ROBOT NAVIGATION RESEARCH AT CESAR*

Deanna L. Barnett, G. de Saussure and F. G. Pin

Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory
P.O. Box 2008
Building 6025, MS-6364
Oak Ridge, Tennessee 37831-6364

A. Sabharwal and S. S. Iyengar

Department of Computer Science
Louisiana State University
Baton Rouge, Louisiana 70803-4020

Invited Presentation: The 28th IEEE Conference on Decision and Control, Hyatt Regency, Tampa, Florida, December 13-15, 1989.

* Research sponsored by the Engineering Research Program of the Office of Basic Energy Sciences, of the U.S Department of Energy, under contract No. DE-AC05-840R21400 with Martin Marietta Energy Systems, Inc.

ROBOT NAVIGATION RESEARCH AT CESAR*

Deanna L. Barnett, G. de Saussure and F. G. Pin

Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory
P.O. Box 2008, Bldg. 6025, MS-6364
Oak Ridge, Tennessee 37831-6364

A. Sabharwal and S. S. Iyengar

Department of Computer Science
Louisiana State University
Baton Rouge, Louisiana 70803-4020

ABSTRACT

A considerable amount of work has been reported on the problem of robot navigation in known static terrains. Algorithms have been proposed and implemented to search for an optimum path to the goal, taking into account the finite size and shape of the robot. Not as much work has been reported on robot navigation in unknown, unstructured, or dynamic environments.

A robot navigating in an unknown environment must explore with its sensors, construct an abstract representation of its global environment to plan a path to the goal, and update or revise its plan based on accumulated data obtained and processed in real-time.

The core of the navigation program for the CESAR robots is a production system developed on the expert-system-shell CLIPS which runs on an NCUBE hypercube on board the robot. The production system can call on C-compiled navigation procedures. The production rules can read the sensor data and address the robot's effectors. This architecture was found efficient and flexible for the development and testing of the navigation algorithms; however, in order to process intelligently unexpected emergencies, it was found necessary to be able to control the production system through externally generated asynchronous data. This led to the design of a new asynchronous production system, APS, which is now being developed on the robot.

---

*Research sponsored by the Engineering Research Program of the Office of Basic Energy Sciences, of the U.S. Department of Energy, under contract No. DE-AC05-840R21400 with Martin Marietta Energy Systems, Inc.

This paper will review some of the navigation algorithms developed and tested at CESAR and will discuss the need for the new APS and how it is being integrated into the robot architecture.

I. INTRODUCTION

A large number of algorithms have been proposed, and are still being developed, for the navigation of mobile robots [1,2]. These algorithms address the problems of autonomous robot navigation in known and unknown, static and dynamic environments. Early research at the Center for Engineering Systems Advanced Research (CESAR) was concerned with experimenting with well-known algorithms, first on computer graphics simulations, then with the series of mobile robots HERMIES (Hostile Environment Robotic Machine Intelligence Experiment Series). New algorithms were also developed [3-7]. It is clear that no navigational method is optimal for all environments.

In the most recent CESAR robots, HERMIES-IIB [8] and HERMIES-III [9], the navigation is controlled by a production system developed on the expert system shell CLIPS [10], which runs on the nodes of a NCUBE hypercube parallel processor on board the robot. The rules of the production system can read the sensor data and address the robot's effectors; the rules can also call on C-coded navigation procedures. With this approach the rule base remains small,fast,efficient, and easy to understand and maintain; the C-coded routines are also small and easy to maintain, and each routine is for a specific function. Modules have been developed for navigating from a map, for searching for a target, and for navigating using edge detection. Currently under development and testing are modules for artificial potential fields and for navigation exploration. Therefore, in principle, the production system can select the most appropriate navigation routine, or change from one mode of navigation to another when changes in the environment so require.

However, in order to deal efficiently with unanticipated events, it was found desirable for the robot to be able to input and rapidly respond to externally generated asynchronous data. This requirement led to the design of a new type of production system, the Asynchronous Production System (APS), now being installed on the robots.

