to appear in: AI Magazine, 16:1, Spring 1995

The Mobile Robot RHINO

Joachim Buhmann, Wolfram Burgard, Armin B. Cremers, Dieter Fox,
Thomas Hofmann, Frank E. Schneider, Jiannis Strikos and Sebastian Thrun

Institut für Informatik III
Universität Bonn
Römerstr. 164, D-53117 Bonn, Germany

Abstract— **RHINO was the University of Bonn's entry in the 1994 AAAI Robot Competition and Exhibition. RHINO is a mobile robot designed for indoor navigation and manipulation tasks. The general scientific goal of the RHINO project is the development and the analysis of autonomous and complex learning systems. This paper briefly describes the major components of the RHINO control software as they were exhibited at the competition. It also sketches the basic philosophy of the RHINO architecture and discusses some of the lessons that we learned during the competition.**

I. General Overview

RHINO, shown in Figure 1, is a B21 mobile robot platform manufactured by Real World Interface Inc. It is equipped with 24 sonar proximity sensors, a dual color camera system mounted on a pan/tilt unit, and two onboard i486 computers. Sonar information is obtained at a rate of 1.3 Hertz, and camera images are processed at a rate of 0.7 Hertz. RHINO communicates with external computers (two SUN Sparcstations) by a tetherless Ethernet link.

The RHINO project is generally concerned with the design of autonomous and complex learning systems [8]. The AAAI competition ended an initial six-month period of software design. Key features of RHINO's control software, as exhibited at the competition, are as follows:

- **Autonomy.** RHINO operates completely autonomously. It has been operated repeatedly for durations as long as one hour in populated office environments without human intervention.
- **Learning.** To increase the flexibility of the software, learning mechanisms support the adaptation of the robot to its sensors and the environment. For example, neural network learning is employed to interpret sonar measurements.
- **Real-time operation.** To act continuously in real-time, any-time solutions [2] are employed wherever possible. Any-time algorithms are able to make decisions regardless of the time spent for computation. The more time that is available, however, the better the results.
- **Reactive control and deliberation.** RHINO's navigation system integrates a fast, reactive on-board obstacle avoidance routine with knowledge- and computation-intense map building and planning algorithms.

RHINO's software consists of a dozen different modules. The interface modules (a base/sonar sensor interface, a camera interface, and a speech interface) control the basic communication to and from the hardware components of the robot. On top of these, a fast obstacle avoidance routine analyzes sonar measurements to avoid collisions with obstacles and walls at a speed as high as 90 centimeters per second. Global metric and topological maps are constructed on-the-fly using a neural network-based approach combined with a database of maps showing typical rooms, doors and hallways. RHINO employs a dynamic programming planner to explore unknown terrain and to navigate to arbitrary target locations. It locates itself by continuously analyzing sonar information. In addition, a fast vision module segments images from two color cameras to find target objects and obstacles that block the path of the robot. RHINO's control flow is monitored by an integrated task planner and a central user interface.

The integration of a dozen different software modules, which all exhibit different timing and response characteristics, requires a flexible scheme for the flow and synchronization of information. The key principles for the design of RHINO's software are as follows:

Figure 1: The RHINO robot of the University of Bonn, Germany .

- **Distributed control and communication.** Each module communicates with several other modules through Ethernet [3]. There is no single control unit, and communication is not centralized.

- **Asynchronous communication.** RHINO's software lacks a central clock. Each of the modules runs independently of the other modules. To resolve conflicts, certain modules (such as the on-board obstacle avoidance module) can take priority over other modules (such as the planner) in determining the robot's motion direction.

- **Software fault tolerance.** RHINO's software is designed to accommodate sudden failures of most of its software components. Almost all modules can be stopped and restarted at any time. Effective mechanisms ensure that restarted modules will immediately obtain the currently available global information.

