ornl

OAK RIDGE NATIONAL LABORATORY

MARTIN MARIETTA

ORNL/TM-11518
(CESAR-90/15)

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES

3 4456 0348640 3

Proceedings of the 1989 CESAR/CEA Workshop on Autonomous Mobile Robots (May 30–June 1, 1989)

Karen S. Harber
François G. Pin

OAK RIDGE NATIONAL LABORATORY
CENTRAL RESEARCH LIBRARY
CIRCULATION SECTION
4500N ROOM 175
LIBRARY LOAN COPY
DO NOT TRANSFER TO ANOTHER PERSON
If you wish someone else to see this report, send in name with report and the library will arrange a loan.

OPERATED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-8401, FTS 626-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.
NTIS price codes—Printed Copy: A19 Microfiche A01

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-11518
CESAR-90/15

Engineering Physics and Mathematics Division

PROCEEDINGS OF THE 1989 CESAR/CEA WORKSHOP ON AUTONOMOUS MOBILE ROBOTS (May 30 – June 1, 1989)

Karen S. Harber and François G. Pin
Center for Engineering Systems Advanced Research

DATE PUBLISHED — March 1990

Workshop Sponsored by the
Engineering Research Program
Office of Basic Energy Sciences
U.S. Department of Energy
and by the
French Commissariat à l'Energie Atomique
Saclay, France

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
operated by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE–AC05–84OR21400

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES

3 4456 0348640 3

TABLE OF CONTENTS

Page

ABSTRACT v

WORKSHOP PROGRAM AND AGENDA vii

WORKSHOP PHOTOGRAPH x

LIST OF PARTICIPANTS xi

SESSION I: EXPERIMENTAL DEMONSTRATIONS 1

ARES Base - Case Demonstration (A. Cossic, J. P. Nominé, A. Malavand, and M. Detoc) 3

Robust Performance of Multiple Tasks by an Autonomous Robot (M. Beckerman, D. L. Barnett, J. R. Einstein, J. P. Jones, P. F. Spelt, and C. R Weisbin 23

Simulation Tools for Benchmarking Robotics Systems: The ARES Approach (G. Dejonghe) 53

SESSION II: PERCEPTION FOR MOBILE ROBOTS 63

SYMPATI 2: An Advanced Vision System for Robotics (D. Juvin) 65

Concurrent Computer Vision on a Hypercube Multicomputer (J. P. Jones) 93

First Experiment With a 3D Sensor (G. Ermont and D. Galley) 115

Machine Vision Research Using a Laser Range Camera (F. J. Sweeney) 129

SESSION III: ENVIRONMENTAL MODELING 139

Sensor-Based Mapping (M. Beckerman) 141

Environment Modeling for Robotics Simulation (G. Dejonghe, A. Cossic, and D. Chaigne) 161

SESSION IV: PLANNING AND NAVIGATION 191

Action Planning Applied to Execution of High Level Controls (D. Schmit) 193

Job Planning and Execution Monitoring for a Human-Machine Symbiotic System (L. E. Parker) 205

3-D Simulation of a World Modeling and Navigation Method (J. P. Nominé) 223

Navigation Algorithms for HERMIES (G. de Saussure and D. L. Barnett) 257

Mobile Robot Navigation Based on Geometrical Approach (P. Favre and J. C. Fanton) 262

TABLE OF CONTENTS (cont.)

Page

SESSION V: RECENT DEVELOPMENTS AND NEW TEST BEDS 293

Autonomous Learning at a Control Panel (P. F. Spelt) 295

Learning by an Autonomous Robot at a Process Control Panel (P. F. Spelt, G. de Saussure, E. Lyness, F. G. Pin, and C. R. Weisbin) 321

Neural Networks and Graph K-Partitioning (L. Hérault and J.-J. Niez) 349

Computer-Aided Teleoperation (CAT) (G. Fraize) 383

APPENDIX A - Preliminary Report on the Base Case Scenario Definition in Preparation of the 1989 CESAR/CEA Workshop on Autonomous Mobile Robot 401

ABSTRACT

The U.S. Department of Energy's (DOE) Center for Engineering Systems Advanced Research (CESAR) at the Oak Ridge National Laboratory (ORNL) and the Commissariat a l'Energie Atomique's (CEA) Office de Robotique et Productique (OREP) within the Directorat à la Valorization are working toward a long-term cooperative agreement and relationship in the area of Intelligent Systems Research (ISR). Specialist meetings and periodic workshops focussed on specific topics of ISR are among the planned cooperation activities. This report presents the proceedings of the first CESAR/CEA Workshop on Autonomous Mobile Robots which took place at ORNL on May 30, 31 and June 1, 1989.

The purpose of the workshop was to present and discuss methodologies and algorithms under development at the two facilities in the area of perception and navigation for autonomous mobile robots in unstructured environments. Experimental demonstration of the algorithms and comparison of some of their features were proposed to take place within the framework of a previously mutually agreed-upon demonstration scenario or "base-case." The base-case scenario described in detail in Appendix A, involved autonomous navigation by the robot in an *a priori* unknown environment with dynamic obstacles, in order to reach a predetermined goal. From the intermediate goal location, the robot had to search for and locate a control panel, move toward it, and dock in front of the panel face. The CESAR demonstration was successfully accomplished using the HERMIES-IIB robot while subsets of the CEA demonstration performed using the ARES robot simulation and animation system were presented. The first session of the workshop focussed on these experimental demonstrations and on the needs and considerations for establishing "benchmarks" for testing autonomous robot control algorithms.

Another objective of the workshop was to enhance discussions between the two research teams and, in particular, promote direct interaction between specialists in respective areas of expertise. This was accomplished through panels and organized participant discussions at the end of each session and through one-to-one specialist meetings during the last afternoon of the workshop. In addition, sessions 2, 3, and 4 of the workshop focussed on presentations and discussions in the specialized topics of perception, environmental modeling, and planning and navigation to provide opportunities for in-depth technical interaction and evaluation of those essential components of autonomous mobile robot operation.

The final session of the workshop focussed on the recently initiated research activities and on the new testbeds under development at the two facilities to serve as a basis for the planning of future cooperative activities and upcoming workshops.

CESAR/CEA WORKSHOP ON AUTONOMOUS MOBILE ROBOTS

May 30 – June 1, 1989

AGENDA

Tuesday, May 30, 1989

1:00 p.m. Welcome and Introductions
C. R. Weisbin, Director of CESAR

1:15 p.m. Workshop Objectives and Organization
A. Kavenoky (CEA) and F. G. Pin (CESAR), Workshop Coordinators

Session: Experimental Demonstrations
Moderators: G. de Saussure (CESAR) and J. P. Nomine (CEA)

1:30 p.m. HERMIES-IIB Base-Case Demonstration and Discussions
D. L. Barnett et al. (CESAR)

2:15 p.m. ARES Base-Case Demonstration (Video Tapes?) and Discussions
A. Cossic et al. (CEA)

3:00 p.m. Benchmark Experiments: Robustness Considerations
M. Beckerman et al. (CESAR)

3:30 p.m. Simulation Tools for Benchmarking Robotics System: The ARES Approach
G. Dejonghe (CEA)

4:00 p.m. Participant Discussion: Toward Benchmarking Autonomous Mobile Robot Experiments
Moderators: C. W. Glover and P. F. Spelt (CESAR) and A. Cossic and G. Dejonghe (CEA)

6:30 p.m. Dinner

Wednesday, May 31, 1989

Session: Perception for Mobile Robots
Moderators: M. Beckerman (CESAR) and P. Favre (CEA)

8:30 a.m. SYMPATI 2 and a K-2D Vision System for Robotics Application
D. Juvin (CEA)

9:00 a.m. A Concurrent On-Board Vision System for a Mobile Robot
J. P. Jones (CESAR)

9:30 a.m. Presentation on 3-D Camera Research at DEMT
J. Gonnord (CEA)

10:00 a.m. Laser Range Finder for HERMIES Applications
F. J. Sweeney (CESAR)

10:30 a.m. Break

Session: Environmental Modeling
Moderators: J. P. Jones (CESAR) and D. Juvin (CEA)

10:45 a.m. Sensor Based Mapping
M. Beckerman (CESAR)

11:15 a.m. Environment Modeling for Robotic Simulation
G. Dejonghe et al. (CEA)

11:45 a.m. Lunch

Session: Planning and Navigation
Moderators: P. F. Spelt (CESAR) and A. Cossic (CEA)

1:30 p.m. Action Planning Applied to Execution of High Level Controls for Mobile Robots
D. Schmit (CEA)

2:00 p.m. Autonomous Job Planning System
L. E. Parker (CESAR)

2:30 p.m. Emulation and Simulation System for Action Planning and Navigation Algorithms
J. P. Nomine (CEA)

3:00 p.m. Navigation Algorithms for HERMIES
G. de Saussure and D. L. Barnett (CESAR)

3:30 p.m. Mobile Robot Navigation Based on a Geometrical Approach
P. Favre and J. C. Fanton (CEA)

4:00 p.m. Participant Discussion: Autonomous Robot Navigation in Unstructured Environments
Moderators: G. de Saussure and L. E. Parker (CESAR) and J. Gonnord and D. Schmit (CEA)

Thursday, June 1, 1989

Session: Recent Developments and New Test Beds
Moderators: F. G. Pin (CESAR) and A. Kavenoky (CEA)

8:30 a.m. Autonomous Learning at a Control Panel
P. F. Spelt (CESAR)

9:00 a.m. Parallelized Graph Processing Applied to Robots Vision and Navigation
J. J. Niez (CEA)

9:30 a.m. Break

9:45 a.m. Presentation of UGRA Experimental Facilities and Benchmark Proposals for Man-Machine Synergy
G. Fraize (CEA)

10:15 a.m. HERMIES-III Presentation
C. W. Glover (CESAR)

10:45 a.m. New Robotics Projects and Recent Developments at CEA
J. Gonnord (CEA)

11:15 a.m. Participant Discussion: Future Cooperative Topics and Next Years' Benchmark
Moderators: F. G. Pin and C. R. Weisbin (CESAR) and J. Gonnord, P. Darier, and G. Fraize (CEA)

12:15 p.m. Lunch

1:30 p.m. Concurrent Meetings

Specialist Meetings — CESAR and CEA Staff Members as Appropriate

Workshop Committee Meeting — A. Kavenoky, F. G. Pin and Other Principal Investigators

4:30 p.m. Adjourn

LIST OF PARTICIPANTS

D. L. Barnett (CESAR)
M. Beckerman (CESAR)
P. F. R. Belmans (CESAR)
A. Cossic (CEA)
P. Darier (CEA)
G. Dejonghe (CEA)
G. de Saussure (CESAR)
P. Favre (CEA)
G. Fraize (CEA)
R.-J. Gibert (CEA)
C. W. Glover (CESAR)
J. Gonnord (CEA)
J. S. Graham (CESAR)
V. G. Hegde (CESAR)
J. P. Jones (CESAR)
D. Juvin (CEA)
A. Kavenoky (CEA)
R. C. Mann (CESAR)
J. P. Nominé (CEA)
L. E. Parker (CESAR)
F. G. Pin (CESAR)
D. Schmit (CEA)
P. F. Spelt (CESAR)
F. J. Sweeney (CESAR)
C. R. Weisbin (CESAR)

Session I: Experimental Demonstrations

ARES Base Case Demonstration

A. Cossic, J.P. Nominé, A. Malavaud, M. Detoc
Centre d'Etudes Nucléaires de Saclay
Département des Etudes Mécaniques et Thermiques
91191 Gif sur Yvette Cedex, France

Abstract - *The ARES System is used to modelize the Hermies IIb mobile robot in its environment. The robot is modelized with its two driving wheels, two five axis manipulator arms and its control sensor platform. The environment is modelized as a flat floor room containing a control panel and several parallelepipedic obstacles. The simulator then allows to realize various experiments by providing multi-purpose basic functions: motion of the different links of the mechanical system - wheels, arms, sensor platform -, motion of obstacles, grasping of objects and sensor data acquisition. An execution of simple tasks is shown. Integration of specific navigation and planning algorithms is currently being studied.*

I. Introduction

The ARES system, being developed at the Commissariat à l'Energie Atomique - Department of Mechanical and Thermal Studies - is the result of a two-year effort on robotics simulation. The preliminary version has been written in FORTRAN 77 and is for now being coded in Ada language.

ARES is used to modelize the Hermies IIb autonomous mobile robot in its environment, in order to simulate the execution of robotic tasks. The robot model includes the definition of the different links of the mechanical structure, and sensor components. The Hermies IIb environment is modelized with its obstacles and control panel, in front of which the robot is supposed to perform monitoring and manipulation tasks.

A short account of the ARES methodology is first dealt with in this paper. This is a general overview of the ARES approach and activities upon robotics.

The following section focuses on the description and modelization of Hermies IIb in its environment.

Then, in part four of this paper, we present the basic means used for prototyping the Hermies IIb demonstration: motion of the wheels, motion of the five axis manipulator units, sensor data acquisition - for Polaroid transducers - motion of obstacles, grasping and removing a small object on the robot's path.

II. ARES system overview

ARES - Atelier Robotique Et Simulation -, standing for Robotics and Simulation Sofware Engineering Tool, was created two years ago. The original idea was to provide an open and user-oriented software environment applied to third generation robotics software development [1].

One of the fundamental aspects of ARES is certainly simulation. ARES may be seen as a simulation bed of virtual robots and mechanical systems - possibly cooperating with each other - in a virtual dynamic environment. ARES intends to provide a help for end-users in computing complex robotic tasks.

The first version has been developed for two years in FORTRAN 77, using an IBM 6150 operated by AIX operating system (UNIX System V-like) coupled with an IBM 5080 graphic workstation. This version has been used to prototype the Hermies IIb demonstration and has been of great use to achieve the final specifications of ARES.

A new version is currently being developed in Ada on Sun4 series graphic workstations. A detailed report on this work is beyond the scope of this paper and we refer the reader to another publication[2].

ARES group is looking for portability and focusing on standards. Portability involves not only the programming language but also the underlying operating system. Ada language has been choosen as the programming language for the software environment and for high-level robotics tasks, and is being studied as a language for low-level robotics software control - for it offers features such as strong typing and data abstraction, tasking, generics, and improves reliability and reusability. UNIX operating system which, apart from its qualities as a development environment, is supported by most machine manufacturers, has been chosen as the host system for the simulation environment, and possibly as a host system for cross compilers. Another choice is the PHIGS (Programmer's Hierarchical Interactive Graphics System) graphics standard which is very powerful and efficient to manipulate 3D objects.

Let us focus on - what we call - the preliminary version of ARES. The system uses a world modeling technique which is suitable for animation and 3D graphic visualization. The world model[3] that is performed is based on a constructive solid geometry tree. Rigid solids are represented by a wire-frame model - allowing reasonably fast 3D visualization and animation on the IBM 5080 workstations.

To realize various virtual experiments, ARES provides a set of basic functions: motion of the different links of the mechanical systems - either rotoïd or prismatic joints -, motion of any solid within the environment, grasping of objects by a mechanical articulated structure and sensor data acquisition for sensor data processing. Two kinds of distance measurement systems are emulated: sonars and telemeters.

Some robotics software pieces have been written and connected to this first generation simulator: a geometric/kinematic/dynamic control model, environment reconstruction and navigation algorithms. Sensor-based control algorithms are also under development.

In the next sections, we describe how the ARES system is used to modelize the Hermies IIb mobile robot and to prepare tasks for it, utilizing all the potentialities of the mobile machine: locomotion system, sensor platform, arms.

III. Modelization[4]

Foreword: modelization has been achieved with data and parameters (dimensions, sensor features...) which may be different of the actual ones. Adjusting these parameters and data can be performed easily if needed. We just wanted to illustrate a possible use of the system.

IIIa. Environment

The environment of Hermies IIb consists of a flat floor room which is approximatively 7.8 by 6.6 meters wide and 2.1m tall. Obstacles are modelized as parallelepipedic solids. A control panel is placed in the back side of the room. The control panel is a metal box with two analog meters, a row of four pushbuttons and two horizontal sliding levers. **Figure 1** lays out the experimental area for Hermies IIb. **Figure 2** shows another 3D point of view.

This is a very simple example of environment model. ARES modeller has been used in far more complex cases (parts of nuclear power plants: see **Figure 3**). The advantage of simple environments is the good response time of the system while running simulation, which is interesting to test algorithms and methods in basic cases. In complex cases, the system spends most of its time updating the graphics structures and refreshing the screen to yield animation, for the graphics processors are far too weak. Another problem is the time spent for internal geometric data management. This should improve with Ada version, first because the host machine is more powerful and still upgradable (both scalar and graphics processor), and because a good data structuration saves code and time... since FORTRAN is not suited to complex data structures management.

IIIb. Hermies IIb[5,6]

We describe here the main features of the Hermies IIb model we built for the demonstration.

Hermies IIb is an autonomous robot system equipped with two five axis manipulator arms and a control sensor platform at its head.

The robot is propelled by two independant wheels having common axle alignment. The maximum speed of the wheels, either forward or backward, is 0.6 $m.s^{-1}$.

Each manipulator is a five-degree-of-freedom unit, including the motion of the grip. The torso assembly for the arms includes a shoulder pitch motion. The two-arm assembly has a total of 13 degrees of freedom.

The sonar system consists of 25 Polaroïd range finders. See Fig. 3. Maximum range is 7.75 meters and range resolution is up to 2.5 cm. Twenty four of the sonar transducers are mounted in six 2x2 matrix clusters. Five of these clusters are mounted in a ring at the head of the robot. The sixth remaining cluster is mounted on a tiltable platform attached to the head. The effective sonar beam is around 10 degrees for each phased-array cluster.The head of the robot can turn on ±180 degrees for a 360-degree scan. The 25th sonar is located in the front side of the robot, between the manipulators.

Figure 4 shows the whole system with its arms and sensing platform. All the preceding elements have been goemetrically modelized (links, joints). Sonars are seen as small cylinders. The two CCD cameras have been added. Apart from this "morphological" external model, we explain below how simple and partial functional models of the sensors have been integrated.

IV. Demonstration

IVa. Computer architecture

The programs run on an IBM 6150 RT PC, operated by AIX operating system, and which is coupled with an IBM 5080 graphic workstation (**Figure 5**).

The graphic workstation and its devices are controlled by the programmer through PHIGS utility routines. The workstation consists of a 1024x1024 pixels color screen and four devices: a mouse tablet, a 32-key choice device, a 8-turnbutton valuator board, and an alphanumeric keyboard. All these devices can be addressed directly from application programs in order to facilitate the manipulation of 3D objects by changing the position of the

operator's eye - zoom, translation, rotation - and requesting actions to be performed - choice requests, pick requests and so on. Up to 128 colors may be visible at the same time, among a set of 4096 colors.

We only used the key panel and the turnbutton board, set in event mode to trigger graphics functions or to continuously manipulate the image on the screen.

These two devices together with their application routines constitute the user-interface of the system. The actions performed by the user control the running simulation, graphically as well as functionnally.

IVb. Computer programs

The simulation module is written mostly in FORTRAN 77, along with some C. It is an independant UNIX process, using the prealably defined CAD models, and dealing with geometrical and graphics tasks.

The simulator can be seen as a slave task accepting three kinds of requests:

- user requests (through the user-interface described above)
- CAD requests, i.e. inquires or actions dealing with the model of the world
- requests from the robot task program, replacing requests to actuators or sensors of a real robot.

The last two kinds of requests are expressed via predefined C functions. These functions can be used to program robot tasks or algorithms.

The basic primitives are a set of utility routines used to control the motion of the wheels, the motion of the arms, the grasping of an object and the various sensing systems - sonars, etc.. - for data acquisition and collision detection. The routines are described in the next sections.

IVc. Moving the robot

The wheels are controlled by commands of the form move(dq1,dq2), where dq1 and dq2 are increments to apply on each of the wheel axes at the path update rate. The robot's position in the experimental area is given by three parameters: the (x,y) location and the rotation angle θ of the robot with respect to the X axis of a frame attached to the room.

The new position/orientation of the robot is after a move command:

$$x(k+1) = x(k) + \tfrac{1}{2}(dq_1 + dq_2).\cos\Theta$$
$$y(k+1) = y(k) + \tfrac{1}{2}(dq_1 + dq_2).\sin\Theta$$
$$\Theta(k+1) = \Theta(k) + (dq_1 - dq_2)/L$$

where L is the length of the driving axle. Trajectory generation for a mobile robot with such kinematics (one non-holonomic joint) is currently under study. The approach is to use trajectories which are piecewise circle arcs, or clothoids, or splines. This basic demonstration simply uses straight line motion, and turns around the center of the axle.

IVd. Moving the arms

The position and orientation of each arm are computed by an inverse kinematics algorithm.

The (x,y,z) position of one manipulator unit is given with respect to the torso assembly.