The autonomous mobile robots HERMIES-IIB, shown in Fig. 1, and HERMIES-III have been described in recent papers [8,9]. Several navigation algorithms developed at CESAR have also been previously discussed [3-7]. In this paper we present some more recent navigation research, and we outline the new Asynchronous Production System now under development.

In the next section we describe an algorithm for the navigation of HERMIES-IIB in a known environment where dynamic obstacles might unexpectedly block a path. This algorithm was developed for application to robot surveillance tasks in buildings with a-priori given maps. In the third section we discuss some current research in navigation in an unknown and dynamic environment. Finally in the last section we outline the new Asynchronous Production System.

II. ROBOT NAVIGATION IN A KNOWN AREA WITH UNEXPECTED OBSTACLES

In one of CESAR's applications project, the feasibility of using an expanded version of a path planning algorithm in the development of an Automatic Alarm Testing Robot (AATR) was investigated. Use of an AATR for security tasks implies the requirement that no communication exists between the robot and the security systems or personnel except in the event of a site emergency or mechanical failure aboard the robot. Furthermore, the execution of the actual alarm testing needs to be carried out in narrow, pre-defined time windows. Thus the path planning algorithm for such a robot should be capable of not only taking the robot from one point to another in a known environment with unexpected (a priori unknown) obstacles in real time, but also of time-synchronization with the security systems.

A computer program, written in C, was developed to run in the real time environment on the HERMIES-IIB NCUBE host computer. The program algorithm uses a breadth-first technique for searching a network defined in the form of connected points or nodes. Although it requires saving a relatively large number of partial paths in memory, the breadth-first algorithm allows finding of all feasible paths from one point of the network to another (essential in the event of "unexpectedly" blocked corridors) and, for fairly regular networks such as those needed to represent building corridors, tends to provide near-optimum solutions first during the search, which can be utilized when stringent computing time constraints exist.

The major function of the program is to plan and control "missions" of the robot with the following features:

- finds feasible paths and generates primitives (actual instructions) for the robot to go from each point (*node*) to the next, in the order specified in the mission file,

- if a particular goal on the mission is inaccessible, for example, if all paths to the goal point (called the *Goal_node* in the program) from the start point (called the *Start_node* in the program) are blocked, the program skips that part of the mission and plans the rest,

- if an *unexpected obstacle* (defined as an obstacle unknown to the robot a priori) is encountered when the robot is executing a particular portion of the path, (i.e., detected using sonar or other sensors) the robot returns to the nearest node it just passed, and replans another feasible path (if it exists) from this node to the *Goal_node*; it also updates the world map to indicate that portion of the path where the obstacle was encountered is blocked,

- if the collision avoidance sensors detect an obstacle, the program checks whether the path to the next node is clear (i.e., the obstacle is actually beyond the next node in the planned path), and if so, the program moves the robot to the next node,

- the motion of the robot is continuous until it reaches a turn (as opposed to stop-and-go at each *node*),

- the program has a sense of *time-synchronization*, i.e., if a particular portion of the mission is not completed in a specified time (known as the mission time) the program will skip the present *Goal_node* and proceed to execute the rest of the mission,

- at regular intervals, the program plans a stop of the robot at a "self-location" station where the robot recalibrates its odometric parameters using its camera vision algorithms and an icon at a known location on the wall.

Demonstrations of the HERMIES-IIB capabilities with this program were performed in the CESAR laboratory using semi-realistic floor plans with simulated corridors and alarm stations. A layout of one of these demonstration experiments is shown in Fig. 2.

The mission was to start from *node 23* and go to *node 1*, do *self-location* by 'snapping a picture' of the icon, test a 'heat detector' alarm located at *node 6* followed by another *self-location* at *node 29*, go the *node 50* to test a 'doorway intrusion alarm,' another *self-location* at *node 48*, test a 'motion detector' at *node 45* and finally return to node 23.