The next sections present some of the key components of the RHINO approach in more detail: the obstacle avoidance module, the modules concerned with sensor interpretation and map building, the planner and explorer, and the visual routines. The article concludes with a discussion that highlights some of the lessons that were learned during the AAAI competition.

II. Fast Obstacle Avoidance

The obstacle avoidance runs on-board, independent of other software components such as the planner. Every 0.25 seconds, a new velocity and motion direction are chosen according to the most recent sonar measurements. To rapidly adapt to new situations, only the last three sonar sweeps are considered. RHINO can react immediately to changes in the environment and hard-to-see and moving obstacles such as humans.

The obstacle avoidance module controls both the velocity and the motion direction of the robot. At every instant in time, the velocity is determined such that no collision will occur within the next two seconds (2-sec rule). The motion direction is determined based on target points, which are generated by the planner (see below). To reach a given target, the robot can choose among different trajectories on which it will travel with different velocities. RHINO selects its motion direction by maximizing its translational velocity, denoted by v, while minimizing the angle to the target point, denoted by θ.

To determine v, a simplified model of the robot's environment is constructed. Proximity information, obtained from RHINO's sonar sensors, is used to construct a two-dimensional obstacle line field. Every sonar reading is converted to a line in this field, as depicted in Figure 2. To avoid collisions with obstacles, the obstacle avoidance routine considers a variety of circular trajectories, one of which is shown in Figure 2. For each trajectory, the distance between the robot and the closest obstacle line along the projected trajectory is computed. This distance determines the translational velocity v, according to the 2-sec rule. The projected angle to the target point, θ, is calculated for the estimated robot position and orientation after 0.25 seconds. For both values v and θ a smoothed histogram is constructed. Because of the dynamic constraints, only a small number of trajectories are reachable within the next 0.25 seconds, and are consequently considered in the histogram. Finally, the trajectory that maximizes a weighted difference of v and θ is chosen. In order to increase the safety of the robot, a security distance of 10 centimeters is kept to surrounding objects. This security distance is increased to up to 30

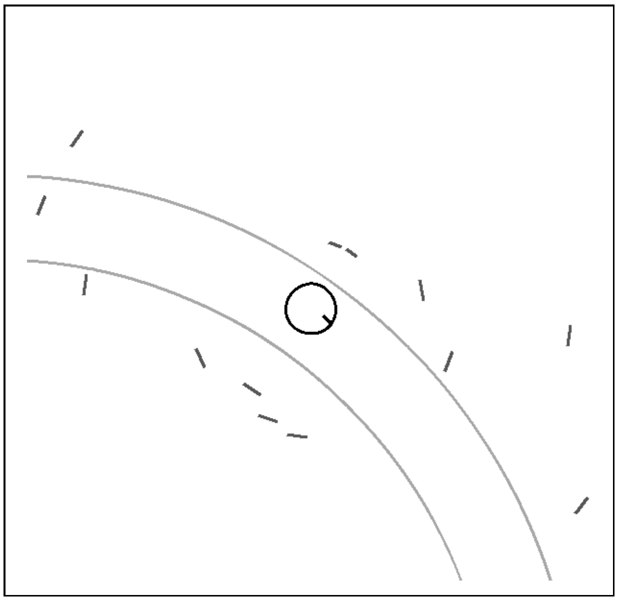

Figure 2: Obstacle line field. Each sonar reading is indicated by a line, centered around the robot. The trajectory, which is finally chosen by RHINO, is also shown.

centimeters, as the robot's velocity increases.

RHINO's obstacle avoidance approach is easily extendible to other sensors. For example, prior to the competition we successfully employed camera information to identify small obstacles on the floor, which block the path of the robot, as described below. Each visually detected obstacle is mapped into a few lines in the obstacle field, very much like the sonar information described above. However, visual information was not used by the obstacle avoidance routine during the AAAI competition, basically because sonar information was fast and accurate enough in the competition ring.