To simplify the inverse kinematics routine, we constrained the gripper axis to be parallel to the floor at the end of the move, when positioning the tool center point. The gripper can then be turned around its axis

We did so only to be able to write rapidly a simple algorithm and to use it in the demonstration. It represents about 20 lines of C source code. Any more sophisticated algorithm could be used. The virtual robot included in the simulator obeys any set of increments of joint variables issued by the task program.

IVe. Grasping of an object

The ARES environment modeling tool is able to take account of dynamic modifications of the world such as attaching an object to another, removing

an object from the scene or adding an object.

By giving the simple name of a solid of an arborescent chain, we can attach it to another solid, for instance the gripper of one of the manipulator units.

By the same way, the robot is able to leave the solid attached to it in another location.

Figure 6 illustrate the grasping of a small cube by Hermies IIb which is then turning on itself and finally leaving the small object outside its way.

IVf. Moving the objects

As mentionned above, objects can have their location changed within the environment.

It is thus easy to simulate the coming out of an human-like object in the scene and its movement. Then we can decide that the object's way will cross the robot's path in order to constrain the robot to stop and wait till the object goes away, thanks to an ultrasonic sensor for instance (see further).

IVg. Sensor data acquisition

A simple model of the sonars is used: the effective sonar beam is modelized as a cone. The sonars detect obstacles present within this cone, between a minimum and a maximum range; the returned information corresponds to the distance of the nearest obstacle within the scope of the sensor (**Figure 7**).

In the simulator the sonar beam is actually discretized into several rays. A ray-tracing technique is used to obtain the impact points on the objects and the point corresponding to the minimum distance ray of a sonar is kept into memory. The only information returned to the robot application program is a distance for each sensor.

This model is quite simple but can be easily improved, taking into account more physical properties of the measuring principle. The basis remains a discretization of the field of the sensor, then a geometrical and/or physical processing baser upon raytracing, then returning a propagation time to the device, which is proportional to the shortest distance.

Note that the accuracy of such sensors is good as far as the distance is concerned, but since the angular resolution is poor, the sensor only acts as a good hit-or-miss indicator. Further use of such sensors for environment reconstruction is possible, but requires sophisticated storage and processing of successively scanned data.

Cameras can be virtually activated, but only to evaluate their visibility scope, and to mark objects visible within this scope. No data such as an image is returned. Raytracing is however a good methodology and would allow to simulate image acquisition if needed, based upon models of the cameras and of the light.

IVh. Execution of simple tasks

At this stage, we have described the main basic utility functions for the simulator. These functions can be used separately or alltogether for testing the execution of robot tasks.

Basically, the system is designed to allow programming of algorithms and tasks in C language. One possibility is to issue commands for the robot from an AI environment. We started the coupling of the robotics functions with SPIRAL, which is an AI environment developed by CEA/DEMT [7]. SPIRAL is organized around a PROLOG-like shell, and is also written in C language.

The simulator is run as a background process and opens a message system. Any other process can communicate with the simulation process, as long as it is able to enter the message system, and to correctly call the communication functions.

Two examples of simple tasks have been chosen. The first one consists in moving the robot close to a small obstacle; Hermies IIb must stop near the obstacle and take it off to free its path. The position of the object is given to the robot.

The second example is the following: a moving object is introduced in the

robot's experimental area. The frontal sonar must detect the unexpected obstacle and the robot has to stop till the object is outside its immediate vicinity (**Figure 8**).

These two samples tasks were directly programmed in C language.

V. Ongoing studies

Implementation of specific obstacle avoidance, navigation and planning algorithms is currently being studied.

We intend to take advantage of the Spiral expert system shell for action planning and high-level decision making.

ARES range of basic research and development includes studies on 3D world modeling, simulation, robotics tasks and mission analysis, IA and software engineering. We have proposed to use the preliminary version of ARES to modelize the Hermies IIb robot in its environment and to show how simulation may be helpful to realize various virtual experiments. The present version of ARES is used to develop small applications and will be replaced at the end of the year by a new one developped in Ada language. The new version will allow much more sophisticated modelization as well as task programming.

VI. Conclusion

The successfully implemented demonstration validates some concepts of ARES.

In particular, the 3D world modeling technique which is necessary for simulation, graphic vizualization and dynamic animation.

The concept of emulation/simulation and off-line programming will be strenghtened in the next version of ARES and a generalized 3D world modeling will be avalaible.

REFERENCES

1. G. DEJONGHE, D. CHAIGNE and A. COSSIC, "ARES: Projet de développement", Internal Report, Commissariat à l'Energie Atomique (1988).

2. G. DEJONGHE, "Simulation Tools for Benchmarking Robotics Systems: the ARES approach", ORNL Workshop on Autonomous Mobile Robots, Oak Ridge, Tennessee, 1989.

3. G. DEJONGHE, A. COSSIC and D. CHAIGNE, "Environment Modeling for Robotics Simulation", ORNL Workshop on Autonomous Mobile Robots, Oak Ridge, Tennessee, 1989.

4. D. CHAIGNE, "Un prototype de modèle géométrique d'environnement 3D pour le simulateur graphique d'ARES", Internal Report, Commissariat à l'Energie Atomique (1987).

5. B.L. BURKS and al.,"A Demonstration of Autonomous Navigation and Machine Vision using the Hermies IIb Robot", Southeastern Regional Media Leadership Council, Conference on Microcomputers and Artificial Intelligence in Communication Technologies, UT at Chattanooga, October 29-31, 1987.

6. B.L. BURKS and al., "Autonomous Navigation, Exploration, and Recognition Using the Hermies IIb Robot", IEEE expert (1987).