The path followed by the robot is given by the 'dotted' line. Note that when the robot was traversing the path between *node 33* and *node 40* an *unexpected obstacle* (in the form of a human being) was placed in the path. On detecting the obstacle, the robot back-tracked to the node it just passed i.e., *node 33*; planned another path to the *Goal_node* (in this case *node 50*) and proceeded along the new path. It also marked the path between *node 33* and *node 40* as being blocked in its memory.

Again an *unexpected obstacle* was placed along the new path in the corridor between *nodes 92* and *98*. The robot promptly replanned another path to the *Goal_node 50* and marked the above corridor as being blocked. At this point in time, the robot calculated that reaching node 50 within the allowed time was impossible. The node 50 subgoal was then removed from the mission phase. The rest of the mission was executed without any incident [18].

III. ROBOT NAVIGATION IN AN UNKNOWN DYNAMIC ENVIRONMENT

A sophisticated module is under development at CESAR to create a robust world model integrating a priori knowledge about the environment with signals from sonars, vision, a laser range-finder and other sensors [11]. While this module is under development, other modes of navigation are being researched using only sonars and a dead-reckoning system based on monitoring the wheel encoders. Indeed, in a dynamic environment or in an environment with sparse convex obstacles, it may not be necessary to keep a world model since the environment keeps changing and since a simple reflexive behavior may be sufficient to avoid obstacle collisions and to reach the goal [12].

The main navigation mode now being implemented with HERMIES-IIB is based on edge detection. The only sensors used, except for the wheel encoders used for dead-reckoning, are sonars. There are twenty Polaroid transceivers mounted in five 2 × 2-arrays with each array operating as a phased-array reducing the effective beam width from thirty degrees for the individual transceiver to about twenty degrees for the phased-array. The five arrays are located on the perimeter of the octagon shaped rotating sensory platform and spaced on center thirty-nine degrees apart (see Fig. 1). A single wide angle sonar is mounted on the front of the robot chassis for collision detection.

Autonomous navigation with sonars is the subject of much current research [13,14]. The main difficulties are due to the poor directionality of the sonars and to the specular reflections of the beam at certain angles. In our approach, the robot remains stationary while taking a scan of the environment. The sonar returns are then analyzed to decide where the robot should move next. Only sonar returns corresponding to a distance smaller than fifteen feet are considered. When a narrow opening between obstacles is detected, the robot moves closer to determine the smallest width of the opening before attempting to navigate through. If several possible openings are detected, the distance to the goal through each opening is estimated, and the opening corresponding to the shortest estimated distance to the goal is examined first. In each forward movement of the robot the front fixed sonar and two lateral sonars are continually activated, and if an unexpected obstacle is detected within two feet, a hardware interrupt stops the robot, sets a flag in the expert system indicating that the last forward move was not completed, and returns to the expert system the distance actually moved, as obtained from the wheel encoders. The flag fires an alert condition which triggers a diagnostic rule base in CLIPS. This rule base requests information from the sonars about the unexpected obstacle: sonar readings are taken at fixed intervals and at different elevations to determine if the obstacle is moving or not and if it is smaller or taller than 3 ft. These features of the obstacle determine the actions of the robot requested by the expert system, according to the diagnostic rule given in Table 1.

The interaction between the expert system and the C-coded procedures can be illustrated here: the decisions on what the robot should do next, when a sonar scan is needed and what type of scan, are made by the production system rule base; the scans and motions are C-procedures called by the rules.

Another navigation mode being researched is based on the Artificial Potential Field approach described by O. Khatib [15]. An attractive virtual force towards the goal is combined with repulsive virtual forces from the directions where obstacles are detected by the sonars. The resultant force is used to determine the speed and direction of motion of the robot. An algorithm was developed and tested on a computer simulation and is being ported to the robot HERMIES-IIB. It is well known that under certain conditions the robot may be "trapped" in a local minimum of the force field. In such cases, the expert system will recognize that the robot is not progressing towards the goal and will cause the robot to switch to another mode of navigation.