III. MAP BUILDING AND POSITION CONTROL

RHINO's global navigation system builds and utilizes occupancy maps of the robot's environment. More specifically, when traveling through possibly unknown terrain, RHINO interprets its sonar readings to generate a two-dimensional, discrete probabilistic occupancy map. Sonar sensors are interpreted using an artificial neural network, which estimates the likelihood of occupancy of any point in a 3 meter-circle around the robot [7]. Multiple measurements are integrated using Bayesian inference [5]. Figure 3a shows a map which was constructed while we manually steered the robot through the competition arena. This map describes an area of approximately 30×20 meters. The hallways, rooms, large obstacles and doors can clearly be recognized.

To navigate based on global metric information, it is imperative that the robot be able to locate itself accurately in its map. RHINO is equipped with fairly accurate wheel encoders. However, even small angular errors in dead-reckoning can have devastating effects on the internal position estimation. In order to compensate for such error, the robot continuously matches its current sonar readings with its global occupancy map. If a mismatch is found between the occupancy map and the obstacles predicted based on the most recent sonar sweep, the internal position is corrected accordingly. In addition, RHINO registers the angular orientation of walls with respect to its current location to correct more accurately for rotational errors. This mechanism, which rests on the assumption that walls are typically perpendicular or parallel to each other, has been found to be very effective for the detection of rotational errors at the competition as well as in various office environments. If RHINO operates some 30 minutes with velocities of up to 90 centimeters per sec in unknown terrain, the total error is usually smaller than 30 centimeters. Without correcting the dead-reckoning, this error often accumulates as much as 30 meters.

To obtain topological information concerning the location of rooms, doors and hallways, RHINO analyses its metric occupancy map continuously. Walls are identified by thresholding. In addition, a large database of examples of door regions, hallways and rooms (and parts thereof) is continuously matched to assign topological labels to the unoccupied areas in the occupancy map. By analyzing the connectivity of the labeled map, RHINO is able to recognize doors, hallways and rooms. An example of a topologically labeled map is shown in Figure 3b. This map, which is based on the metrical map shown in Figure 3a, subdivides the terrain into 7 rooms and/or hallways (gray) by 9 door regions (white). As is easy to see, most of the rooms and hallways have been identified correctly. In the bottom left corner of that figure, however, a small room has not been identified. This is because due to sensor noise the occupancy map failed to capture a small wall—a problem that may particularly occur with very thin walls, such as those that were found at the competition.

The topological map-analyzer works continuously. At any point in time, it can be queried to output a topological map. However, the quality of the topological maps increases in time. The generation of the labels shown