7. Y. SOUCHET, "Manuel utilisateur Spiral", Internal Report, Commisariat à l'Energie Atomique (1988).

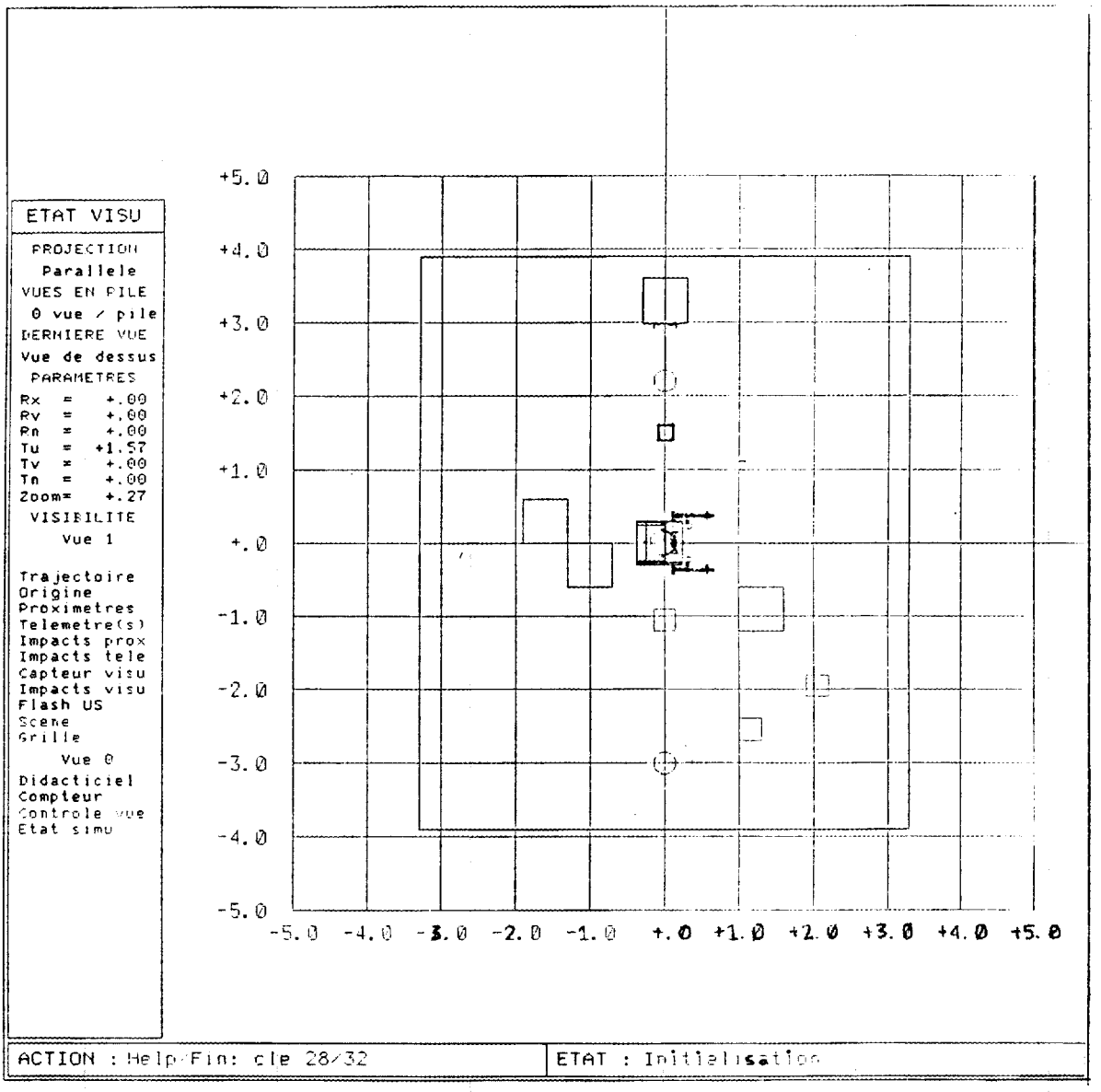


Figure 1 : ARES model of the experimental area for Hermies IIb

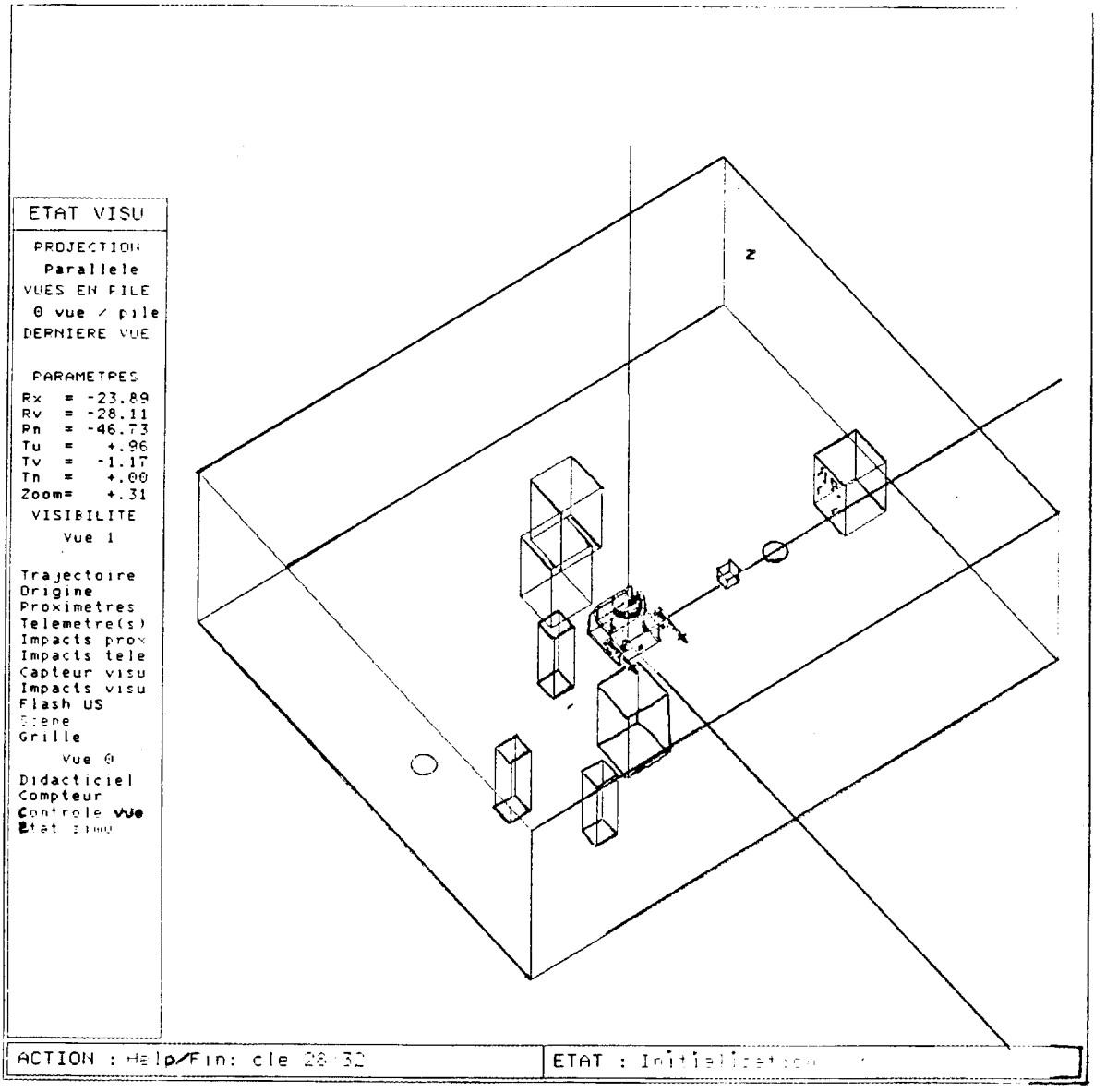


Figure 2 : Another point of view

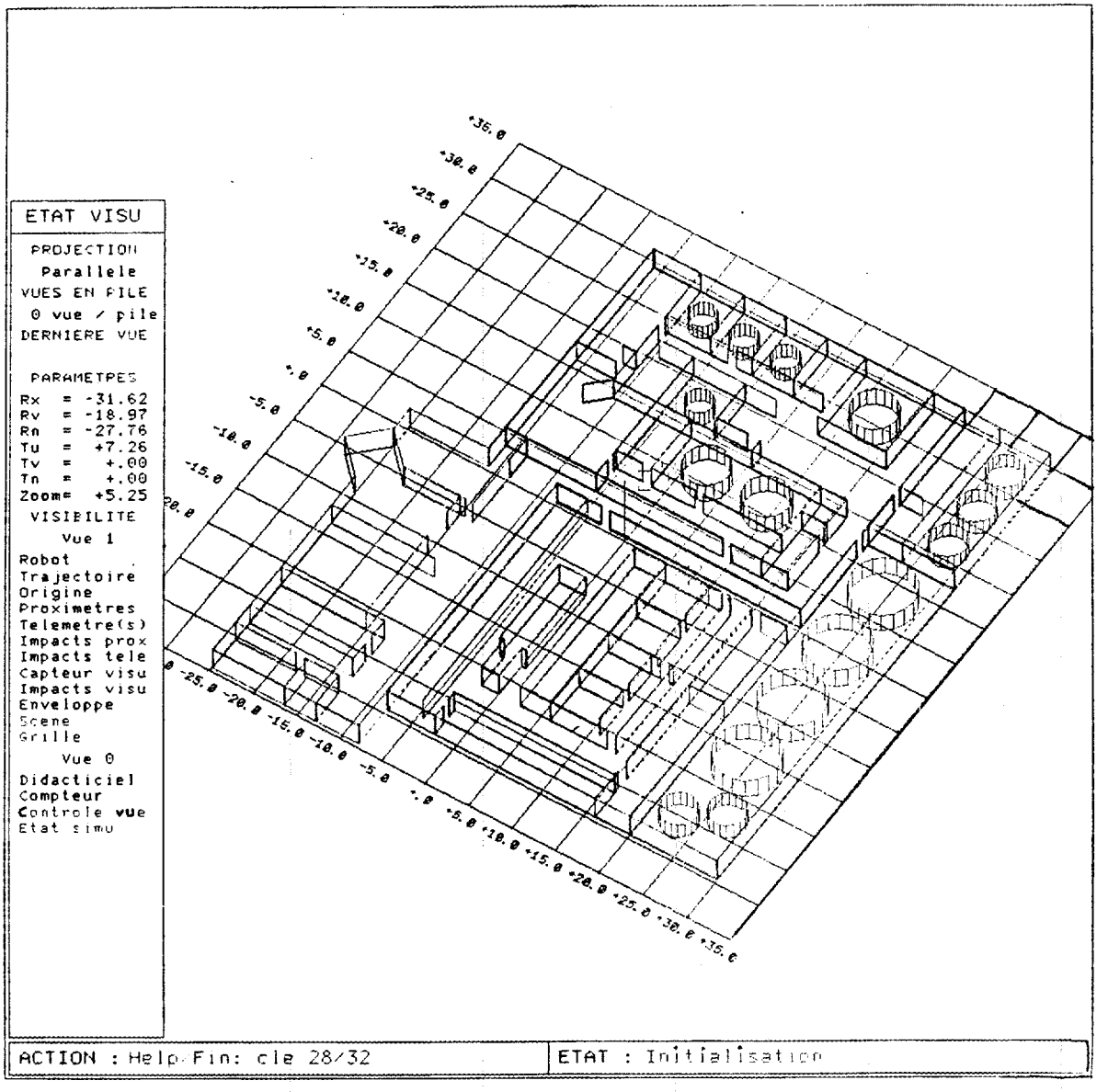


Figure 3 : Simplified CAD model of a part of nuclear power plant

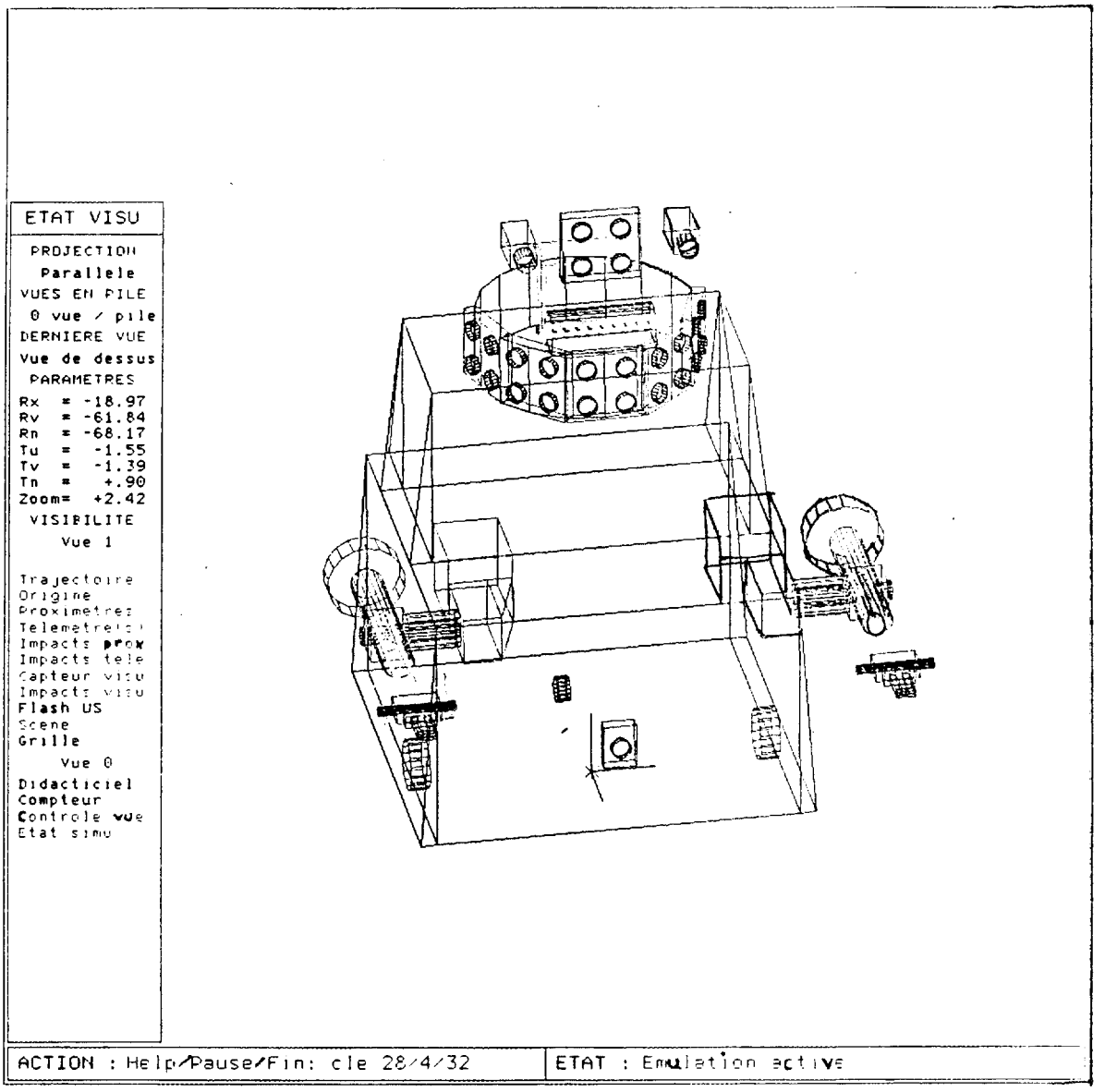


Figure 4 : Example of ARES model for HERMIES IIb

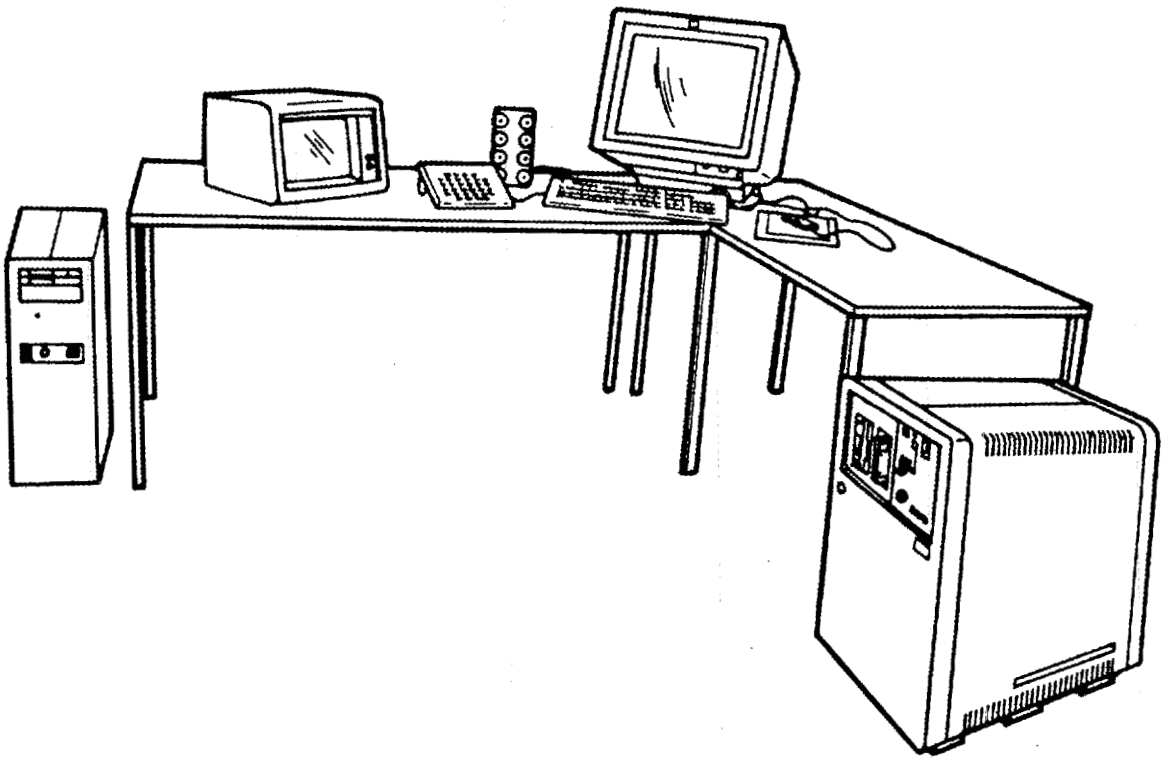


Figure 5 : Layout of the ARES workstation

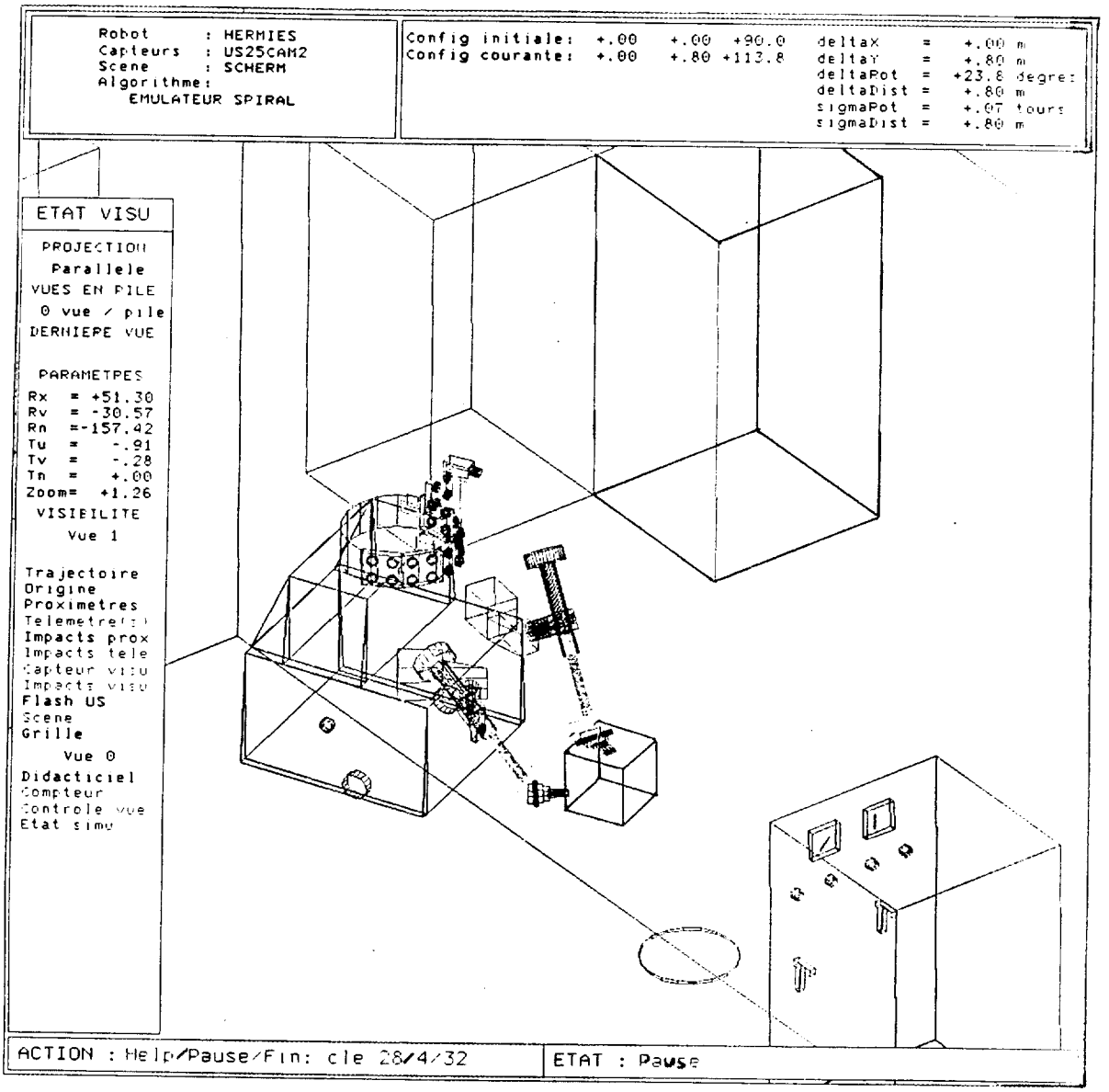


Figure 6 : Grasping of a cube by Hermies IIb

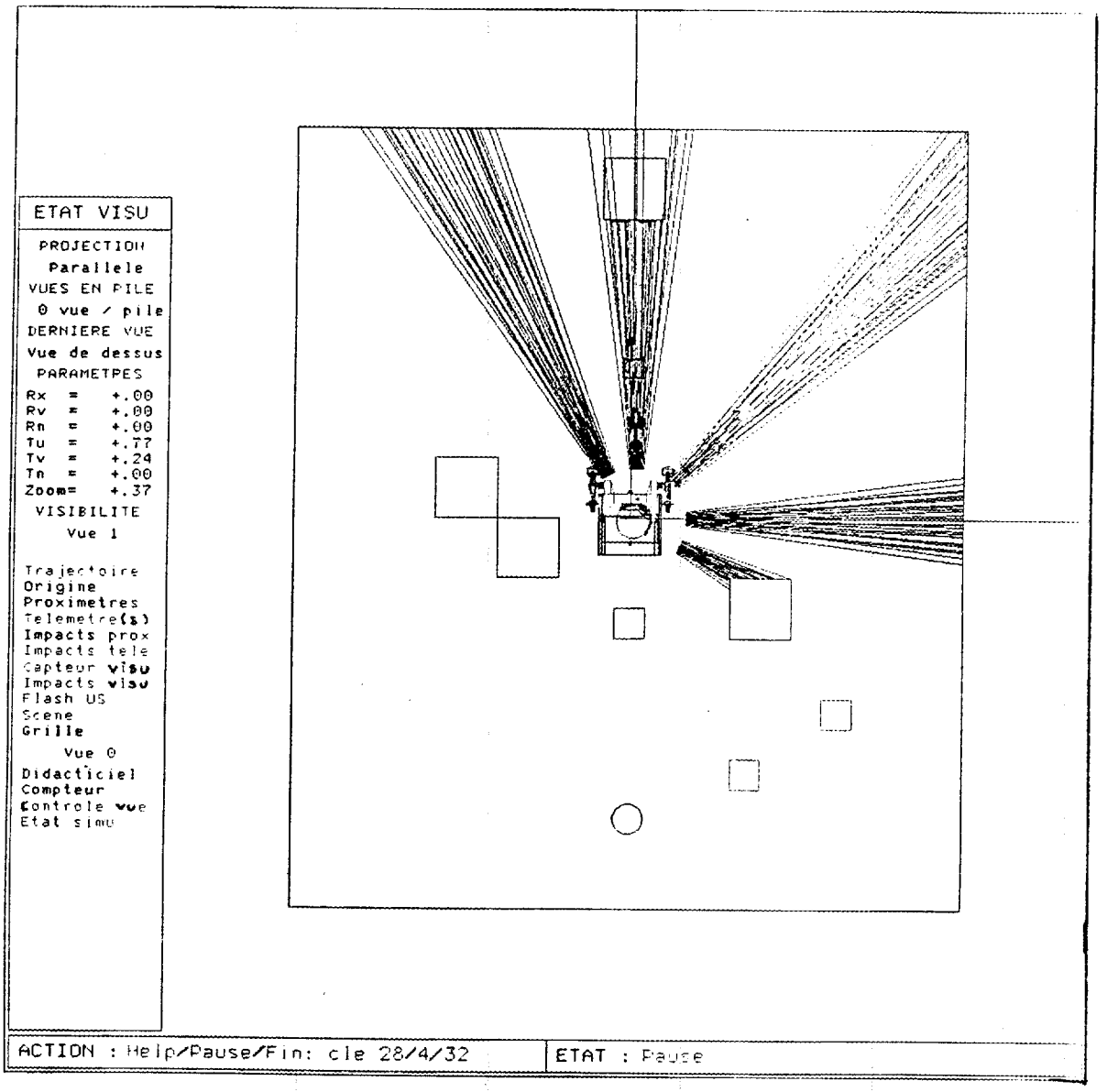


Figure 7 : Simulation of sonars

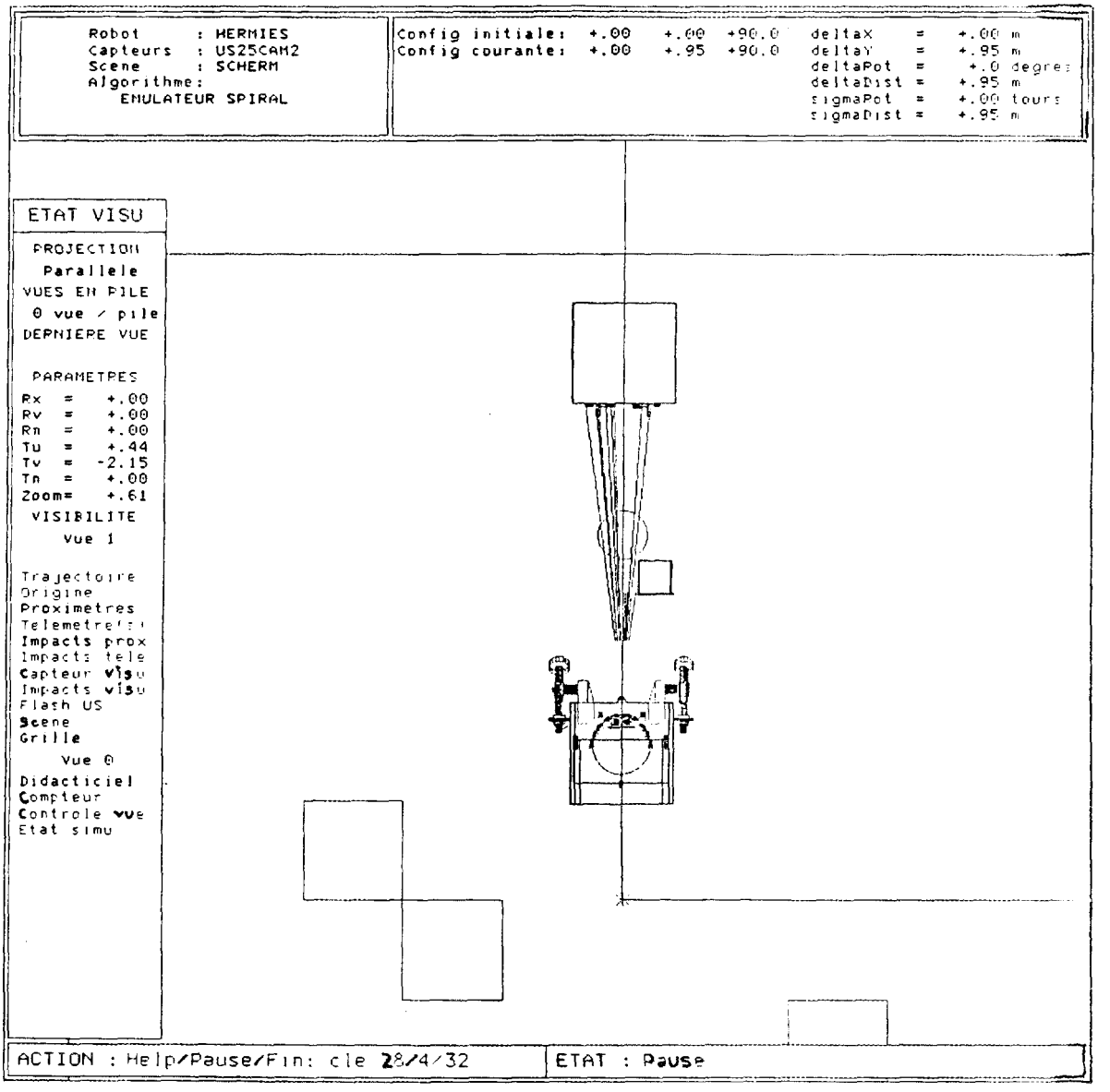


Figure 8 : Hermies IIb stopping in front of a moving obstacle

ARES BASE CASE DEMONSTRATION

- ARES ACTIVITIES OVERVIEW
- MODELIZATION
- DEMONSTRATION

OVERVIEW

- HISTORY

- ARES FEATURES

Open and user-oriented software environment

Simulation

Virtual robots in a virtual environment
Computation of complex robotic tasks

First version (1986-1988)

FORTRAN 77, C
PHIGS
IBM 6150 RT PC (AIX)
IBM 5080 graphic workstation

Second generation (1989->)

Ada coding
PHIGS
Sun4 series graphic workstations

MODELIZATION

- WORLD MODEL
  - CSG tree
  - Wire-frame representation
- ENVIRONMENT OF HERMIES IIb
  - Room
  - Obstacles
  - Control panel
- HERMIES IIb
  - Wheels
  - Manipulator units
  - Sonar system
  - Sensing platform

DEMONSTRATION

- COMPUTER PROGRAMS

  Simulation process

  Requests accepted

  User requests
  CAD requests
  Actuator and sensor requests

  Basic primitives

  Motion of the wheels
  Motion of the arms
  Grasping of an object
  Motion of an obstacle
  Sensor data acquisition

  Execution of simple tasks

DEMONSTRATION

- **TASK 1**

  Positioning its arms

  360-degree scan

  Moving forward

  Removing a small obstacle

  Arms configuration computation
  Taking off the object
  Turning on itself
  Leaving the object
  Turning back

  Reaching the goal

DEMONSTRATION

- TASK 2

  Moving forward

  A human-like obstacle is going to cross the robot's path

  Detecting the moving obstacle

  Waiting till the obstacle is out of the robot's vicinity

  Reaching the goal

CESAR-89/17

ROBUST PERFORMANCE OF MULTIPLE TASKS BY AN AUTONOMOUS ROBOT*

Martin Beckerman, Deanna L. Barnett, J. Ralph Einstein,
Judson P. Jones, Philip F. Spelt and Charles R. Weisbin

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

"The submitted manuscript has been authored by a contractor of the U.S. Government under contract DE-AC05-84OR21400. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes."

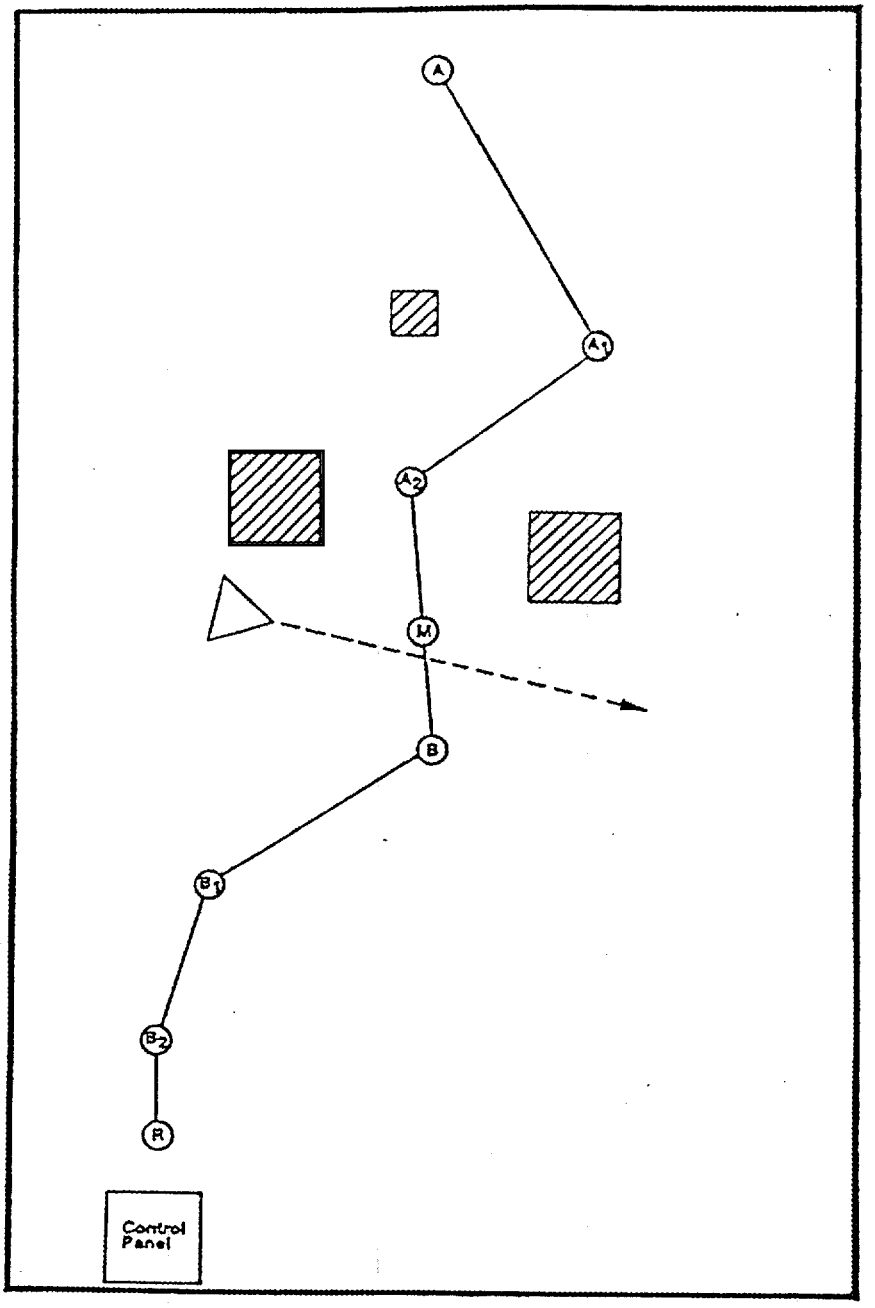
Invited Presentation: Workshop on Integration of AI and Robotic Systems,
Scottsdale, Arizona, May 19, 1989.

* Research sponsored by the Engineering Research Program at 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.

Robust Performance of Multiple Tasks by a Mobile Robot

- Integrated experiment:
  (i) Sonar-guided navigation
  (ii) Vision-guided navigation
  (iii) Vision-guided manipulation

- Tasks in the experiment include:
  (i) Sensor-based world modelling
  (ii) Path planning
  (iii) Navigation
  (iv) Manipulation
  (v) Machine learning


A
A1
A2
M
B
B1
B2
R
Control Panel

Sonar-Guided Navigation
Ultrasonic Sensing

- 120 samples of range data
  Spanning 360 degrees
  Distance to the nearest object
  Units of 0.1 ft.