IV. THE ASYNCHRONOUS PRODUCTION SYSTEM

Two major deficiencies have traditionally impaired the use of production systems for real-time applications: (a) their slow execution speed and (b) their insensitivity to asynchronous, external events (modification of a working memory element or reading of new information only happens when specifically called for by the rule currently processed in the "execute" module). Both of these conditions are impediments to the robot's ability to detect unexpected events in its environment and rapidly respond to these events, sometimes according to a reflex-model behavior. In an attempt to remedy these problems, research at CESAR has focussed on the development of Asynchronous Production Systems (APS) [16,17] for the HERMIES robot expert systems "brain."

APS [1] retain the rule-based architecture of traditional production system tools, thus providing their convenience and usefulness, but incorporate fundamental changes in the data structures and the execution mechanism so as to provide the needed capabilities of perceiving (inputing) and processing real-time events occurring in the environment, and thus generating prioritized, event-driven responses to real-time stimuli. The basic architecture of the APS is represented in Fig. 3. Two major features differentiate the APS from conventional production systems: the parallelism and concurrent execution of the match, select, and execute modules of the system; and the existence of an additional data set corresponding to the "external input" to the system.

The three concurrent, asynchronous modules of the APS' inference engine communicate with each other only through four globally-available *data sets*: the *working memory*, the *external input*, the *conflict set* and the *selected rule*. A particular module is activated when appropriate changes are established in its input data-set(s). On activation, the module performs its designated task and then terminates by making appropriate changes in its output data-set. Since module-activation is solely dependent on the changes occurring in the global data-sets, simultaneous changes in the data sets can activate different modules simultaneously and thus the modules should be capable of concurrent execution if implemented on a separate processor in a multiprocessor environment. An important manifestation of such an execution mechanism is that as long as only the working memory changes, the APS inference engine emulates the conventional recognize-act cycle. However, as soon as there are changes in the external input, the APS inference engine starts concurrent and asynchronous execution of the modules and thus provides the event-driven response to real-time stimuli.

The external input data set contains a set of variables called *external input elements* (EIE) each of which typically corresponds to a dynamic real-time source of data, external to the production system. The external input cannot be manipulated by the RHS actions of the productions and is affected only by changes in the corresponding external data-sources. The syntax of the external input element is the same as that of the working memory element and the attributes of these elements reflect the corresponding physical values indicated by the real-time data-sources.

The match module is activated when there is a change in either the working memory (due to the execution of a RHS action) or in the external input (due to a real-time data input) the new set of satisfied productions are entered into the conflict set and any previously satisfied instantiations which no longer match their data-set elements are deleted from the conflict set. Implementation of the APS on HERMIES-III will create a more powerful testbed for examining the ability of the APS to handle the unexpected events in real time.

V. CONCLUSIONS

Our experience with navigation research using the mobile robot HERMIES-IIB suggests that the use of a production system to make high-level, mostly heuristic, decisions coupled to low-level algorithmic procedures, produces an efficient architecture for the control of the robot and for research into different modes of navigation. The use of the newly developed Asynchronous Production System enables the robot to attend in real time to "unexpected events" and emergencies. After further extensive experimentation with the smaller robot HERMIES-IIB, and after completion of a module for constructing a detailed world model by sensor integration, the system will be ported to HERMIES-III, a much larger mobile robot capable of performing human-size manipulations [9].

ACKNOWLEDGEMENTS

N. Sreenath designed, developed and implemented the module for robot navigation in a known area. P. Graham, J. Graham and V. Hedge implemented the APS on simulation. The authors are grateful to the CESAR team members for suggestions and help with the robot hardware. S. M. Killough designed and constructed the robot dead reckoning and other robot primitives.

REFERENCES

[1] For a review, see for instance S.H. Whitesides, "Computational Geometry and Motion Planning" in "Computational Geometry," G. T. Toussaint, ed. Elsevier Science Publishing Co, New York (1985), and references therein.

[2] Special Issue of Computer, on Autonomous Intelligent Machines (June 1989) edited by S. S. Iyengar and R. L. Kashiap.

[3] C. C. Jorgensen, W. R. Hamel and C. R. Weisbin, "Exploring Autonomous Navigation," *Byte* pp. 223-235 (January 1986).