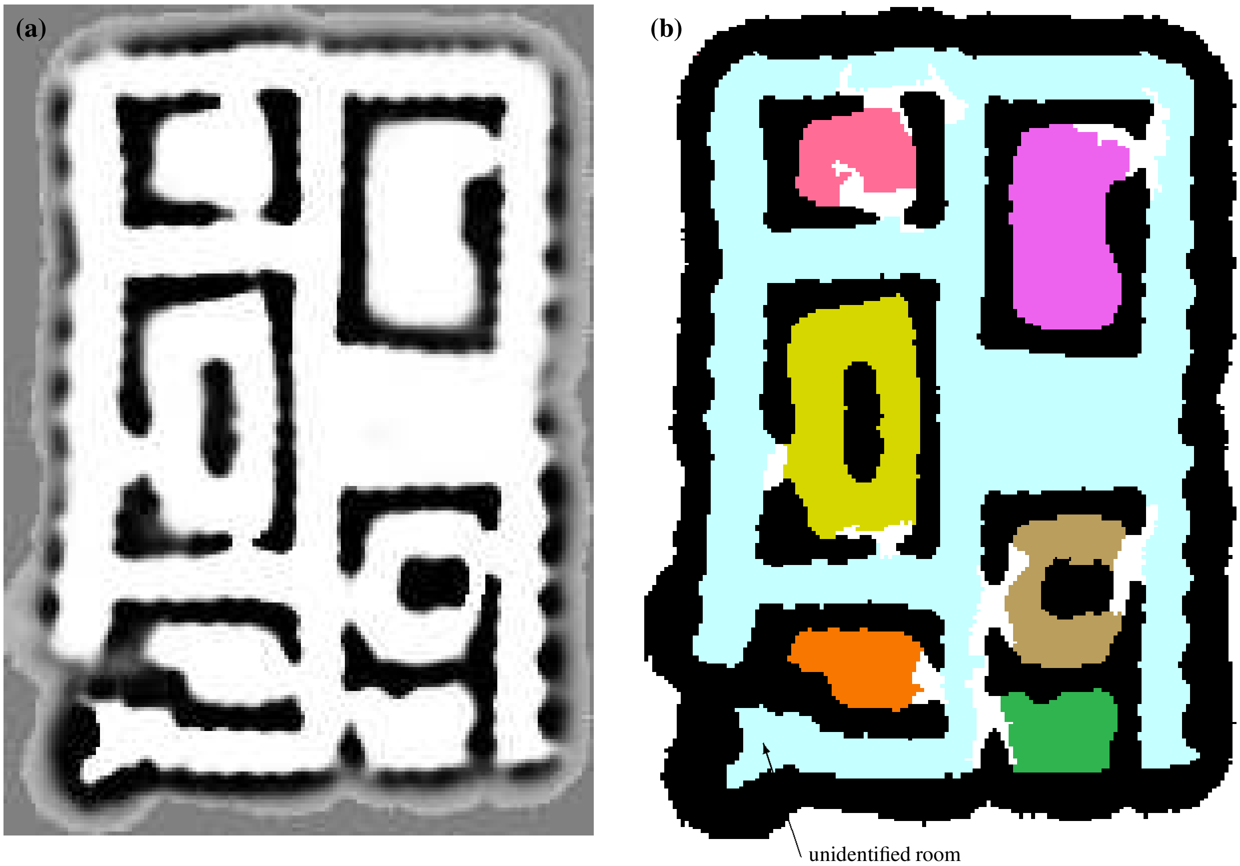

Figure 3: sonar measurements. Bright regions indicate free-space, and dark regions indicate walls and obstacles. Walls and obstacles are enlarged by a robot diameter. (b) Shown here is a topological analysis of the map. Obstacles are shown in black. As indicated by the different shading, the free-space is divided into 7 rooms/hallways (gray) by 9 door regions (white). The arrow points to an unidentified room in the competition ring.

in Figure 3b requires approximately 15 minutes of processing time on a SUN Sparcstation 10. Note that the underlying database of topological examples consists of preselected prototypes based on occupancy maps which were constructed at the University of Bonn prior to the competition.

IV. Planning and Exploration

The previous section presented RHINO's approach to mapping its environment. In this section we describe how occupancy maps are used when controlling the robot. RHINO's planner generates minimum-cost paths to arbitrary goal locations or, as described below, to unexplored regions. These paths are constantly refined and communicated to the obstacle avoidance routine, which then determines the final motion direction and velocity of the robot.

RHINO's main planning engine consists of a dynamic programming routine, which computes trajectories with minimum cost to a goal location [4]. The occupancy map is translated into a cost function, such that occupied territory results in high traversal cost, and free territory in low traversal cost. Dynamic programming propagates path information from the goal(s) to arbitrary locations in the map. Consequently, steepest descent results in a minimum cost path to the "cost-nearest" goal. Control can be generated at any time without any significant computation. However, deliberation time is traded for the quality of the resulting path.