- Each sonar unit produces a 50 kHz burst
  1 msec in duration
  Effective beam width is about 18 degrees

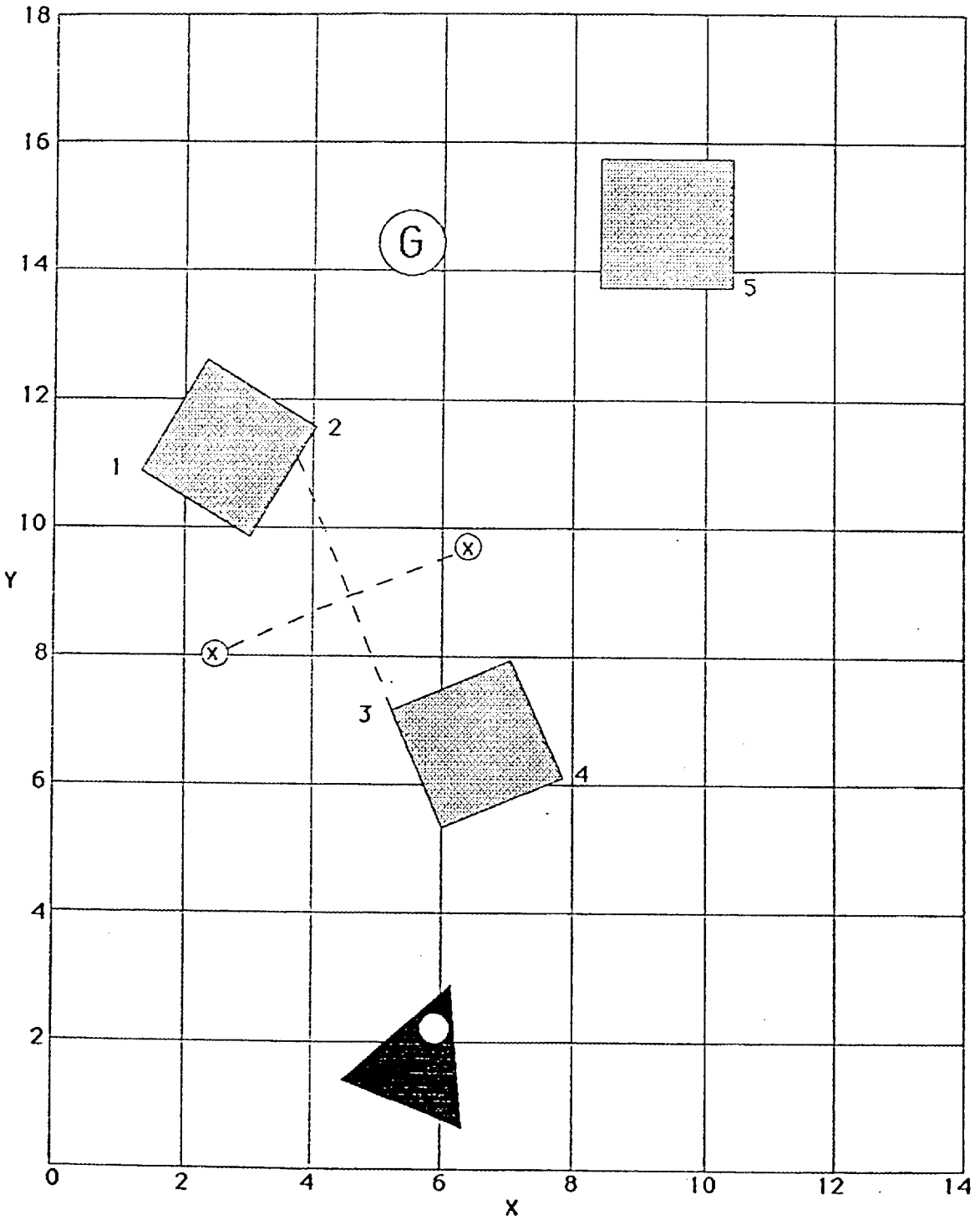
- Systematic errors in processing:
  (i) Distortions
  (ii) Specular relfections

Sonar-Guided Navigation
Navigation

- Sole scan processing
  (i) Find depth discontinuities
  (ii) Make list of edges
  (iii) Then create list of corridors

- Path planning
  (i) Prune list of excessively narrow corridors
  (ii) Using a depth-dependent minimum width to partially take into account distortions
  (iii) If not clear to the goal, plan a path to an intermediate destination

- Iterate

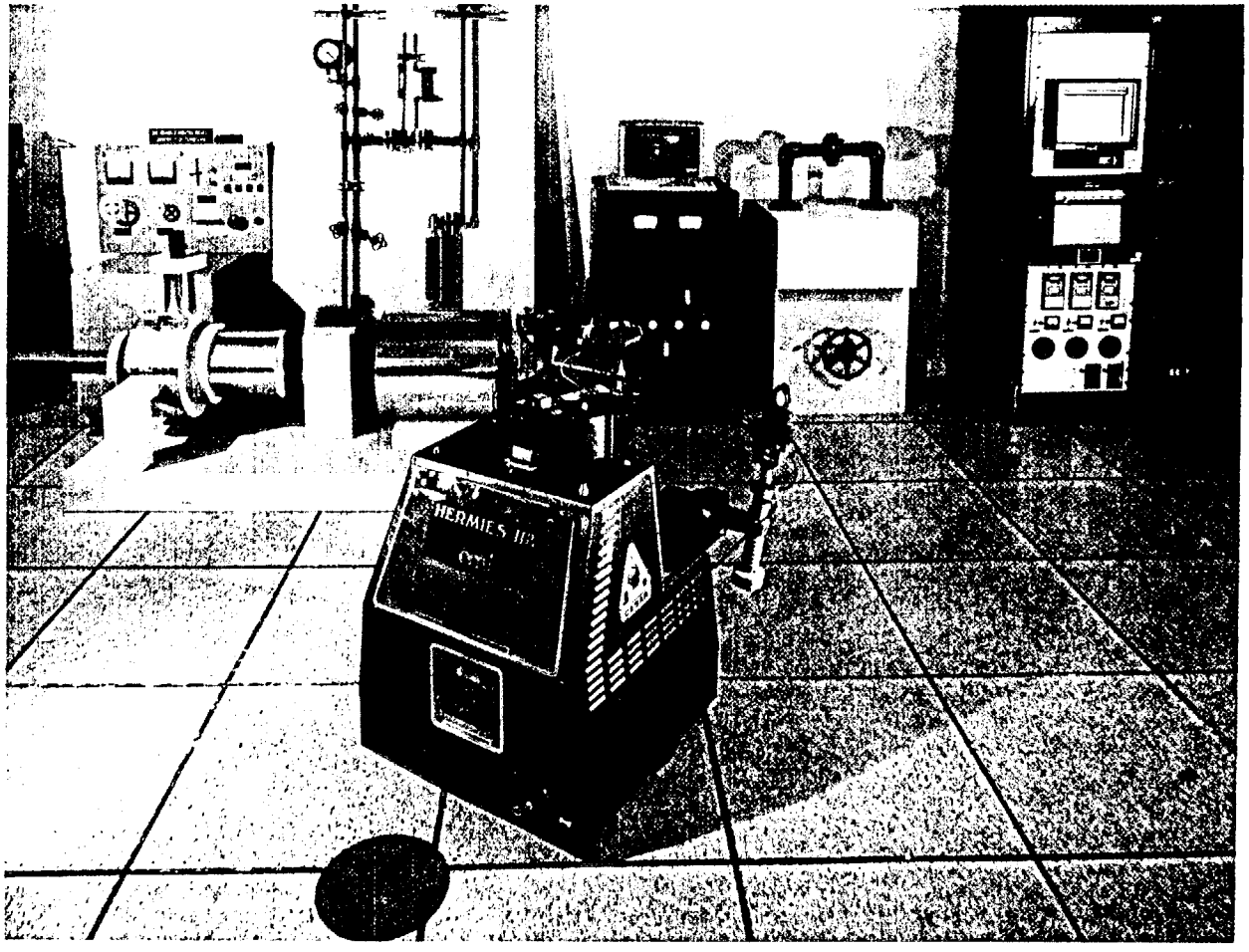

18
16
14
12
10
8
6
4
2
0
2
4
6
8
10
12
14
Y
X
G
1
2
3
4
5

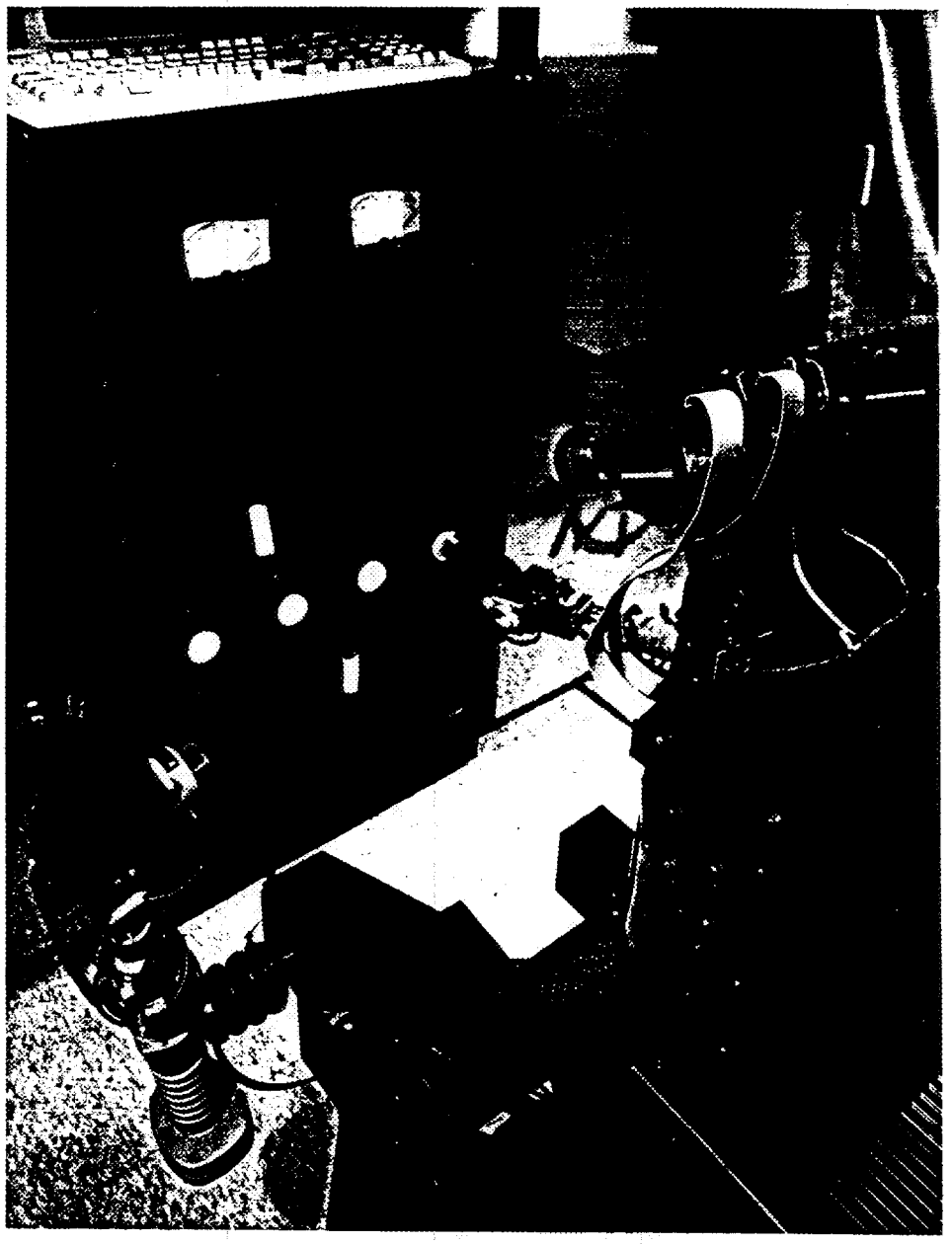
Sonar-Guided Navigation
Navigation

- Clips Expert System:
  Checks - clear to goal/goal is reached
  Requests data
  Requests paths

- Reliability:
  (i) Minimum corridor width is 5 ft.
  (ii) Dependence on object surface properties

- Fails when:
  (i) Clear paths appear blocked
  (ii) The robot becomes trapped in a loop


HERMIES III



Vision-Guided Navigation

- Required accuracy in docking is 2" x 3"
- Dimensions of the meters known
  Optics of the CCD camera known
- Procedure:
  (i) Identify the control panel
  (ii) Use dimensions of meters in image
  Calculate distance and angle
  (iii) At large distances (15') 20% accuracy
  At small distances (3') 1% accuracy
  (iv) Choose intermediate destination
- Iterate

Vision-Guided Navigation and Docking at the Control Panel

- Done in 3 to 5 steps
- Docked successfully within 1" of center line, 19" to 22" from front surface
- However, to do so:
  (i) Robot orientation at start of experiment must be known to 1 degree accuracy
  (ii) Sensor turret must be calibrated to 1 degree accuracy
  (iii) Location of the optic axis in the CCD array must be known to single pixel accuracy

Identifying the Control Panel
Visual Data Processing

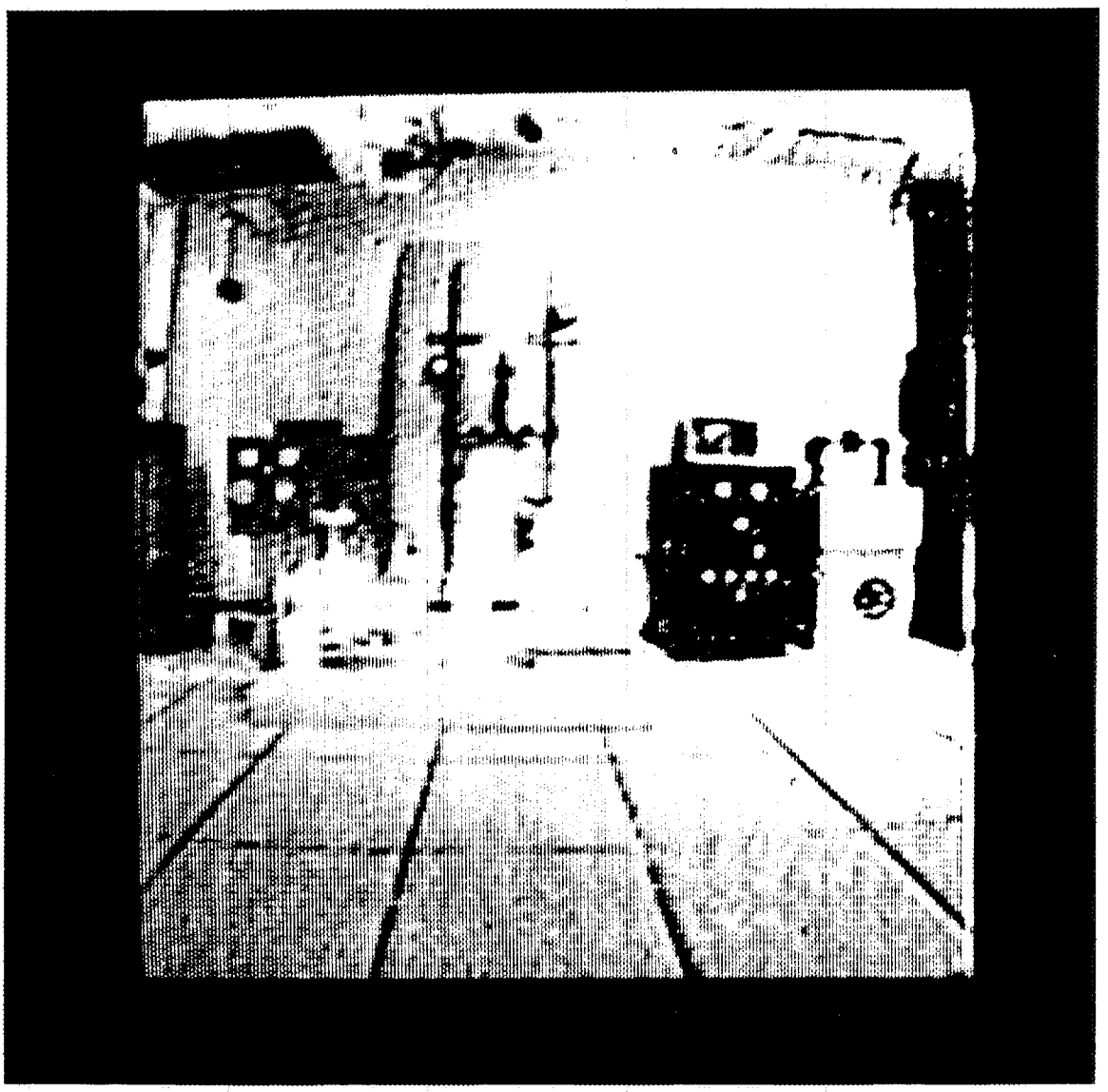
- Grey-scale morphology
- Binary morphology
- make list of all dark regions
  make list of all light regions
- for each entry in the list
  calculate geometric properties:
  area (cardinality), (x,y) centers-of-gravity
- prune list by imposing geometric conditions
  until only the control panel and its two
  meters remain

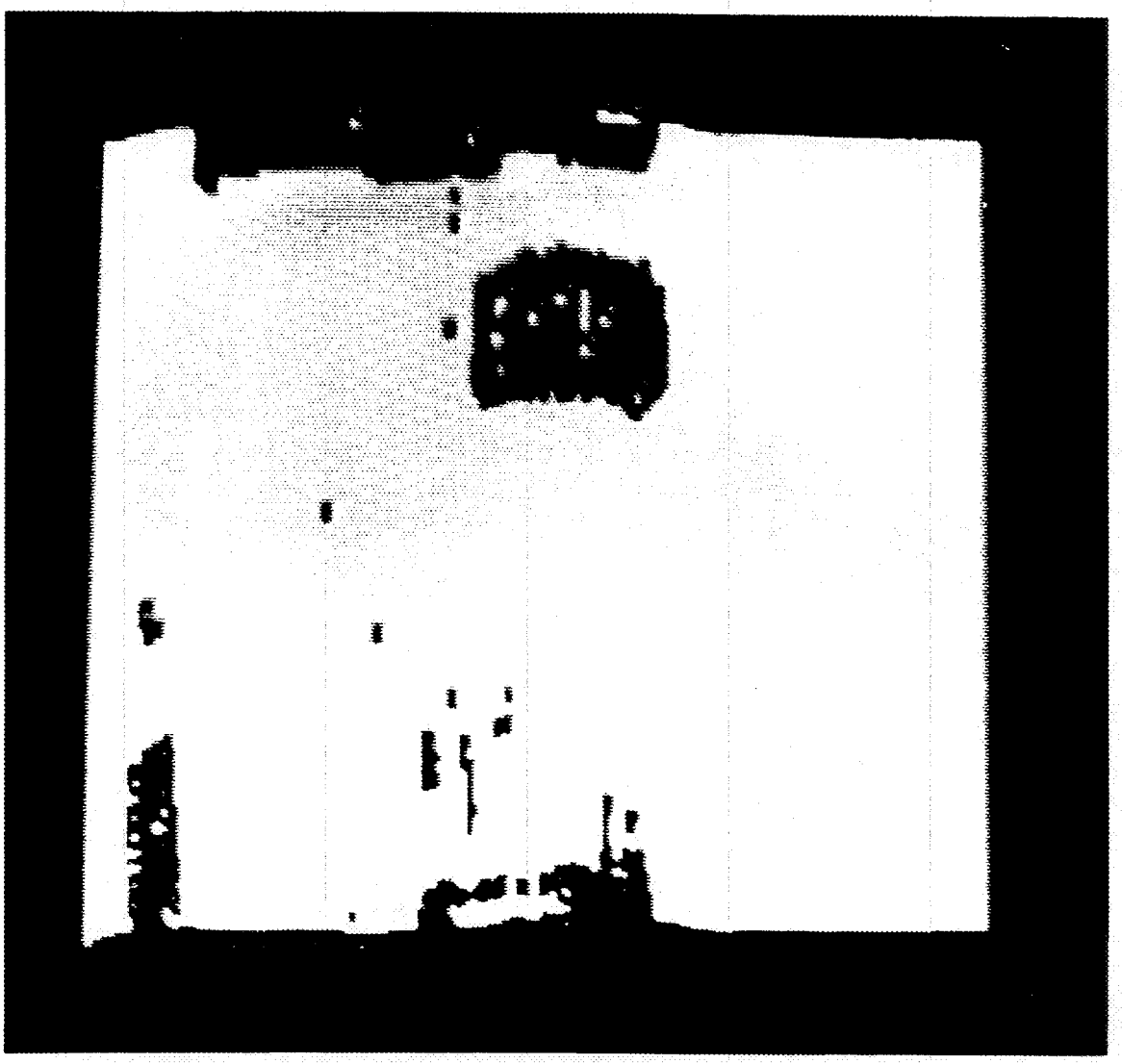
Vision-Guided Navigation and Identifying the Control Panel