[4] C. R. Weisbin, G. de Saussure and D. W. Kammer, "Self-Controlled: A Real-Time Expert System for an Autonomous Robot," *Computers in Mechanical Engineering*, Vol. 5, 2 pp. 12-19 (September 1986).

[5] N. S. V. Rao, S. S. Iyengar and C. C. Jorgensen, "Terrain Model Acquisition by a Finite-Sized Mobile Robot in Plane," Proceedings of 1987 IEEE Robotics and Automation Conference, Raleigh, N.C., Vol. 1 pp. 283-288 (1987).

[6] B.L.Burks et al., "Autonomous Navigation, Exploration and Recognition," Winter 1987 Issue of *IEEE Expert* pp. 18-27.

[7] N. S. V. Rao, S. S. Iyengar and G. de Saussure, "The Visit Problem: Visibility Graph-Based Solution," *Proceedings 9*, pp. 1650-1655, 1988 International Conference on Robotics and Automation, Philadelphia, PA, April 25-29, 1988.

[8] C. R. Weisbin et al., "Autonomous Mobile Robot Navigation and Learning," *Computer*, pp. 29-37 (June 1989).

[9] W. W. Manges et al., "Development of the HERMIES-III Mobile Robot Research Testbed at ORNL," Proceedings of the ANS Third Topical Meeting on Robotics and Remote Systems, Charleston, S.C., March 13-16, 1989.

[10] C. J. Culbert, *CLIPS Reference Manual*, NASA/Johnson Space Flight Center, Houston, Texas (1987).

[11] M.Beckerman et al., "World Modeling and Multi-Sensor Integration for a Mobile Robot," Proceedings of the ANS Third Topical Meeting on Robotics and Remote Systems, Charleston, S.C., March 13-16, 1989.

[12] T. L. Anderson and M. Donath, "Synthesis of Reflexive Behavior for a Mobile Robot Based Upon a Stimulus-Response Paradigm," Mobile Robots III of the *SPIE*, Vol. 1007 (November 1987).

[13] M. Beckerman et al., "Spatial Reasoning in the Treatment of Systematic Sensor Errors," Proceedings of the SPIE Conference on Sensor Fusion: Spatial Reasoning and Scene Interpretation, Cambridge, MA, November 6-11, 1988.

[14] R. Kuc and B. Barshan, "Navigating Vehicles through an Unstructured Environment with Sonar," Proceedings of the 1989 IEEE International Conference on Robotics and Automation, Scottsdale AR, May 14-19, 1989.

[15] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," 1985 IEEE international Conference on Robotics and Automation, St. Louis, Missouri, March 25-28, 1985.

[16] A. Sabharwal et al., "Asynchronous Production System for Real-Time Expert System," Proceedings of AVIGNON 88: Eighth International Workshop on Expert Systems and their Applications.

[17] A. Sabharwal et al., "Asynchronous Production Systems," *Knowledge-Based Systems*, Vol 2, No. 2, pp. 117-127 (1989).

[18] N. Sreenath, "Path-Planning in a Known Environment with Unexpected Obstacles: Potential Application to an Automatic Alarm Testing Robot," ORNL/TM-10726, Cesar-88/09 (May 1988).