Because occupancy maps are often too inaccurate to generate collision-free motion control, in dynamic environments, RHINO's planner commands only the rough motion direction, which is then finalized by the collision avoidance routine. Consequently, if unmodeled obstacles block the robot's path, the planner is faced with

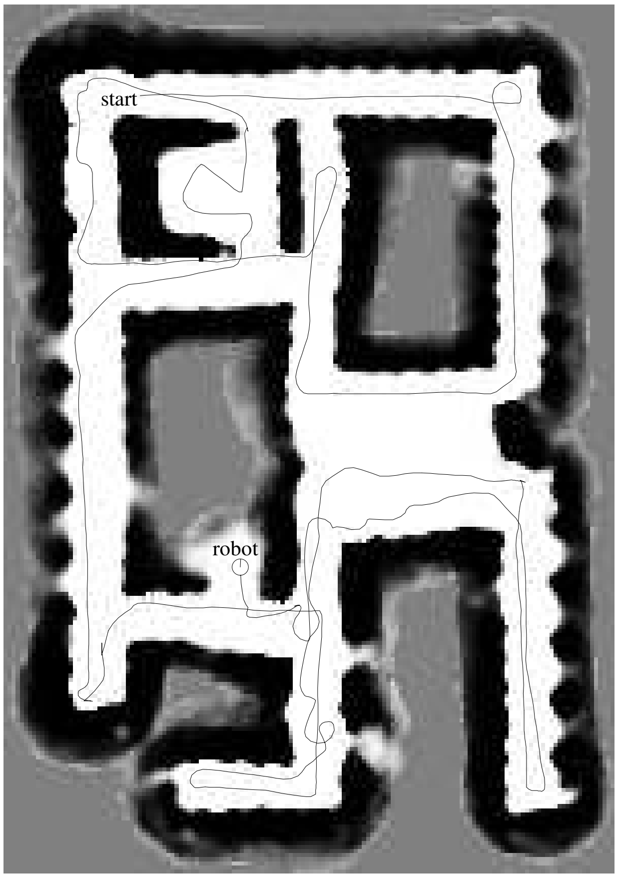

Figure 4: Occupancy map, constructed from scratch during 15 minutes of autonomous robot exploration. The robot's path, which starts at the upper left corner, is also shown.

unexpected robot actions. Dynamic programming preplans for arbitrary robot locations. This is because goal information is propagated for every location in the map, not just the current location of the robot. Consequently, RHINO can quickly react if it finds itself to be in an unexpected location, and generate appropriate motion directions without any additional computational effort. This rapid exception handling ability provides the necessary freedom for the collision avoidance routine to modify actions commanded by the planner at its own will.

In both stages of the competition, RHINO explored and mapped unknown terrain. RHINO's planning mechanism can easily be applied to generate explorative paths, lacking a specific goal point. If the set of goal positions is defined as the set of positions, for which no map information is available, RHINO moves straight to the unexplored. Figure 4 illustrates the path of some 15 minutes autonomous robot exploration in the competition ring. In this prototypical example, the main hallways have already been traversed, and RHINO continues to explore the unexplored rooms. RHINO's speed at the straight-line segments of the exploration path was generally between 50 and 90 centimeters per second. Further details on planning and exploration can be found in [7].

V. Vision and Object Recognition

The images from the color camera system are the input to a four-stage vision system, which solves two different tasks. First, it has to recognize important objects typically found in the environment (e.g., objects in an office environment). Second, it supplies valuable information for the robot navigation task by providing local occupancy maps to the map builder and obstacle locations relative to the robots position to the collision avoidance module. This second task, however, was not performed during the final runs at the competition. Here, map building and obstacle avoidance relied solely on sonar information, which turned out to be sufficiently reliable in the competition ring.

In the first stage of low-level processing, images are low-pass filtered and sub-sampled, to reduce the data transfer via the radio link and to pre-process the image for the next stage. This process is performed on one of the on-board i486 computers. Sampling in space (image size) and in time (frame rate) is done dynamically, dependent on the actual velocity of the robot. Thereby the transmission channel capacity is allocated in a task-driven way.