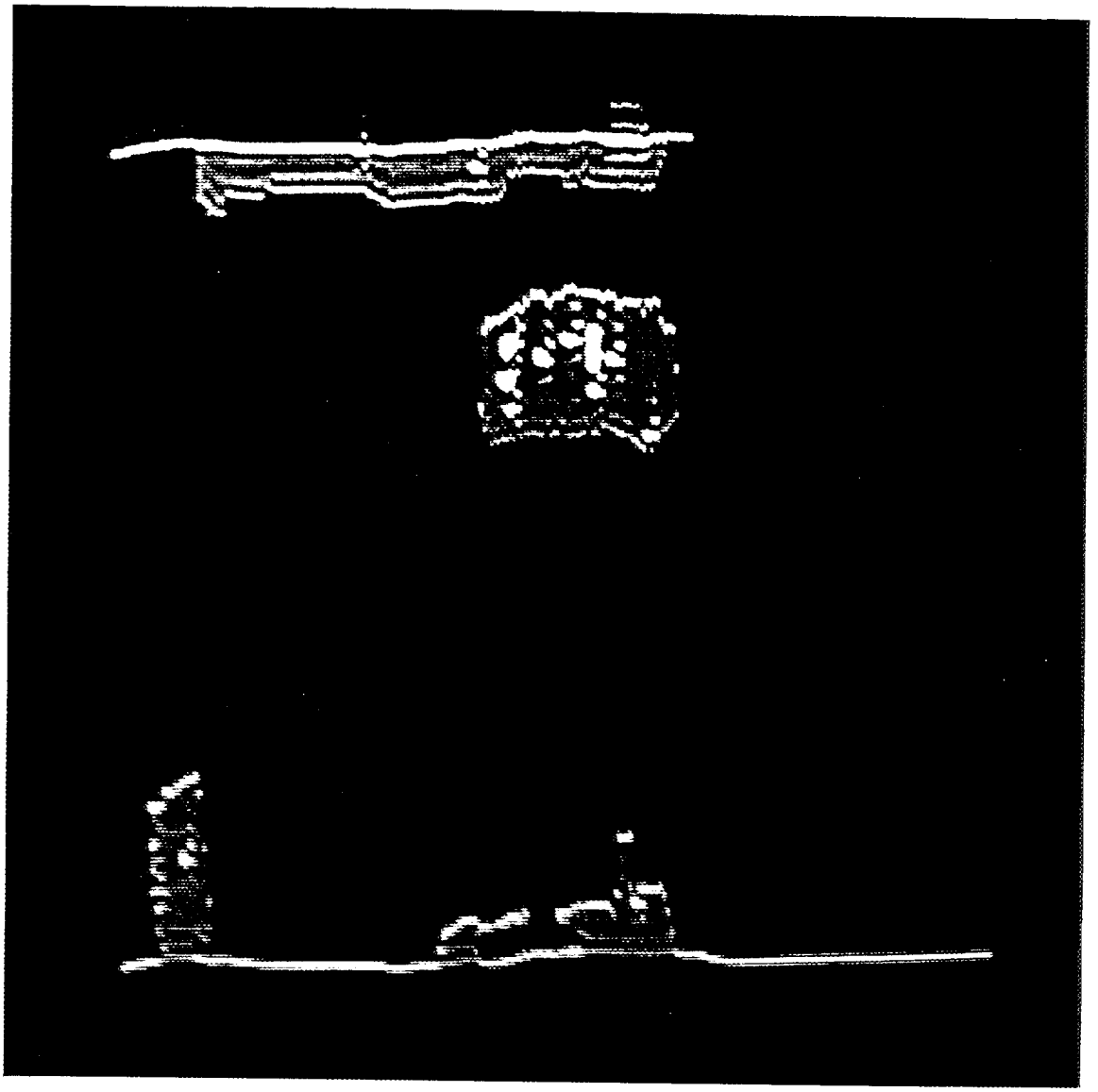
- Two meters (white regions) side-by-side
  Inside
  A control panel (dark region)

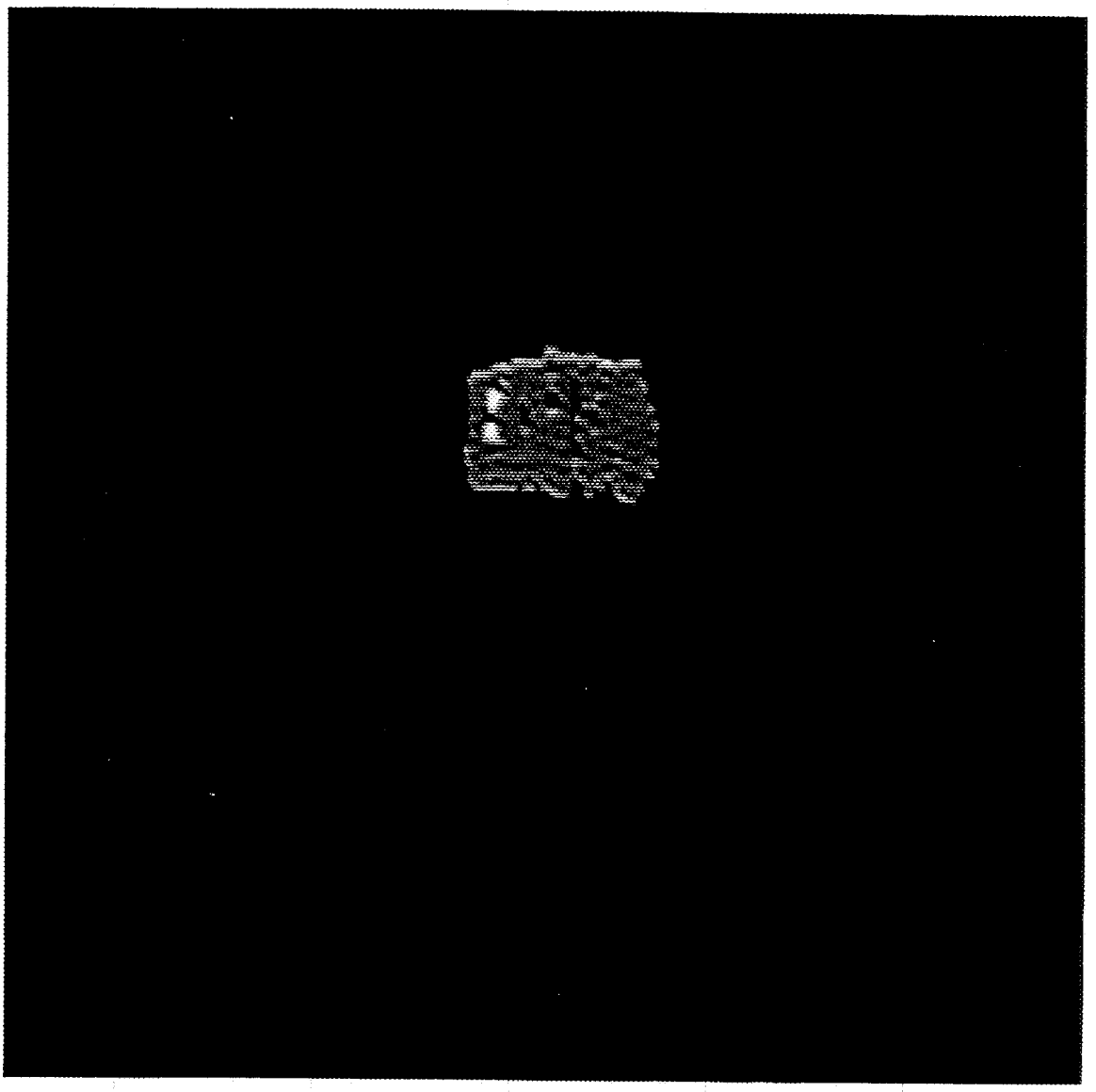
- Successful from 15' to 18"

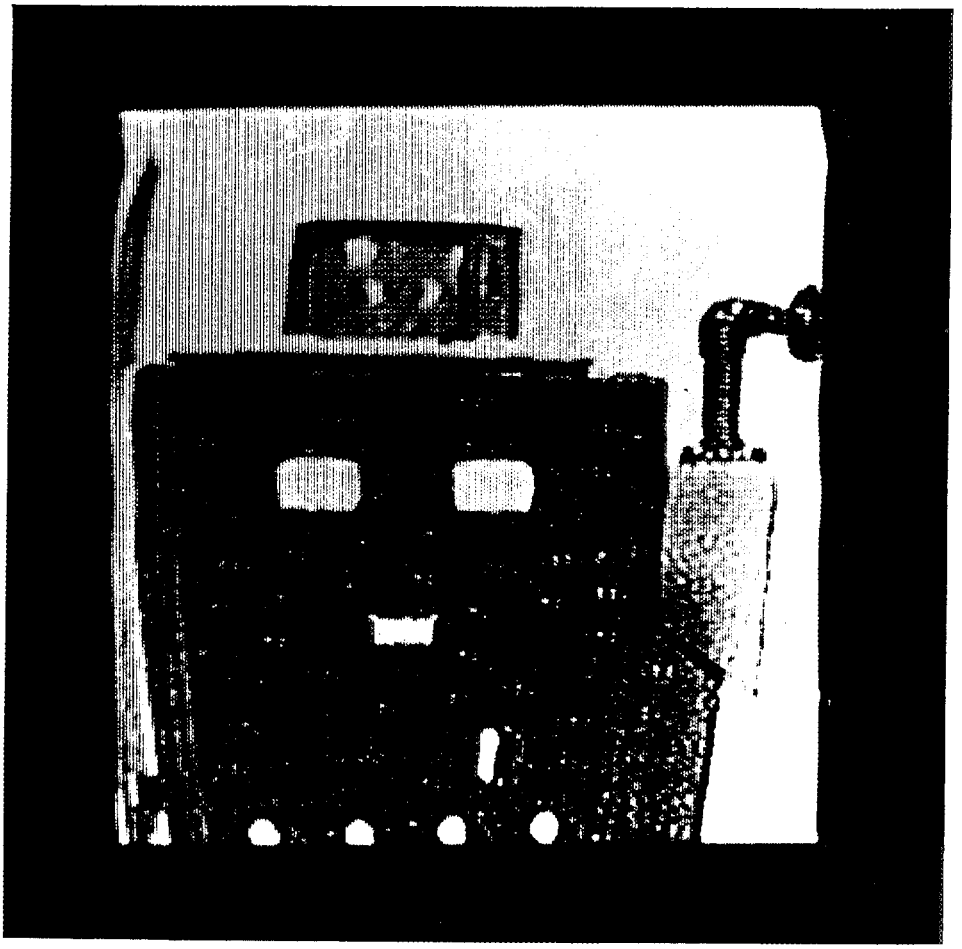
- However, to do so:
  (i) Must be within the field-of-view
  (ii) No other objects satisfying the
  meter-panel criteria in view
  (iii) Sensitive to lighting (glare/shadows)

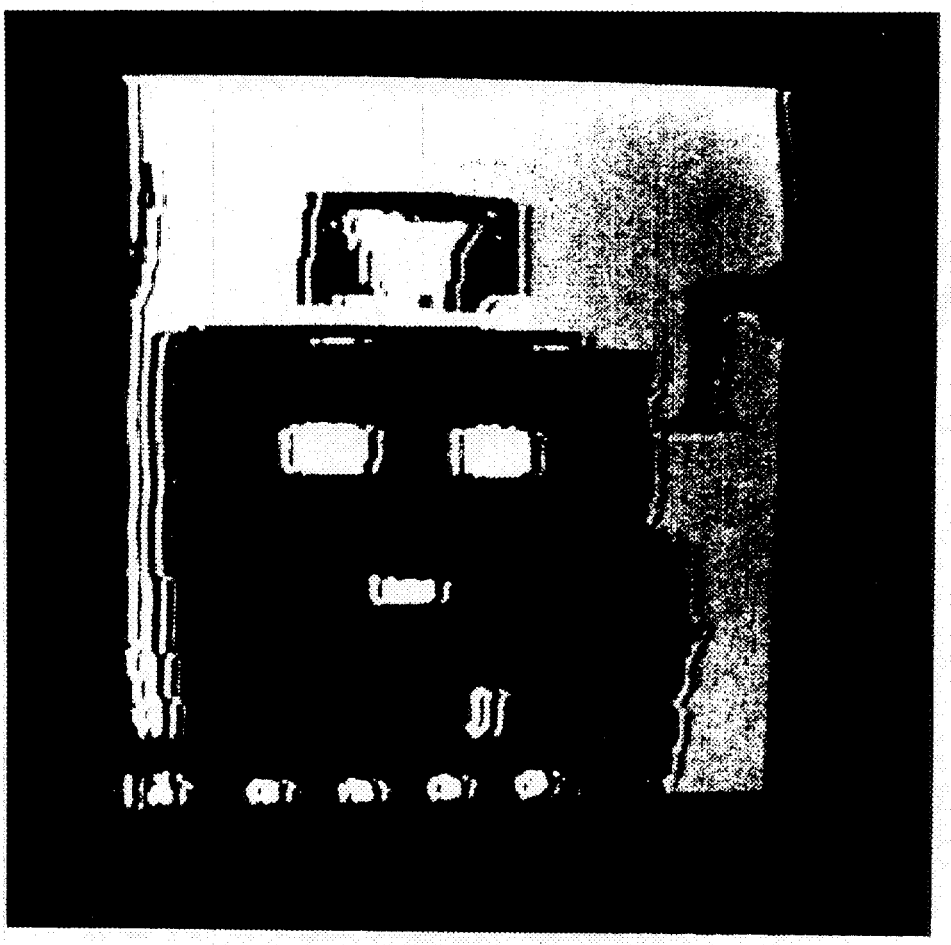


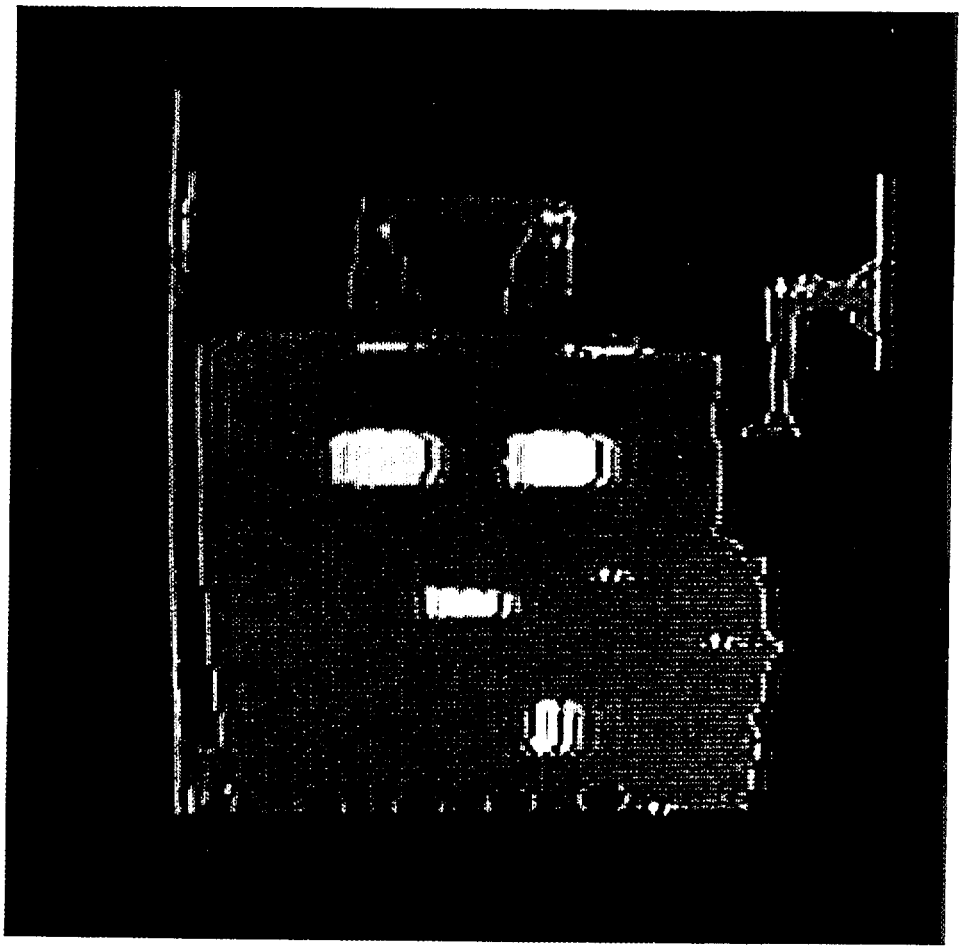


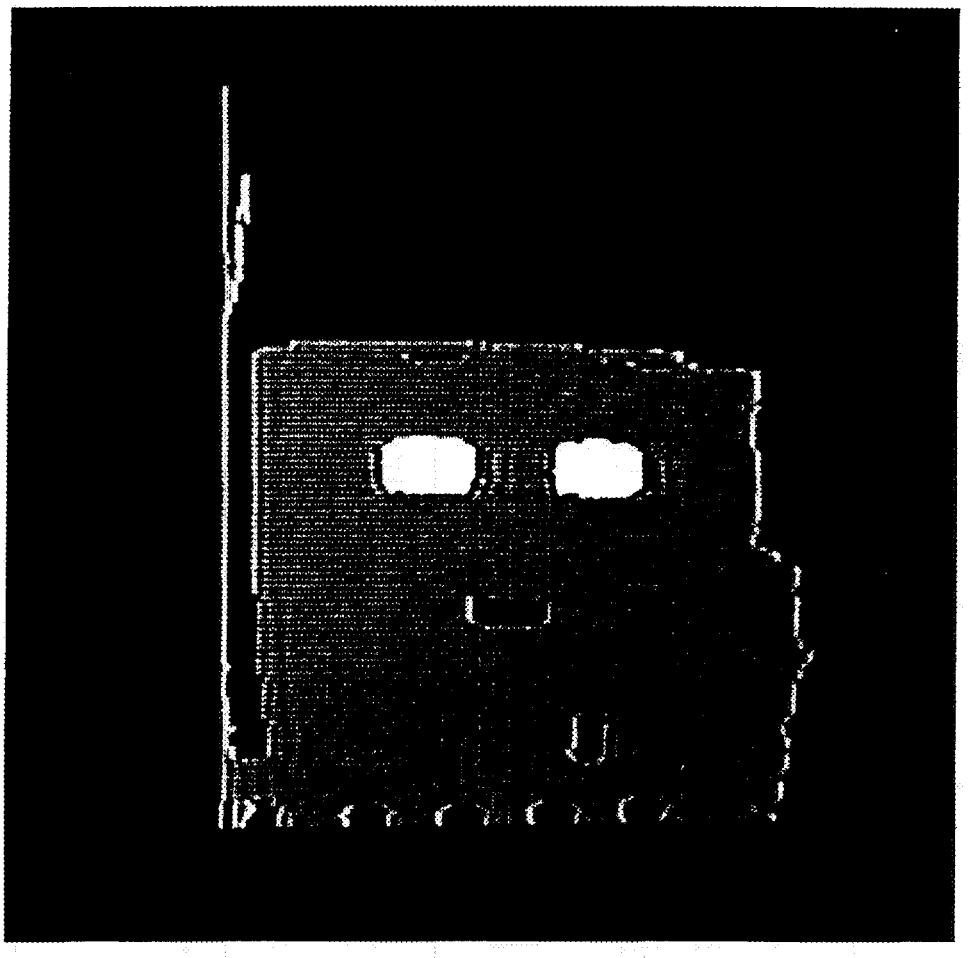












Manipulating a Control Panel Using Visual Feedback

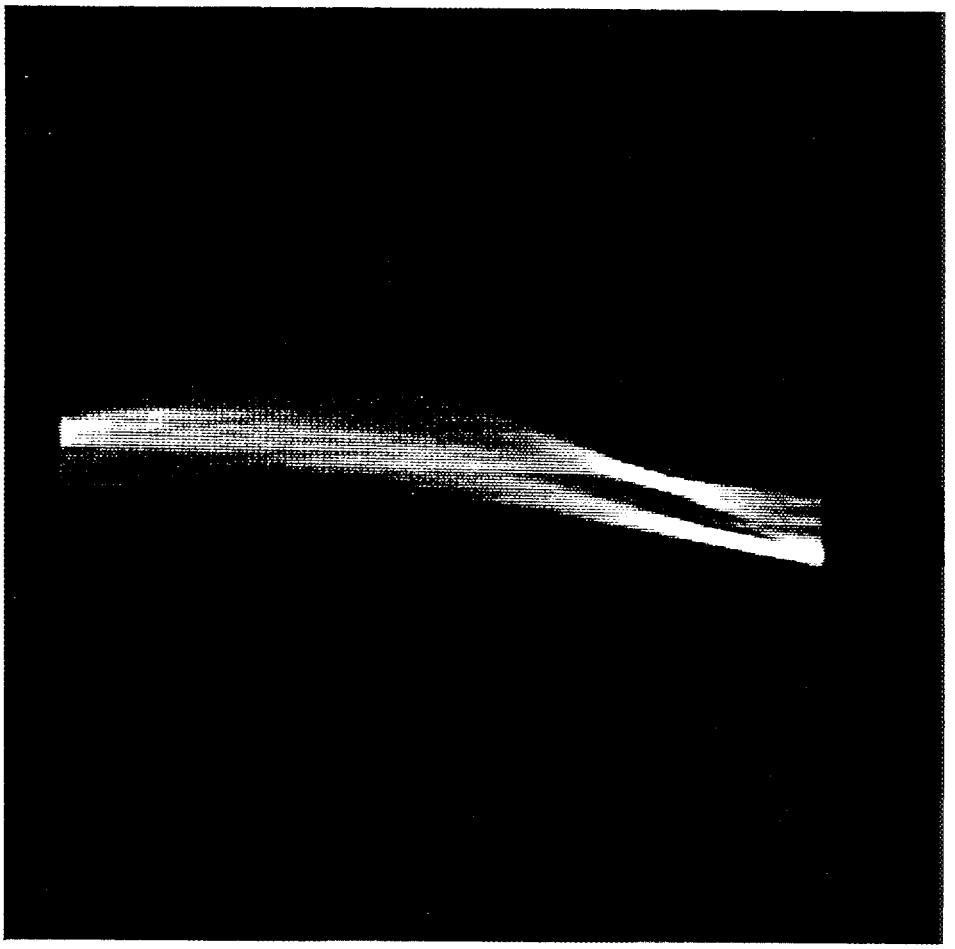
- Main problems:
  (i) No encoders
  (ii) Considerable backlash
- Used pair of CCD cameras; stereo
- LEDs placed in tip of end-effectors
- Procedure:
  (i) Determine location of end-effector
  (ii) Choose destination of end-effector
  (iii) Find path from one to the other
- Iterate

