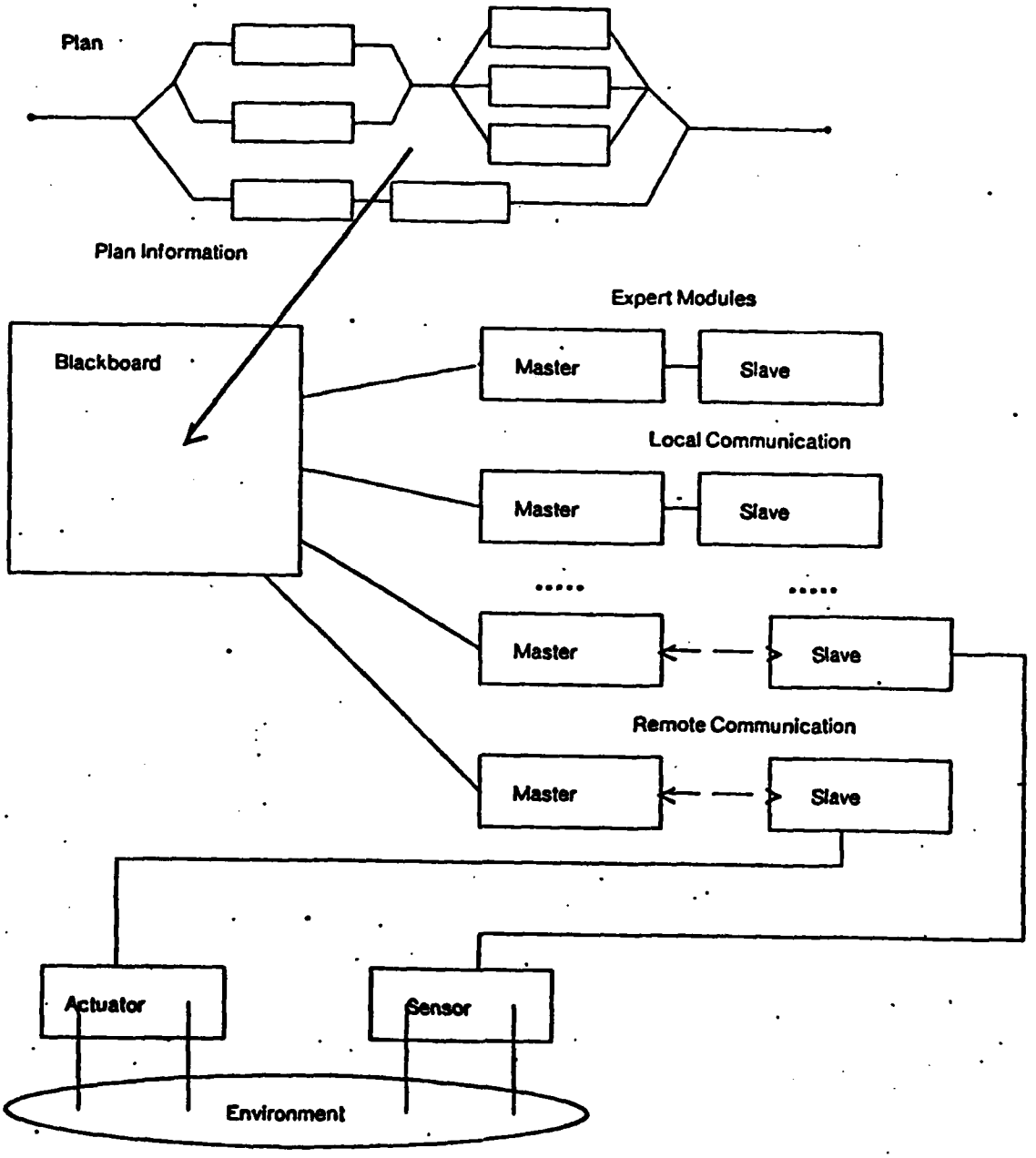



Figure 3: Structure of the Distributed Control System

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

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

sensor commands (logical->physical translation) and feedback and sensory information (physical->logical translation). The *mailing* routine constructs messages to be sent to other processors, and receives incoming messages. Messages have *headers* with routing information (source and destination) and *information*, which is in the form of <name, value> pairs. The dedicated, 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/suspension/abortion/continuation is done on the basis of: a)the inherent priority of the module; b)the priority of internal events, generated either by software or by hardware; c)the priority of related incoming messages. The resident kernel also takes care of flow control, buffering, message fragmentation, broadcasting,etc.