The second processing stage is done by an image segmentation algorithm which partitions the transmitted image into homogeneous, connected regions (*cf.* Fig. 5). Homogeneity is measured by a dissimilarity measure between neighboring image sites (pixels or blocks of pixels). For reasons of efficiency, the dissimilarity measure is restricted to a weighted squared sum of color and luminance differences between sites (with an additional threshold). Formally, the segmentation task can be described as a minimization problem of a cost function, which sums up the local inhomogeneities of all regions for a given partition. To achieve real-time performance without the need for special hardware, the segmentation is implemented by a fast region-merging scheme. The decision whether or not two neighboring regions should be merged depends on a comparison between the current costs and the costs after merging.

The third stage takes the segmented image as an input and

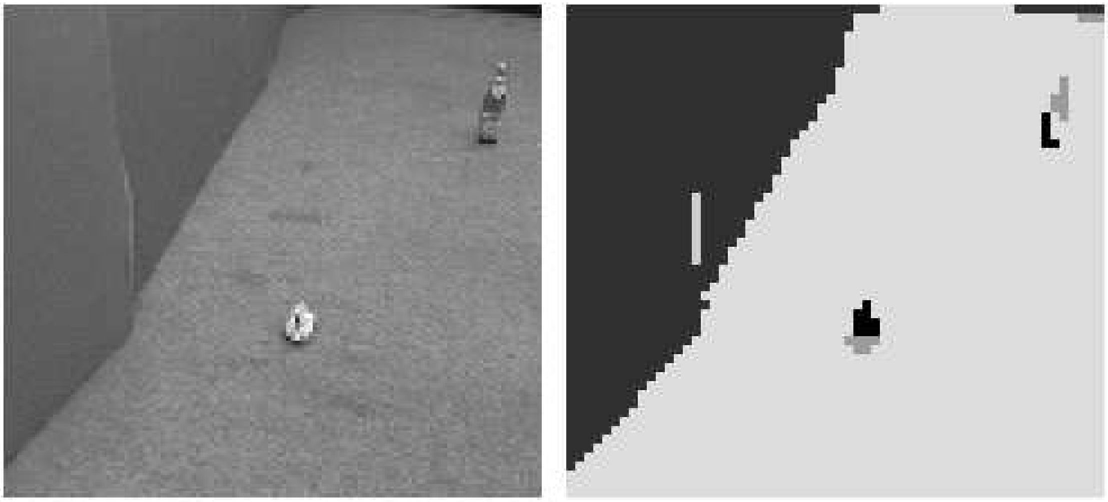

Figure 5: (a) Raw image and (b) segmented image in coarse resolution (9 segments).

seeks to identify and label certain elements of a typical indoor scene, e.g., the floor, walls and doors. Important further information for both navigation and object recognition can be derived: The distances and sizes of all objects or regions located on the floor are calculated based on knowledge of the position, viewing angle, etc. of the cameras. This distance and size information for walls and objects is incorporated into the occupancy map.

At the top of RHINO's vision processing architecture, a feature-based object recognizer detects objects of interest in the environment. The recognition module is able to learn from labeled examples of feature vectors, extracted from example images. For every type object, a Gaussian model (mean and covariance matrix) is estimated according to the maximum likelihood principle. Typical features are the normalized mean and the variance of object luminance, the mean and the variance of object color (hue and saturation), and geometrical features like the absolute size, width and height of the object, estimated based on the object location calculated in the third stage of the vision system. At the competition, we used between 30 and 50 training examples to model each of the 7 object classes. These example images showed different objects varying in object distance, lighting conditions and the choice of the class representative. The assignment to a class is done by minimizing the Mahalanobis distance to the class mean. If the likelihood is below a certain threshold, the candidate object is not accepted as a member of a known class and is considered unclassified.

Once a target object has been found, its location is communicated to the planner and other modules concerned with task and motion control.

VI. Results and Discussion

This paper surveys the software architecture of the fully autonomous RHINO robot, as it was exhibited at the AAAI Robot Competition and Exhibition. RHINO is controlled by a dozen software modules which work and communicate asynchronously. Special emphasis is put on real-time operation, learning, and the integration of reactivity and global map knowledge. RHINO does not require prior knowledge on the locations of walls/obstacles, nor on the topology of its environment.