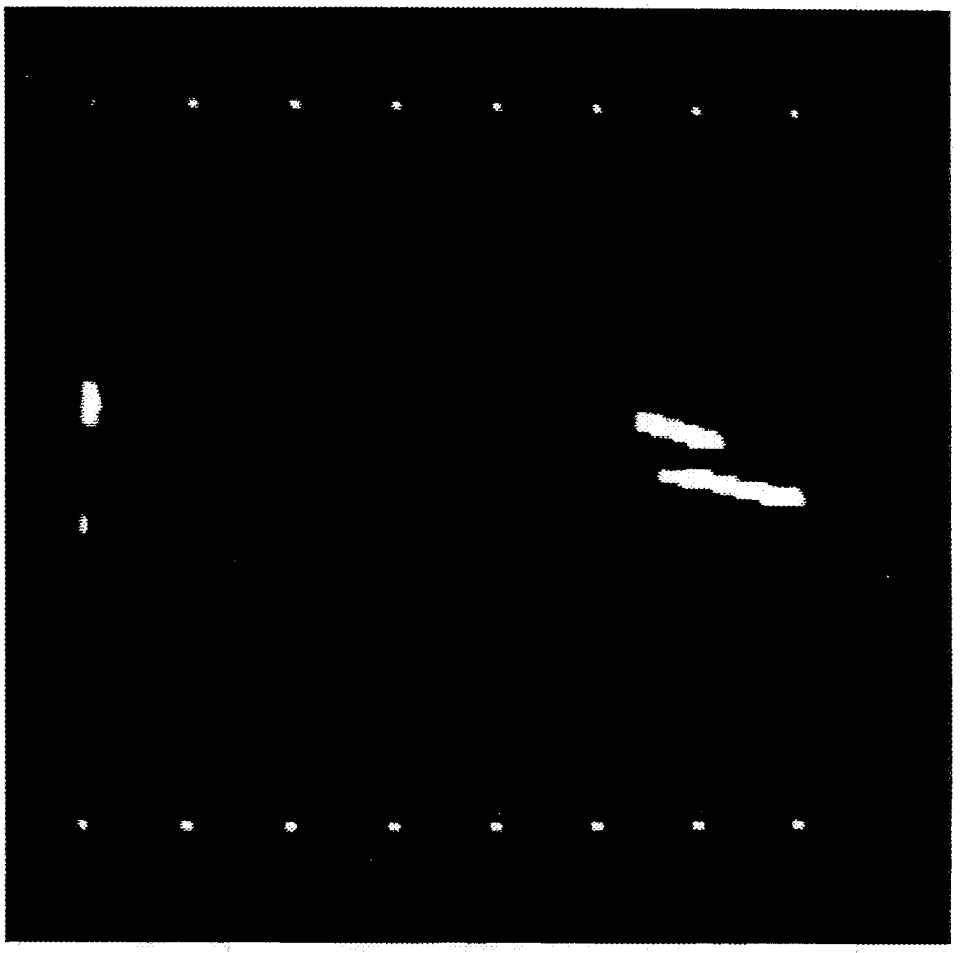
Manipulating a Control Panel Using Visual Feedback

Task	Accuracy Required
Push buttons	2 cm radius
	1.5 cm var in depth
Move slides	3 cm var in pos
	5 cm var in height
	5 cm var in depth

- All accuracy requirements met

- However, in doing so:
  (i) Characteristics of the control panel known
  (ii) Designed to be consistent with the manipulation/sensing requirements of robot





Hough Transform and Reading a Meter

- Determine orientation of a needle
- Use shape parametrization: $\rho = x \cdot \cos\theta + y \cdot \sin\theta$
- Procedure:
  (i) Make binary image from grey-scale subimage
  (ii) Parametrize and increment Hough accumulators
  (iii) Select local maxima in Hough space

Hough Transform and Reading a Meter

- Achieved 2% (full-scale) accuracy
- However, in doing so:
  (i) Background details produce additional maxima
  (ii) Used knowledge of meter geometry
  to define subimage
  (iii) Used lookup table to relate needle angle
  to absolute meter value

Robust Performance of Multiple Tasks by a Mobile Robot

- Integrated approach:
  (i) Build an entire system
  (ii) Determine the bottleneck issues
  (iii) Work on them

- Work in progress on errors:
  (i) Systematic error reduction
  in multiple scan sonar data processing
  (ii) Multisensor integration
  (iii) Error recovery

- Observation: AI methodologies must take into account sensor and mechanical errors, uncertainties and limitations

Simulation Tools for Benchmarking Robotics Systems: the ARES approach

G. Dejonghe
Centre d'Etudes Nucléaires de Saclay
Département des Etudes Mécaniques et Thermiques
91191 Gif sur Yvette Cedex, France

Abstract - *An ideal software environment for robotics task benchmarking must first provide two clearly separated components: a simulator which intends to modelize the 'real world' physics and behavior, and an emulator whose purpose is to replace the robot regarded as a controlled mechanical system. These two entities constitue a platform to develop and test task programs. The emulator plays the part of an interface between task programs and the simulator, by issuing appropriate requests to the simulator - for sensoring, motion, tooling...-. The simulator returns physical data expressed as sensor data by the emulator toward the task program. A benchmark experiment may be seen as (1) describing a robotic task through the emulator, (2) describing the initial context - physics of the robot and of its environment -, (3) submitting the task program to the emulator/simulator, and (4) observing, collecting, and analyzing results of the simulation. ARES first provides a simulator dedicated to robotics. Then, a software environment supports the development of required emulators, and eventually provides a library of generic robotics packages and AI tools dedicated to task programming.*

I. Introduction

A robotic system can be seen as a controlled system in closed loop (with man possibly operating in the loop) designed to perform an arbitrarily complex given task, regulation being done using sensor feedback.

Any subsystem has four components:

(1) an action system generates low-level actions toward the real world (in terms of physical quantities acting on the real world) in order to perform a request (action of upper level of abstraction). With such a definition, this component is a specific implementation among many other possible of an inverse model - action oriented model,

(2) a sensor system is at the contrary designed to collect physical quantities from the physical world and to translate them into informations expressed in the same units as the input of the action system. This component performs a direct physical (approximate) model - or knowledge oriented model,

(3) a decision system to control the loop. This decision system may be realized through hardware (low-level control), or be implemented through software, or consist in human supervision or presence (any hybrid solution is possible),

(4) a communication component whose purpose is to ensure communication with other subsystems and exception handlers.

Designing a robotic system, and testing its ability to perform reliably a set of tasks, are problems that require the collaboration of many different fields of physics and engineering (mechanics, electronics, automation, computer science).

Although it is obviously irrealistic to develop a general multi-purpose simulation tool able to deal with all the heterogeneous features of a robotic system, including all the possible interactions with the environment, dedicated simulation tools may be of great help to study some specific components.

Whatever the general architecture of the system may be, a functional hierarchical scale (in terms of abstraction of variables and data) obviously exists, from low-level devices managing electrical current or tension with gain coefficients, up to the highest levels required by autonomy, managed for instance by AI predicates coupled with world models.

Complexity of software increases in volume and also in abstraction with the level with respect to this scale, so that the software depends less and less on technological components. So, large parts of software should be reusable from one system to another. Reconfiguration efforts should thereby lighten (compared with a complete recomputation), as long as packages are generic enough, and some CASE rules have been applied during package development.

Simulation tools (most of the time not robotics oriented) already exist in the field of mechanics, dynamics and control.

Some begin to appear, dedicated to off-line programming of robotic tasks (mainly for workcell applications), but there is a lack of products emphasizing the development and benchmarking of software components for autonomous robots.

So the decision to undergo such a software project was taken two years ago at Commissariat à l'Energie Atomique. The name of this project is ARES - Atelier Robotique Et Simulation (Robotics and Simulation Software Engineering Tool). An overview of ARES system is given in [1].

II. Simulation in ARES

ARES is mainly designed as an open software environment dedicated to the development and testing of robotics task programs and algorithms. Some additional internal models may be required to study the physics of some specific robotic subsystems.

The main idea is to analyze and test the capability of a user application program, seen as a software component

of a robotic system, to realize a set of tasks when the robot is acting on and inside a modelized world (scene).

The scenario is the evolution of the scene with respect to time, according to mathematical models (world model).

The program controls some of the input variables of the model of the world and is able to inquire about the status of some world variables.

It is important to notice that the robotic system is at the same time an operator which acts on the world, and a component of the world (through its mechanical characteristics).

For this reason, there is always an ambiguity when talking about simulation.

The aim of simulation in ARES is to test the action/perception control process performed by the user application program, more than to study the dynamic properties of the robot as a mechanical system.

The first problem is a robotics problem, the second one a mechanics problem. Nevertheless, the world model must implement accurate dynamic models when the user program emulates a low-level control loop.

This accuracy is not obviously needed when attention is focused on higher levels of decision. It can be assumed that dedicated low-level subsystems do their job correctly in a nominal situation (an uncertainty function is most of the time sufficient to deal with models approximations).

If modelization is appropriate, execution of various scenarii may be considered as a set of low-cost experiments from the point of view of the user program. Running such simulations allow to debug then to test the robustness of the software component.

General functional architecture

The ARES system is organized around a simulation kernel.

This kernel (simulator) intends to replace the 'real world' physics from the user application program point of view.

The simulator communicates with any application program in terms of physical entities. The basic hypothesis of simulation is that these quantities have to be considered as exact.

The simulator is a process which aims at maintaining consistent internal models of the scene.

The evolution of the models with respect to time is driven by a set of equations which modelize the physics of the world (objects of the environment and robots included in the scene, which are then seen as passive components of the world).

The simulator may receive (or not) at any time a set of primitive requests of actions from the user application program which can be classified as:

(1) motion requests.

These requests may be kinematic (expressed in terms of positions, velocities and accelerations) or dynamic (expressed in terms of forces and torques).

(2) tooling requests.

(3) sensoring requests.

In the latter case, the simulator has to return basic variables issued from its internal models (geometric, kinematic, or dynamic variables) to the user application program.

At each step, the simulator performs a set of actions. The simulator:

(1) accepts the requests,

(2) automatically processes the requests by setting and solving an appropriate system of equations (for motion) or by running computational geometry algorithms (for distance sensors or tools),

(3) checks if geometric and kinematic constraints are satisfied and raises error messages if not,

(4) updates its internal models,

(5) outputs sensors primitive data toward the application program.

Most of the efforts of ARES group are focused in developing and improving the functionalities, the quality and the performances of the models of this kernel.

The user application program consists in two components of different nature:

(1) the user program by itself which is a set of code and data that the user intends to test.

The ultimate aim of this program is to be used in real conditions, as a component of the real robotic system.

This program implements algorithms which may involve world models of its own, but do not share any component with the simulator (although some CAD models used by the simulator may be of interest for robotics programs, as discussed in [2] for instance).

A complete independency between the program and the simulator is a necessary (but not sufficient) condition to give some credit to the simulation.

(2) The interface (emulator) between the program and the simulator.

The aim of the emulator is to play the part of the interface between the input/output variables of the program and the input/output variables of the simulator.

The emulator replaces all the intermediate subsystems (software and technological ones) of lower level, that operate between the program and the world.

An ideal subsystem emulator is a software component which respects input/output variables of the real subsystem (with their uncertainties) i.e. reproduces its interfaces, and acts like the real subsystem (has the same transfer function as the real subsystem).

III. The Simulator

Simulation in ARES focuses on synchronized motions of a set of robots interacting within a complex (non static) scene. This implies to be able to modelize large sets of bodies (up to several hundred - fortunately a lot of them do not move in a real scene) and their interactions (joints). This leads to the world kinematic model which is described below.

An important feature is that geometric computations are most of the time needed, either to build models of sensors from range informations and surfacic attributes or to check possible violations of geometric constraints. Another consideration which stresses the importance of geometric modelization is the choice of 3D graphic animation as the main element of the user interface.

Graphic animation is indeed performed during the simulation.

This deliberate choice has an advantage and an inconvenient:

- the advantage is to avoid the development of a specific interface between the simulator and the graphic monitor, which is in this case a subtask of the simulator
- the inconvenient is to slow down the animation. Anyway the animation process by itself is a big machine resources consumer.

III.1 Basic CAD data

(1) <u>body</u>

The basic notion common to all models is the notion of body. A body is assumed to be a rigid solid of homogeneous density. A set of tables of attributes is assigned to each body, which will be used during simulation. Tables contain attributes dedicated to sensor models and motions.

The main tables of attributes are :

* the volumic attributes table which contains geometric 3D models of the body, mainly:
  - a CSG (Constructive Solid Geometry) tree model which is well suited to synthetic construction of complex objects, from primitive elements such as block, sphere, cone, cylinder , torus, using union, intersection and difference operators, and range information acquisition through ray tracing techniques;

- a B-REP (Boundary Representation) model describing the solid by its boundary surfaces. This model allows an explicit representation of the solid in terms of vertices, edges, contours and surfaces, and may be used for anticollision computations, detection and nature identification of contacts between solids, and other purposes. These geometric attributes and related topics are discussed in [2].

* the surfacic attributes table which gathers miscellaneous physical attributes needed by sensors models and dynamic contact models.

* the kinematic attributes table which describes position, velocity and acceleration of the body expressed in world cordinates. This table is created at initialization of the scenario, and is updated during simulation.

* the dynamic attributes table which contains physical constant (with respect to time) quantities used for dynamic simulation, such as density, mass, center of mass, inertia tensor.

* the force and torques attributes table, which describes resultant force and torque at center of mass, also updated during simulation.

All the attributes are optional, and depend on the desired accuracy of the simulation (dynamic quantities are not needed if pure kinematic simulation is performed), and on the nature of sensors.

It is important to let the user free to extend the list of predefined attributes with some of his own, in order to implement his personnal emulated components models.

(2) <u>joint</u>

Bodies are assembled within a scene by joints. Joints may have from zero to six degrees of freedom, and may be of any nature and type:

. technological joints (such as prismatic or rotoid joints,...);

. natural (gravity,..) or circumstancial ones (contact, grasping ...);

A catalogue of the main technological joints (holonomic) and their associated functions, which enable to describe the relative positions, velocities and accelerations of the two adjacent bodies, is provided.

We insist that the user should be free to extend the catalogue with his own nature and type attributes, for special cases where joints do not belong to the predefined list.

Tables of attributes describe the nature and type of the joint on one hand, kinematic attributes status during simulation on the other hand.

(3) <u>assembly</u>

An assembly is a set of bodies connected by technological joints (a body appears as a primitive assembly).

An assembly may have kinematic loops.

The notion of assembly is very important, both for CAD description (bodies within an assembly are described by their relative positions, and not in the scene frame), and for simulation, because motions are driven by joint kinematics.

(4) <u>scene</u>

The scene is the general model of all assemblies and simple bodies in presence. It may be seen as a super assembly (root).

Joints between bodies define the general topology of the scene.

Kinematic (resp. dynamic) attributes of all the bodies in presence define the kinematic (resp. dynamic) current state of the scene during simulation.

The scene may have specific attributes (gravity intensity and direction, coefficients to characterize wave propagation, upon which specific sensors measuring principles are based...) and defines the reference world frame (which is assumed to be Galilean as soon as dynamic models are concerned).

III.2 Motion simulation

* The world (kinematic) model

The choice of emphasizing simulation of motions, versus simulation of physics of technological components, naturally leads to a graph model for representing the scene.

The scene is modelized as a multibody system, which consists in the set of bodies (links) of all robotic systems in presence, and in the bodies constituting their environment.

(1) A node of this graph is a body i, to which a frame Fi (Oi,Xi,Yi,Zi) is attached. The system of vectors Xi,Yi,Zi is orthonormal, and has a direct orientation.

The coordinates of any point P of body i are invariant in this frame.

A frame F0 assigned to a virtual node 0 defines the absolute world coordinates.

(2) An edge of the graph is a joint between (exactly) two nodes.

An orientation is given to the edge, so that the frame associated with its terminal node j can be deduced from the frame of its initial node, knowing a joint (4X4) homogeneous matrix Lij(Q,t) which expresses the change of coordinates of any point from frame j to frame i.

Q is a the vector of joint parameters (of dimension 0 to 6) and t the time variable.

The orientation of the edge is chosen arbitrarily (actually, the orientation that gives the simplest expression of L matrix is chosen).

(3) The notion of subgraph is also used for two main reasons :

* the first one is to allow a synthetic description of technological assemblies (such as robot arms), and expression of action requests (such as attachment of a tool or any other object to an arm) ;

* the second one is to implement justified approximations during execution of the scenario (such as neglecting the very small motion of a heavy unmoving carrier induced by the motion of a light carried object).

Finally, a node of the graph may be either a body or another graph (subgraph). an edge is a joint between two bodies, two assemblies or one assembly and one body (the link of the assembly to which the joint applies has to be specified).

* kinematic attributes of components

Each body and joint have a set of kinematic attributes tables, that are automatically created by the simulator at the initialization of the scenario, and then maintained during its execution.

For a body, the kinematic quantities needed are the absolute location matrix (in world frame F0) and its first and second order time derivatives.

Location matrix is a 4X4 homogeneous matrix containing a 3X3 rotation matrix and a translation vector.

Derivatives of matrix of such form are represented as a couple of vectors, one for translation velocity (resp. acceleration) , the other for angular velocity (resp. acceleration).

For a joint, kinematic attributes are :

. the actual values of the vector of Q parameters at current time t, and the first and second order time derivatives of this vector;

. the joint matrix Lij(Q,t) describing the transformation of coordinates from frame j (terminal link of the joint) to frame i (initial one)

. the first and second order partial derivatives of Lij with respect to each Qk parameter (and possibly time variable), expressed in frame i;

Other kinematic attributes are also needed for a joint:

. the first and second time derivatives of joint matrix, expressed in frame i;

. the first and second time derivatives of joint matrix, expressed in frame 0;

These attributes are automatically computed from previous tables.

We obtain a very general and uniform model, able to deal with any type of joint, as soon as joint attributes (nature and type of the joint, and associated mathematical models to be performed during simulation) are expressed.

This formalism offers a set of advantages:

(1) defining assemblies and scene is easy.
(2) implementing recursive algorithms to update kinematics attributes during simulation is possible
(3) updating the topology of the graph, i.e. adding one or several bodies to the scene (apparition of a new obstacle, action of a tool which cuts one body into several parts), creating a joint, deleting a joint ('natural contact') is easy;
(4) the link with modern hierarchical graphic 3D structures (PHIGS package) is easy.

* motion computation

Motions of robotic (active) components are driven by user application program, through the emulator.
The actual simulation is kinematic (no dynamics calculations are done).
In this mode, trajectories of passive bodies have also to be provided, in the description of the scenario.
A more realistic calculation (including dynamics) is under development.
Motions of others components will be deduced from the laws of dynamics.
Both modes of simulation are supported by the world kinematic model.
In dynamic mode, dynamic attributes tables are taken into account.