### 3.4 Discussion

The system presented reflects the structure of a community of *cooperating experts*. These communicate asynchronously over the processor network, generating and absorbing streams of data. Concomitantly, they embody a *hierarchical model* of distributed computation, where the arrangement of the processors can be seen as a tree. This reflects the fact that the decision-making model is partially hierarchical: control decisions are, 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, specially from the motor and sensor subsystems level on upwards, is relatively small. This results from the use of asynchronous processes, which are made possible by the use of the blackboard. It leads to higher performance and better adaptability to dynamically changing conditions [10].

The blackboard mechanism has been used by several researchers, e.g. in speech understanding [8], image understanding [11], and tracking of objects [12]. 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 part" of the Hearsay-II Knowledge Sources [8] was expanded to a more encompassing *master/slave* relation. A separate control plan, integrated into the overall structure, and which lacked in Hearsay-II, was also considered essential, in our case. Other researchers have also felt the need to use separate focusing and goal-proposing mechanisms [11,12].

## 4 Implementation

This section describes the present implementation status of the project. The several physical subsystems of the Rover have been tested and are operational. Final assembly and testing are now being undertaken. Concomitantly, one specific control configuration, composed of a set of expert modules, as well as the underlying support software, was developed. This set of modules includes: the *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 *Watcher* prevents dangerous situations from happening, and the *User-Interaction* module permits communication with a human user. These modules have been implemented and tested in a simulated environment. The control system proved very flexible and adequate, and is now awaiting *in sito* testing, running the Rover.

## 5 Conclusion

Although the development of such a complex piece of hardware and software is very demanding, it has already brought many new insights in various different areas, such as control, planning, interaction with the environment, benefits that arise from the use of a processor network, etc. The Distributed Control Structure presented fulfilled the initial design goals, and greater experience with it will be gained when exercising the Rover in different kinds of tasks.

Future work to be done includes the graceful integration of information coming from the sensors into a more sophisticated Internal Model, the development of a more elaborate planning system, and research in automatic knowledge acquisition and learning.

## 6 References

1. Moravec, H.P. "The CMU Rover." *Proceedings of the AAAI-82*, August 1982.

2. Moravec, Hans P. "The Stanford Cart and The CMU Rover." *Proceedings of the IEEE*, to appear, 1983.

3. Moravec, Hans 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.

4. Raibert, M. H. & Sutherland, I. E. "Walking Machines." *Scientific American*, January 1983.

5. Julliere, M. & Marce, L. *Contribution a l'Autonomie des Robots Mobiles.* Laboratoire d'Applications des Techniques Electroniques Avancees, Institut National des Sciences Appliquees, Rennes, 1982.

6. NASA *Machine Intelligence and Robotics: Report of the NASA Study Group. Final Report*, N81-21769, March 1980.

7. Ayres, R. & Miller, S. *the Impacts of Industrial Robots*, CMU Robotics Institute TR-81-7, November 1981.

8. Erman, L.D. et alli "The HEARSAY-II 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. Talukdar, S.N., Pyo, S.S. & Gyros, T.C. "Asynchronous Procedures for Parallel Processing", submitted to PICA-83.

11. Hanson, A.R. & Riseman, E.M. "Visions: A computer system for interpreting scenes." In Hanson, A.R. & Riseman, E.M. (eds.), *Computer Vision Systems.* Academic Press, NY, 1978.

12. Corkill, D.D., Lesser, V.R. & Hudlicka,E. "Unifying Data-Directed and Goal-Directed Control: An Example and Experiments." *Proceedings of the AAAI-82*, August 1982.

13. Donner, M.D. "The Design of OWL: A Language for Walking." Submitted to SIGPLAN 83.


FIL