In the first stage of the competition ("office delivery"), RHINO had to move to a designated target location (see [6] for a detailed description of the competition). This stage consisted of three trials, two of which counted for the final score. Because we are specifically interested in navigation without prior information, we attempted to use the first trial for exploring and mapping the competition ring, and only the remaining two trials for the delivery task. However, although RHINO traveled fast, we learned that the arena could not be explored completely without prior information in the allotted time. Consequently, we had to "buy" a metric map for subsequent trials.

One of the major problems we encountered at the competition was RHINO's unreliable radio link. Unpredictable radio communication, possibly based on interference with other radio links, caused RHINO's on-board operating system (Linux, a PC-version of Unix) to suspend obstacle avoidance for periods of 10 seconds or more. In the second and third trial of stage one, RHINO suffered from severe communication failures and collided repeatedly with walls. Because of this, the first stage of the competition could not be completed, and RHINO was excluded from the finals of this stage.

The communication problem was fixed in the second stage of the competition by moving the radio link closer to the competition ring. In this stage ("cleaning-up an office"), RHINO was required to find and fetch objects like soda cans and paper wads, to pick them up, and to drop them in groups of three into a nearby trash bin. Since RHINO is currently not supplied with a manipulator, it indicated its intention to pick up and to drop objects by voice. RHINO used essentially the same exploration routines as in the first stage but at a reduced speed. In addition, the visual routines described above were employed for the identification of obstacles. At the competition, RHINO found most of the objects in the starting room and then continued to clean up the hallway. Here RHINO scored second, defeated only by a collaborating team of three robots, described in a different article in the same volume [1].

The AAAI competition ends an initial six-month period of software engineering. RHINO's software is generally applicable to autonomous navigation in indoor environments. In the future, RHINO shall operate 24 hours a day, interrupted only by battery charging. Our main scientific interest is the study and the design of autonomous, complex learning systems, which in the domain of robotics includes adaptive approaches to sensory processing and lifelong robot learning [8]. We are currently implementing various learning techniques that allow RHINO to adapt to new situations, and to acquire new skills necessary for achieving a broad variety of tasks.

Acknowledgment

Some of the low-level software (TCX [3], device drivers for the speech board and the cameras) were provided by Carnegie Mellon University, which is gratefully acknowledged. We also acknowledge the steady and helpful support by Real World Interface Inc. Travel to the competition would not have been possible without a generous travel grant by AAAI, and the invaluable assistance by Peter Lachart, Wolli Steiner and Peter Wallossek. One of the authors (T.H.) was partially supported by the Ministry of Science and Research of the state North Rhine-Westphalia.

References

[1] T. Balch, G. Boone, T. Collins, H. Forbes, D. MacKenzie, and J. C. Santamaria. Io, Ganymede and Callisto – a multiagent robot janitorial team. *AI Magazine*, 16(1), Spring 1995. (this issue).

[2] T. L. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceeding of Seventh National Conference on Artificial Intelligence AAAI-92*, pages 49–54, Menlo Park, CA, 1988. AAAI, AAAI Press/The MIT Press.

[3] C. Fedor. TCX. An interprocess communication system for building robotic architectures. programmer's guide to version 10.xx. Carnegie Mellon University, Pittsburgh, PA 15213, December 1993.

[4] R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press and Wiley, 1960.

[5] H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, Summer 1988.

[6] R. Simmons. The 1994 AAAI robot competition and exhibition. *AI Magazine*, 16(1), Spring 1995. (this issue).

[7] S. Thrun. Exploration and model building in mobile robot domains. In *Proceedings of the ICNN-93*, pages 175–180, San Francisco, CA, March 1993. IEEE Neural Network Council.

[8] S. Thrun. A lifelong learning perspective for mobile robot control. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, September 1994.