* constraint checking

Topologic constraints induced by closed kinematic chains are implemented in the set of kinematic equations by adding closure conditions, for any elementary cycle detected in the graph.
Associated equations in dynamic model are included using Lagrange multipliers.
Geometric constraints are detected by computational geometry methods, applied to volumic models of bodies.

III.3 Basic services for sensor emulation

The emulator directly gets basic informations about positions, velocities, accelerations, forces and torques by inquiring the status of internal variables of the world model.

So the main efforts have been focused on distance computations.
Ray tracing techniques are applied on CSG volumic models of bodies in the scene.
The informations returned to the emulator are:
- the distance in one fixed direction
- the first body (and surface) encountered in this direction
- the normal vector to the surface at the intersection point.

Rays may have an infinite or finite range.
Emulated sensor routines create their own outputs from this information (as discussed in [1]).
Multiple reflections can be computed, and enlightment can be simulated if needed.
The algorithm runs recursively through the branches of CSG tree, and performs an intersection computation when reaching a leaf (simple model).
In order to speed up computations, the scene is partitioned into boxes.
The only bodies candidate for intersection computations are those which intersect a box located in the incident direction.
Another implemented method consists in including the bodies in spheres, and to first estimate whether the ray intersects the sphere.

III.4 Graphic monitor

The graphic monitor is a subtask of the simulator, which manages 3D graphic objects organized hierarchically.

As we do not intend to make any developments in the field of computer graphics, we chose the PHIGS ANSI standard to support the graphic interface.

PHIGS concepts fit our modelization requirements very well:

Trees of PHIGS graphic models can be easily derivated from our general graph model. Both wireframe and volumic graphic models can also be easily obtained from our boundary representation model.

Moreover, thanks to ANSI standard, PHIGS libraries are linked with hardware graphics processors of most workstations.

Graphic objects are created at the initialization of the scenario, and their positions are updated with respect to simulator computations.

The graphic monitor eventually offers a set of services to the observer, such as view manipulations (rotations, translations, zooming), manipulation of visualization attributes such as colors, projection types..., control of the display rate, display and layout of various informations about status variables - such as velocities or joint variables curves.

IV. ARES and Ada programming language

The ARES simulation kernel used up to now as a support for demonstrators is written in FORTRAN 77 language.

The concept of clear separation betwwen the user application program and the simulator is already applied in this version (programs driving simulations have been developped in C and/or FORTRAN language).

The two entities communicate via IPC (Inter Process Communication) services of UNIX System V operating system.

But FORTRAN has shown its limitations when one has to deal with complex interconnected data structures, such as described above, especially in development stages. Neither FORTRAN nor C provide sufficient type abstraction and packaging (compiler controlled) facilities that are necessary conditions for the development of an open system.

We were interested in Ada programming language from the beginning of the project, as the programming language of the simulator, because of these features. Unfortunately, Ada compilers were at this time full of residual bugs, and associated toolkits such as mathematical or graphic libraries were very poor. The situation greatly improved during the last two years, so we decided to rewrite the simulator in Ada.

Real work on the Ada develoment of the simulator started at the beginning of this year, and we are currently finishing a geometric modeler (25000 Ada source code lines). This simulator is much more powerful than the FORTRAN version. Other developments in Ada on world modelization are going on.

Our experience with Telesoft compiler running on SUN 4 workstations is that Ada programs are very readable, maintainable, and may easily and very quickly restructured with complete reliability.

Moreover, we discovered that Ada programming language is a nice choice as a CAD language for body, assembly and scene description. These properties are illustrated in [2].

The last feature which stresses our interest in Ada is that the language is claimed to have been designed for real time system development as well as for large software applications. Studies (beyond ARES scope) have started at CEA to evaluate these properties for robot low-level programming.

V. Conclusion

This paper mainly describes the specifications and ongoing developments about ARES simulation kernel.

Some robotics studies have been undertaken meanwhile, based upon the preliminary version of this kernel:

- sensor based control algorithms
- trajectory generation for mobile robots
- navigation algorithms
- off-line programming of a robot arm by a textual language
- coupling with AI tools for robotics mission preparation and execution monitoring.

REFERENCES

[1] - A. COSSIC, J.P. NOMINE, A. MALAVAUD, M. DETOC, "ARES Base Case Demonstration", CEA/ORNL Workshop on Autonomous Mobile Robots, Oak Ridge, Tennessee, 1989.

[2] - G. DEJONGHE, A. COSSIC, D. CHAIGNE, "Environment Modeling for Robotics Simulation", CEA/ORNL Workshop on Autonomous Mobile Robots, Oak Ridge, Tennesse, 1989.

Session II: Perception for Mobile Robots

SYMPATI 2

AN ADVANCED VISION SYSTEM FOR ROBOTICS

Didier JUVIN

I INTRODUCTION

II THE LINEAR PROCESSOR STRUCTURE

III THE PROCESSING ELEMENT (P.E.)

IV PERFORMANCES

V CONCLUSION

cea DIVISION leti

SYMPATI 2

AN ADVANCED VISION SYSTEM FOR ROBOTICS

Didier JUVIN

I INTRODUCTION

II THE LINEAR PROCESSOR STRUCTURE

III THE PROCESSING ELEMENT (P.E.)

IV PERFORMANCES

V CONCLUSION

SYMPATI 2

AN ADVANCED VISION SYSTEM FOR ROBOTICS

Didier JUVIN

I INTRODUCTION

II THE LINEAR PROCESSOR STRUCTURE

III THE PROCESSING ELEMENT (P.E.)

IV PERFORMANCES

V CONCLUSION

SYMPATI 2

AN ADVANCED VISION SYSTEM FOR ROBOTICS

Didier JUVIN

I INTRODUCTION

II THE LINEAR PROCESSOR STRUCTURE

III THE PROCESSING ELEMENT (P.E.)

IV PERFORMANCES

V CONCLUSION

cea DIVISION leti

SYMPATI 2

AN ADVANCED VISION SYSTEM FOR ROBOTICS

Didier JUVIN

I INTRODUCTION

II THE LINEAR PROCESSOR STRUCTURE

III THE PROCESSING ELEMENT (P.E.)

IV PERFORMANCES

V CONCLUSION

cea DIVISION leti

SYMPATI 2

AN ADVANCED VISION SYSTEM FOR ROBOTICS

Didier JUVIN

I INTRODUCTION

II THE LINEAR PROCESSOR STRUCTURE

III THE PROCESSING ELEMENT (P.E.)

IV PERFORMANCES

V CONCLUSION

cea DIVISION leti

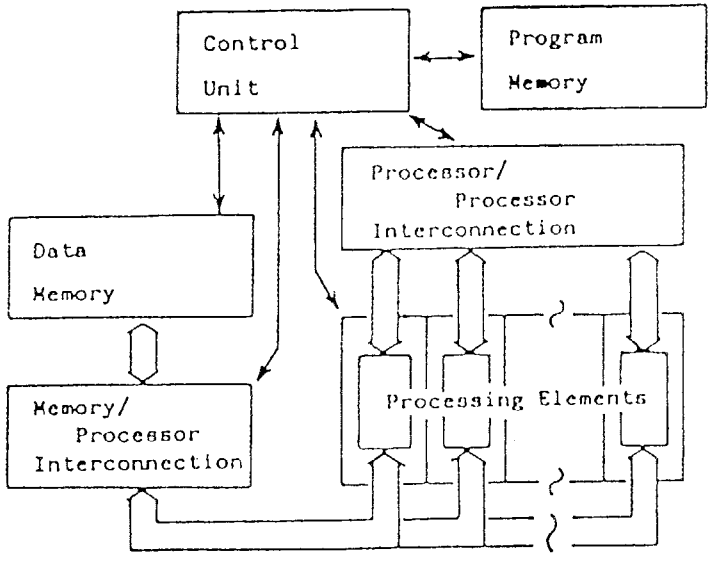
I - INTRODUCTION

IMAGE PROCESSING :

IMAGE	PIXEL LEVEL	REGION LEVEL	DECISION
	Filtering Enhancement Restoration Feature extraction Segmentation	Pattern Recognition ...	

Data Base

cea DIVISION leti



PROCESSOR ARRAY GENERAL STRUCTURE

S Y M P A T I 2

Y

M

P

A a Linear Processor ARRAY 1.5 D

T

I

2

PRINCIPLES :

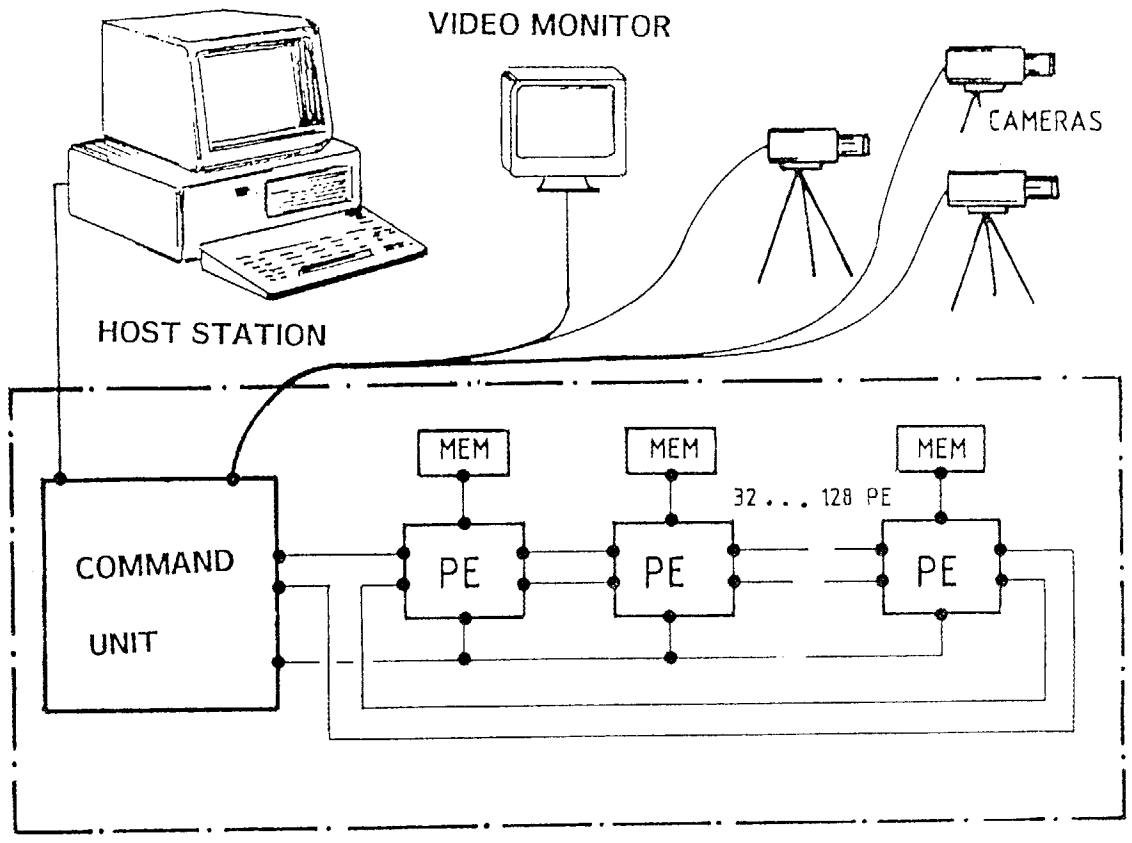
- SIMD Processor Array
- 32 to 128 processing elements (P.E.) working in parallel on the image without access conflict and without border effects

SYMPATI 2 = LINEAR PROCESSOR ARRAY

+ LARGE NEIGHBORHOOD DIRECT ACCESS

+ HELICOIDAL DATA ORGANIZATION

cea DIVISION leti

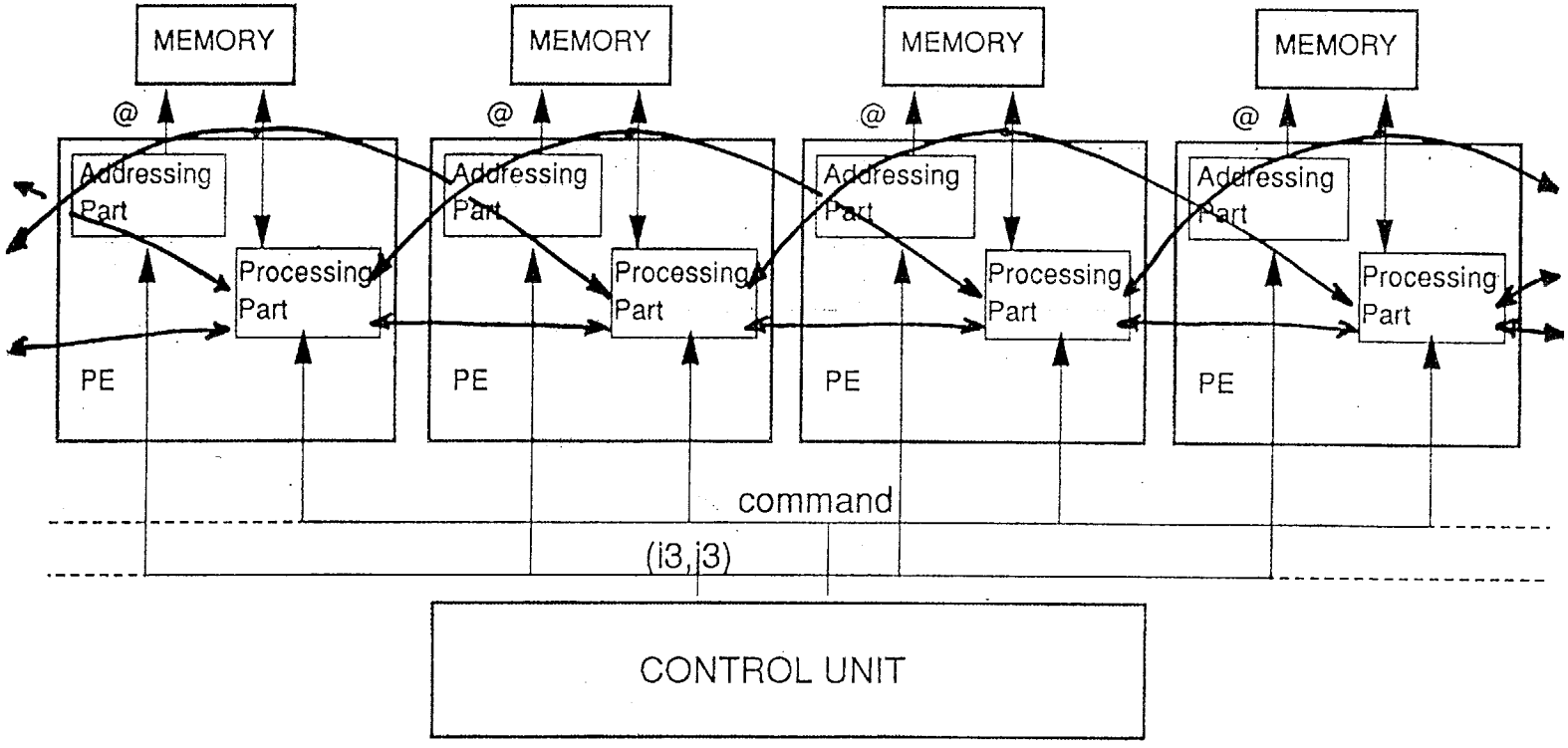


SYMPATI 2 SYSTEM

cea DIVISION leti

II - THE LINEAR PROCESSOR STRUCTURE

- GENERAL PRESENTATION
- P.E. INTERCONNECTION
- HELICOIDAL DATA STRUCTURE


MEMORY
MEMORY
MEMORY
MEMORY
@
@
@
@
Addressing Part
Addressing Part
Addressing Part
Addressing Part
Processing Part
Processing Part
Processing Part
Processing Part
PE
PE
PE
PE
command
(i3, j3)
CONTROL UNIT

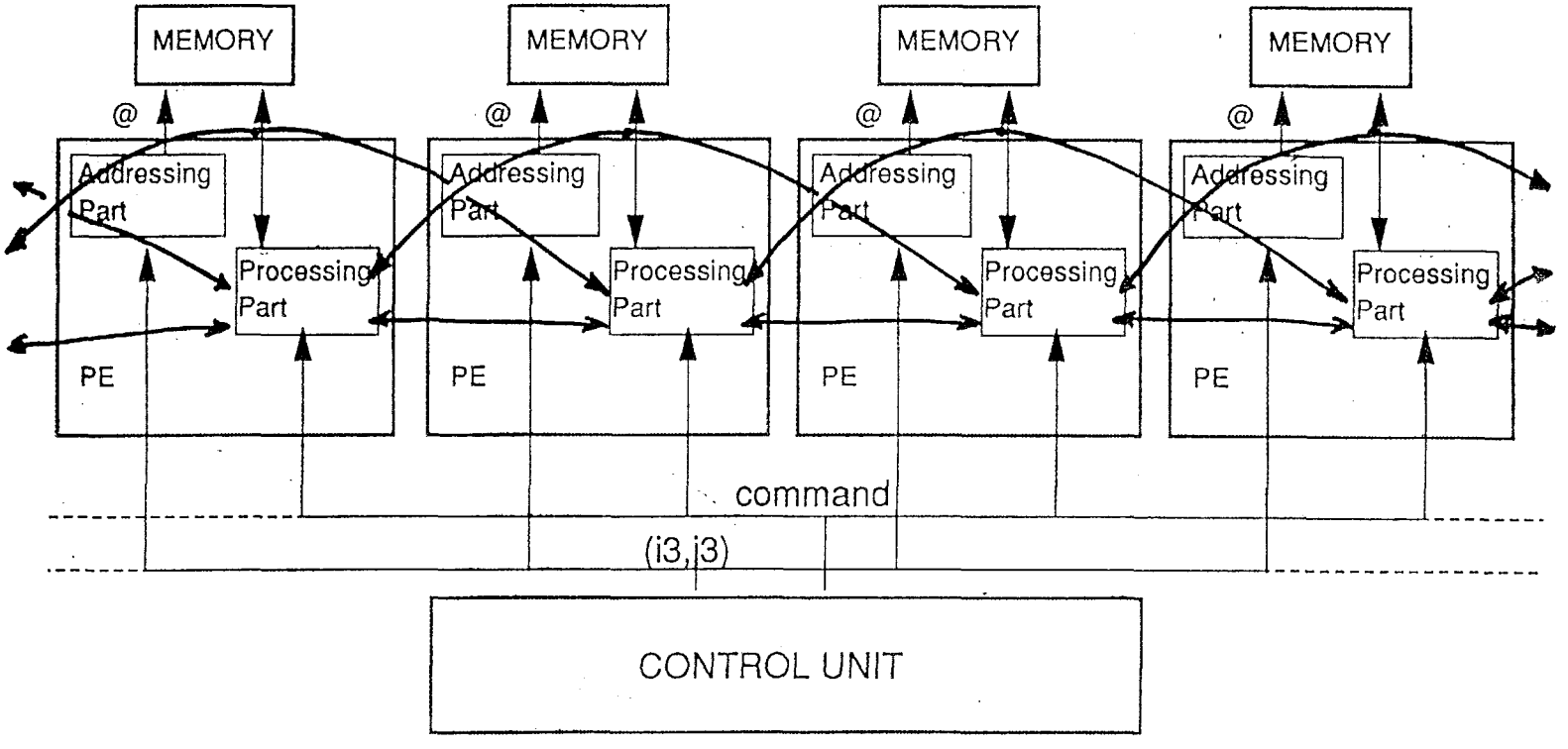
INTERCONNECTION

possible access right or left :

SP = F (SP,L) up to distance 3

M = F (SP,L) ‖ up to distance 2

SP = F (M,L) ‖


MEMORY
MEMORY
MEMORY
MEMORY
@
@
@
@
Addressing Part
Addressing Part
Addressing Part
Addressing Part
Processing Part
Processing Part
Processing Part
Processing Part
PE
PE
PE
PE
command
(i3,j3)
CONTROL UNIT

THE HELICOIDAL DATA STRUCTURE

Problem : To organize 2D structured data in a linear way with the following properties :

- possible access without confict to any row or column located anywhere in the image

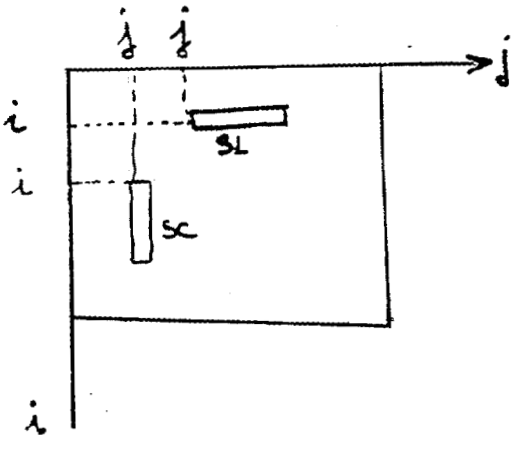
- topological image properties conserved in the two canonic directions.