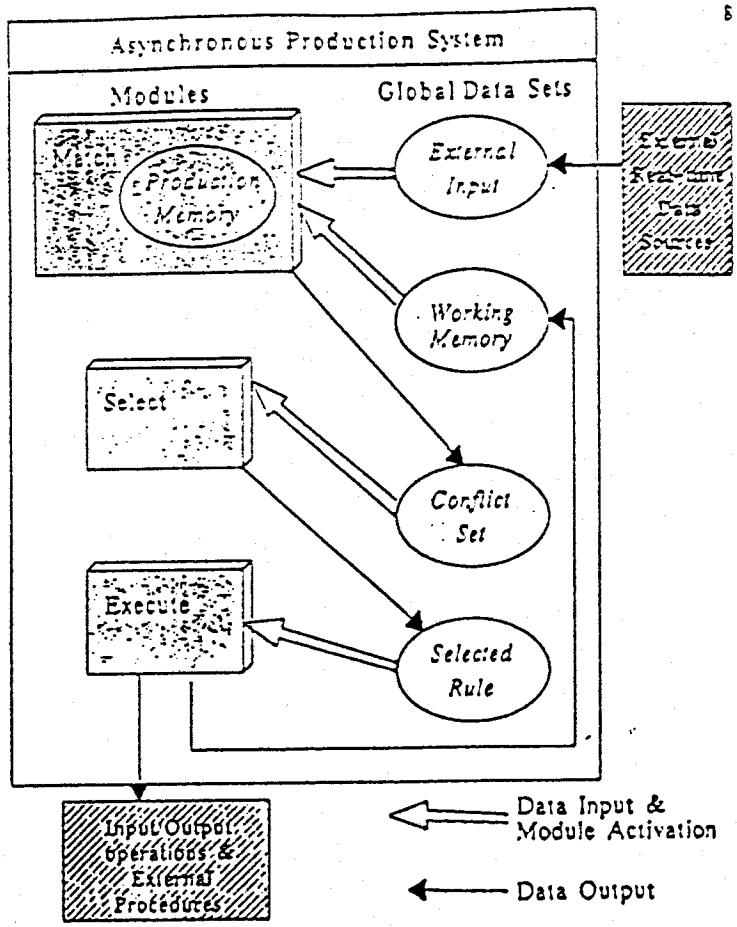
Table 1. Diagnostic Rules for Unexpected Obstacles

Obstacle Characteristic	Action to Take
1. Stationary and over 3 ft. tall.	1. Start with navigation algorithm from the current position.
2. Stationary and less than 3 ft. tall.	2. Move forward to the obstacle, pick it up with the manipulator arms, put it to one side and proceed to the original destination. Anything shorter than 3 ft. is guaranteed to be light enough to lift.
3. Has moved out of the way.	3. Proceed to the original destination.
4. Is moving away from the robot.	4. Wait for the obstacle to clear the path and proceed to the original destination.

Fig. 1. HERMIES-IIB Robot

Fig. 2. Layout of demonstration experiments

Fig. 3. The execution mechanism for the APS



APS Execution Mechanism		
Modules	Input Data Sets	Output Data Sets
Match	Working Memory External Input	Conflict Set
Select	Conflict Set	Select Rule
Execute	Select Rule	Working Memory

Fig. 3. The execution mechanism for the APS

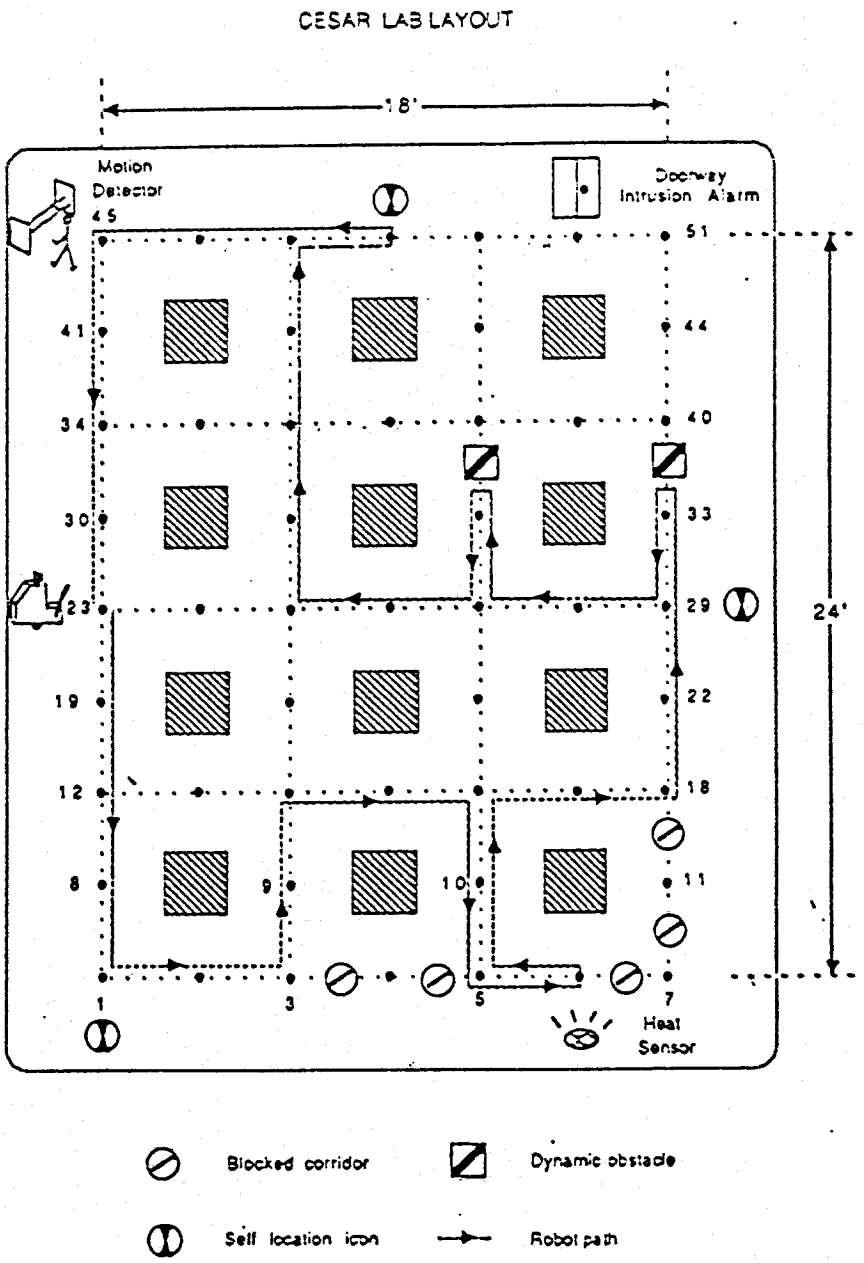


Fig. 2. Layout of demonstration Experiments

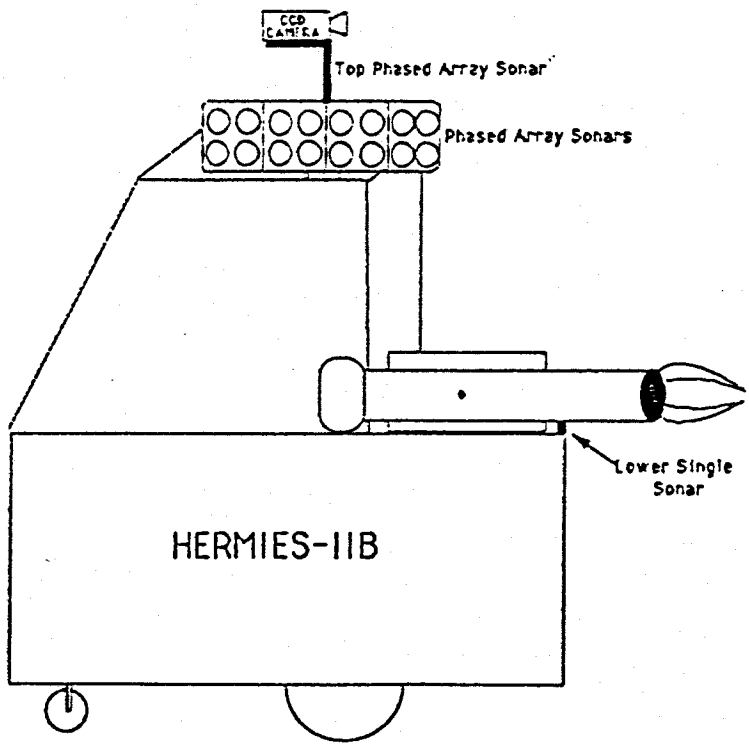


Fig. 1. HERMIES-IIB ROBOT