Memory Bank	0	1	2	3	--	M-1
0	(0,0)	(0,1)	(0,2)	(0,3)		(0,M-1)
1	(0,M)	(0,M+1)	(0,M+2)	(0,M+3)		(0,2M-1)
:	:	:	:	:		:
N/M	(0,N-M)	(0,N-M+1)	-	-	--	(0,N-1)
:	(1,M-1)	(1,0)	(1,1)	(1,2)	--	(1,M-2)
:	(1,2M-1)	(1,M)	(1,M+1)	(1,M+2)	--	(1,2M-2)
:	:	:	:	:		:
:	(1,N-1)	(1,N-M)	:	:	--	(1,N-2)
:	(2,M-2)	(2,M-1)	(2,0)	(2,1)	--	(2,M-3)
:	(2,2M-2)	(2,2M-1)	(2,M)	(2,M+1)	--	(2,2M-3)
:	:	:	:	:		:
:	:	:	:	:		(N-2,1)
:	(N-1,1)	(N-1,2)	(N-1,3)	(N-1,4)	--	(N-1,0)
:	:	:	:	:		:
N2/M	(N-1,N-M+1)	--	--	--		(N-1,N-M)

SL (0,3)
SC (0,1)

HELICOIDAL DATA ORGANIZATION

- M Processors
- NxN Image



III - THE PROCESSOR

THE PROCESSOR ADDRESSING PART

The command unit sends to the P.E.

- the segment head ccordinate (i, j)
- the type of scanning SC SL

Each P.E. calculate :

- its own pixel relative coordinate (i^r, j^r)

Horizontal scanning	Vertical scanning
$i^r = i + (NUMPE-k) \text{ Mod } M$	$i^r = i$
$j^r = j$	$j^r = j + (NUMPE-K) \text{ Mod } M$

$$k = (i + j)_{\text{MOD}\,M}$$

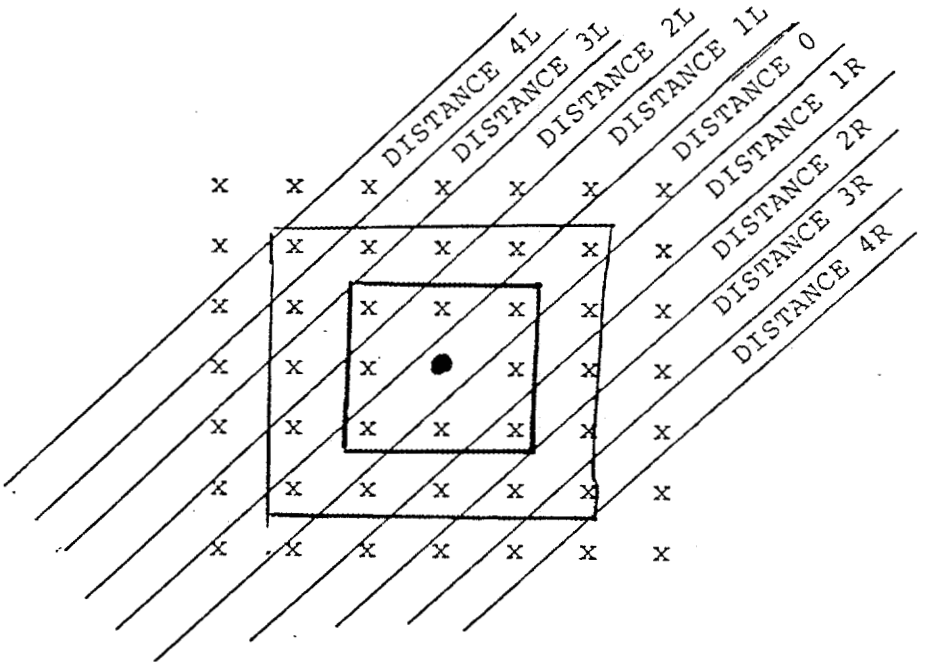
- performs helicoïdal transformation (address in its band)

$$\boxed{\omega = i^r * d + j^r/M \qquad d = N/M}$$

- compares its own pixel relative coordinate to window coordinates and inhibits, if necessary the process.

An other type of addressing is the tabulation mode.

CORRESPONDANCE IMAGE ⇔ HELICOIDAL DATA ORGANIZATION



● = processed pixel

HELICOIDAL SCHEME

⊕

DISTANCE 3 ACCESS

⇒

LARGE NEIGHBORHOOD

DIRECT ACCESS

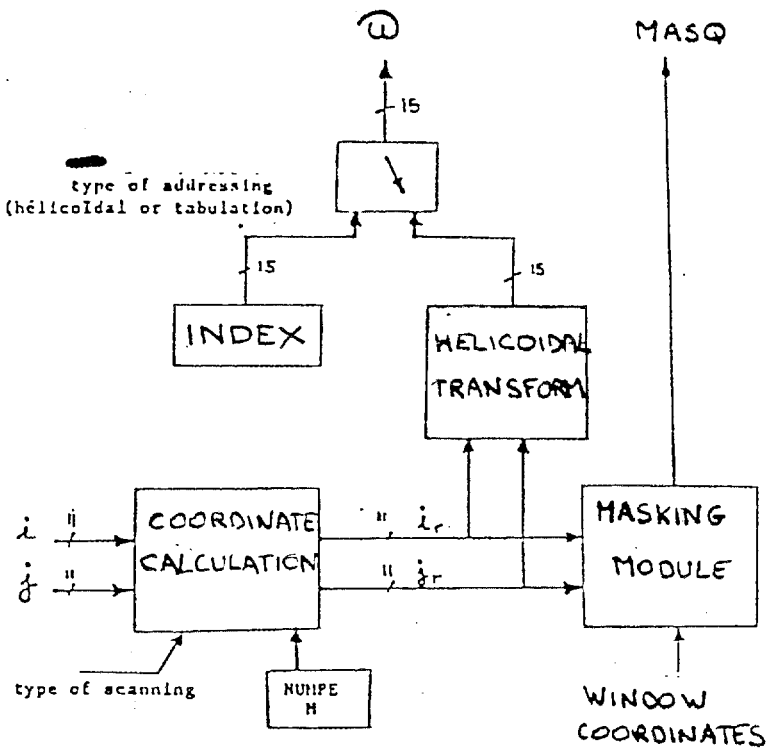
III - THE PROCESSING ELEMENTS

- ADDRESSING PART
- PROCESSING PART

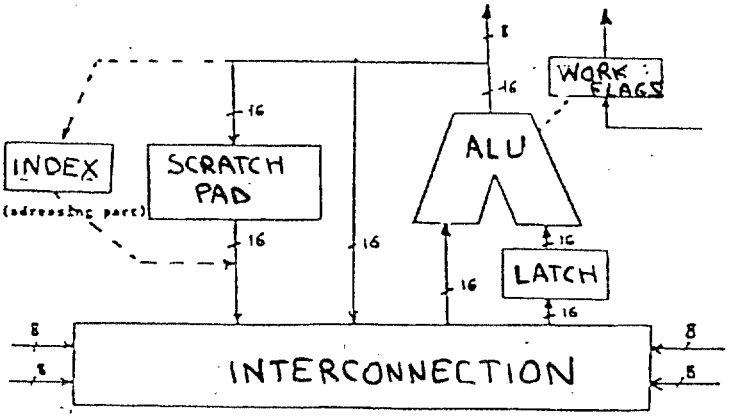
III - THE PROCESSING ELEMENTS

- ADDRESSING PART
- PROCESSING PART

THE ADDRESSING PART :



THE PROCESSING PART :



HARDWARE PRESENTATION

SYMPATI 2 is a modulal structure Multibus based.

- 1 COMMAND UNIT CARD { Video interface / Host processor interface
- 1 to 4 P.E. CARD (320 to 1200 MIPS)

Realization of a standard cell 20.000 gate ASIC integrating 4 P.E.

1 CYCLE TIME = 100 NS

= (OPERAND ACCESS DISTANCE 0 to 3 + ALU OPERATION + STORAGE)

IV - PERFORMANCES

. PROGRAMMING ENVIRONMENT :

- 4 LP (Low Level Language for Line Processor)
- Compiler, Desassembler
- Debugger
- Simulator

. ICONIC ALGORITHMS WELL SUITED FOR SYMPATI 2 :

- Gradient operators : Prewitt, Sobel, Roberts, Merö-Vassy
- Laplacian operator
- Filters : IIRF, FIRF
- Edge extractors : Canny, Deriche, Chen
- Texture extractors : Law-mask, Co-occurency
- non linear filters : median, local means, Min-Max ....
- morphology : binary or grey
- geometrical transformations
- Inter or Intra Image Operation
- Optical flow computations
- Segmentation : labelling
- Skelettonnization
- Hough transform

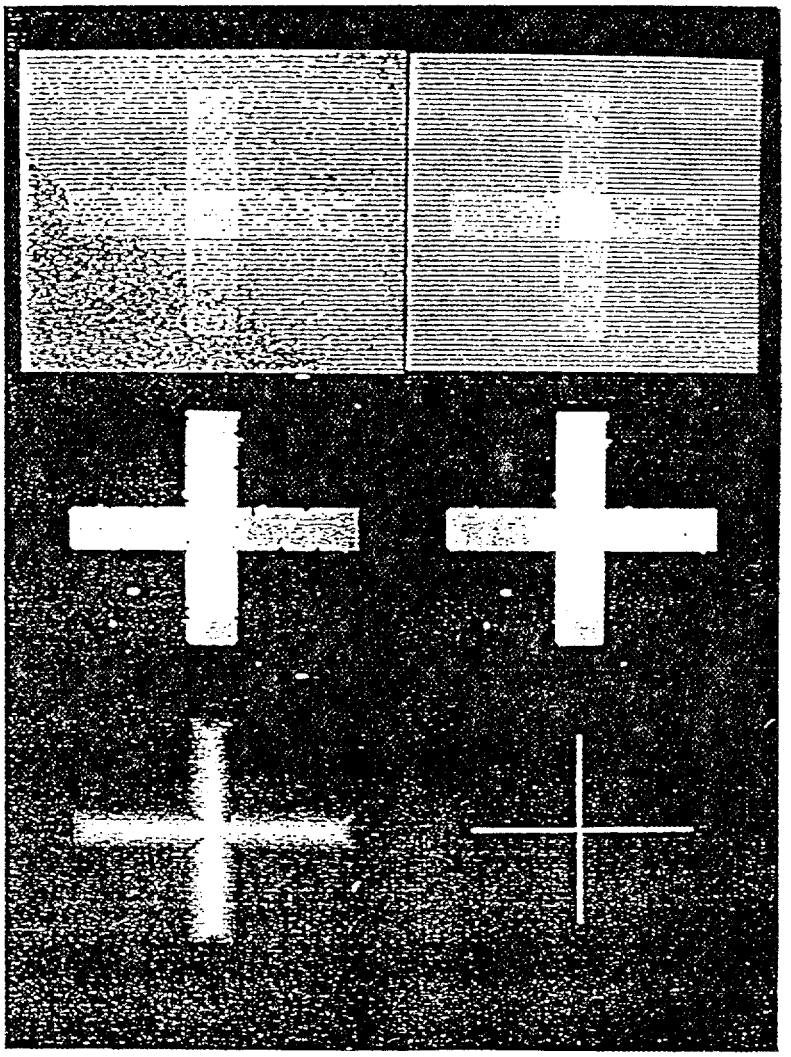
leti DIVISION

RESULTS : examples on 256 x 256 x 8 bits

	32 PE	128 PE
3x3 Convolution	3 ms	0.75 ms
5x5 Law mask	15 ms	3.75 ms
Contour extractor	5 ms	1.2 ms
Texture extractor	160 ms	40 ms
Labelling, encoding	*	*
Hough transform	7 ms	3ms
Motion extractor	8.5 ms	2.1 ms
Connectionnism simulation (256 neurons Hopfield network)		
learning	0.7 ms	
relaxation	0.4 ms	

ABINGDON CROSS BENCHMARK QF = 6

cea DIVISION leti



a : Initial image
b : Filtered image
c : Thresholded filtered image
d : Cleaned binary image
e : Boader distances
f : skeleton

FIGURE 1

Output of the main step

TABLE 1.

The different steps required for the Abingdon Cross Benchmark.

	Number of cycles per segment
1. Pseudo-Tukey	18
2. Thresholded	4
3. OR	9
4. AND	9
5. Boarder distances	21
6. Skeleton	33
7. Thresholded	4
	========
TOTAL	98

TABLE 2.

Total execution time for different configuration and image sizes

N	128x128	256x256	512x512
#PE			
32	5,017ms	20,07 ms	80,2 ms
64	2,5 ms	10,03 ms	40,1 ms
128	1,25 ms	5,017ms	20,07ms

TABLE 3.

Quality factor with 128 x 128 image for different configurations.

#PE	QF = N/T	Magnitude
32	25513	5
64	51200	5
128	102400	6

Table 5
PRIP System Quality Factor Magnitudes

PRIP System	Magnitude 1	2	3	4	5	6
1024XM (Megavision)	-	-	E	-	-	-
AIS1000 (Applied Intelligent Systems)	-	-	-	E	-	-
AIS5000 (Applied Intelligent Systems)	-	-	-	-	A	-
CAAPP (University of Massachusetts)	-	-	-	-	-	H
CAM-6 (Systems Concepts)	-	-	E	-	-	-
CLAP (Cellular Logic Systems)	-	-	H	-	-	-
CLIP4 (University College London)	-	-	-	A	-	-
CLIP (Stonefield-Omicron)	-	-	E	-	-	-
CLOPIPE (Johns Hopkins University)	-	-	A	-	-	-
CM-1 and -2 (Thinking Machines)	-	-	-	-	A	-
CYTO II (Environmental Research Institute of Michigan)	-	A	-	-	-	-
CYTO III (Environmental Research Institute of Michigan)	-	-	A	-	-	-
CYTO-HSS	-	-	-	-	-	-
DAP (International Computers Ltd.)	-	-	-	-	E	-
DAP 510 (Active Memory Technology)	-	-	-	-	A	-
diff3/GLOPR (Coulter Biomedical Research)	-	-	-	-	E	-
DIP (University of Delft)	-	A	-	-	-	-
FLIP (FIM, Karlsruhe)	-	E	-	-	-	-
GAPP (Martin Marietta)	-	-	-	-	A	-
GENESIS/1 MACH V-1 (Machine Vision International)	-	-	-	A	-	-
GENESIS/1 MACH V-7 (Machine Vision International)	-	-	-	-	E	-
GE/WARP	-	-	E	-	-	-
IP8500 (ETH Zurich)	-	A	-	-	-	-
IP9200 (Perceptics)	-	-	A	-	-	-
Magiscan-2 (Joyce-Loebl)	-	A	-	-	-	-
MaxVideo (DataCube)	-	-	-	A	-	-
MV2000 (Machine Vision International)	-	-	-	A	-	-
MPP (NASA Goddard)	-	-	-	-	E	-
MVP/AT (Matrox)	-	-	E	-	-	-
PHP (Carnegie Institute of Technology)	A	E	-	-	-	-
PICAP (University of Linkoping)	-	E	-	-	-	-
PIP4000 (ADS Company Ltd.)	-	A	-	-	-	-
PIP4500 (ADS Company Ltd.)	-	-	E	-	-	-
PIXAR/1chap (Pixar)	-	-	-	A	-	-
PIXAR/3chap (Pixar)	-	-	-	E	-	-
POP II (Royal Holloway College)	A	-	-	-	-	-
PSICOM 327 (Perceptive Systems)	-	A	-	-	-	-
Scope-20 (Symbolics) 1024	-	-	A	-	-	-
SPDS (Amber Engineering)	-	-	-	-	E	-
SYMPATI (CERFIA)	-	-	-	-	-	H
TAS-Plus (Leitz GmbH)	-	-	E	-	-	-
TERAGON (Teragon)	-	-	A	-	-	-
TOSPIX II (Toshiba)	-	-	A	-	-	-
TRAPIX 5500 (Recognition Concepts Inc.)	-	E	-	-	-	-
VAP (University of Berne)	-	A	-	-	-	-
VIA 1000 (Boeckeler Instruments)	-	E	-	-	-	-
VICOM 10 (Vicom)	-	A	-	-	-	-
VICOM VME (Vicom)	-	-	-	E	-	-
VITec (Visual Information Technologies)	-	-	-	A	-	-
WARP (Mellon Institute)	-	-	A	-	-	-

CONCLUSION

SYMPATI 2 properties :

- LOW COST
- MODULAR STRUCTURE
- REAL PARALLELISM
- NO BORDER EFFECTS
- USER FRIENDLY

cea DIVISION leti

Concurrent Computer Vision on a Hypercube Multicomputer

J.P. Jones

Advanced Computers and Integrated Sensor Systems
Center For Engineering Systems Advanced Research
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6364 USA

jov@stc10.ctd.ornl.gov

Motivation

- increase computing speed
- general purpose support
- research environment

Outline

- hypercubes
- System Design
- Performance

Hypercubes are...

-> Distributed memory

-> Message passing

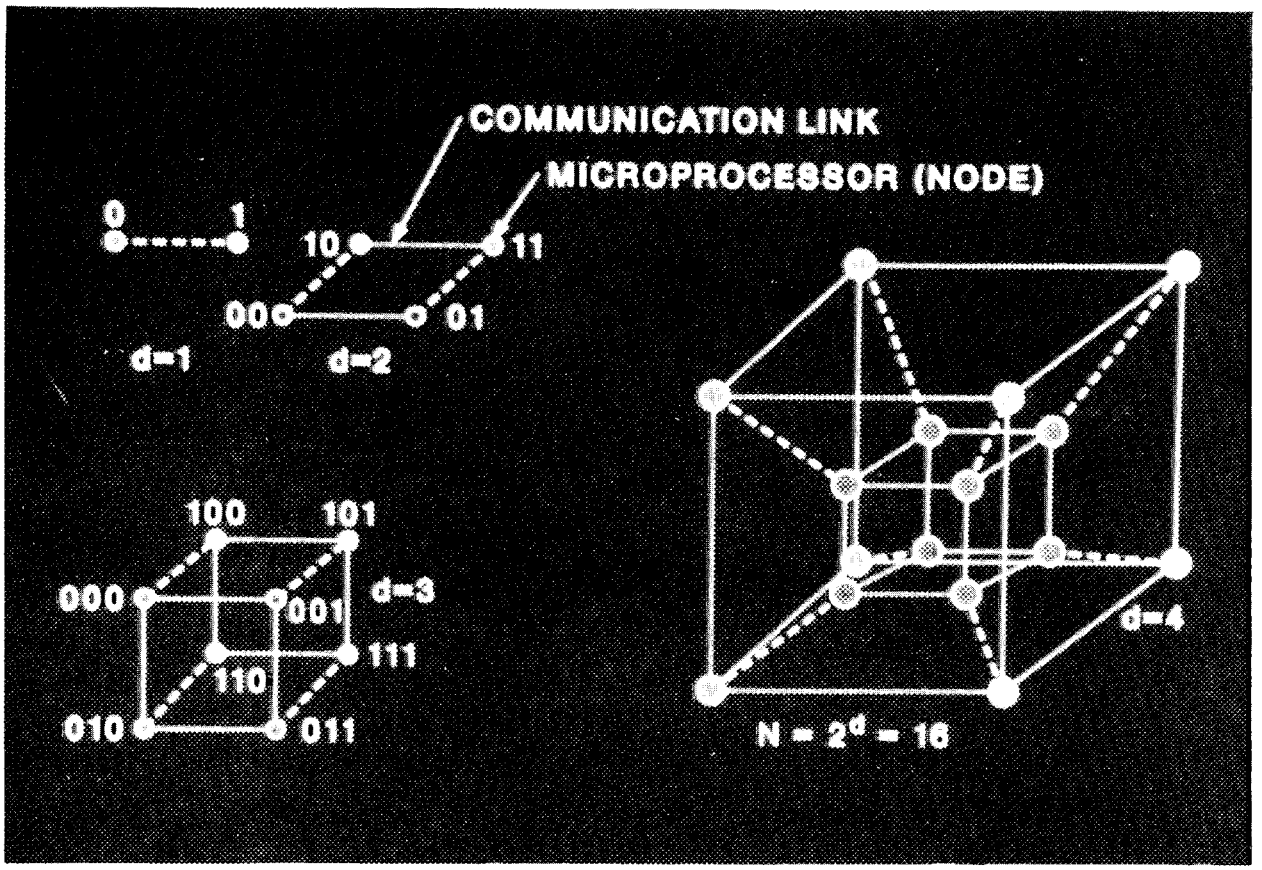
-> (typically) Medium-grained

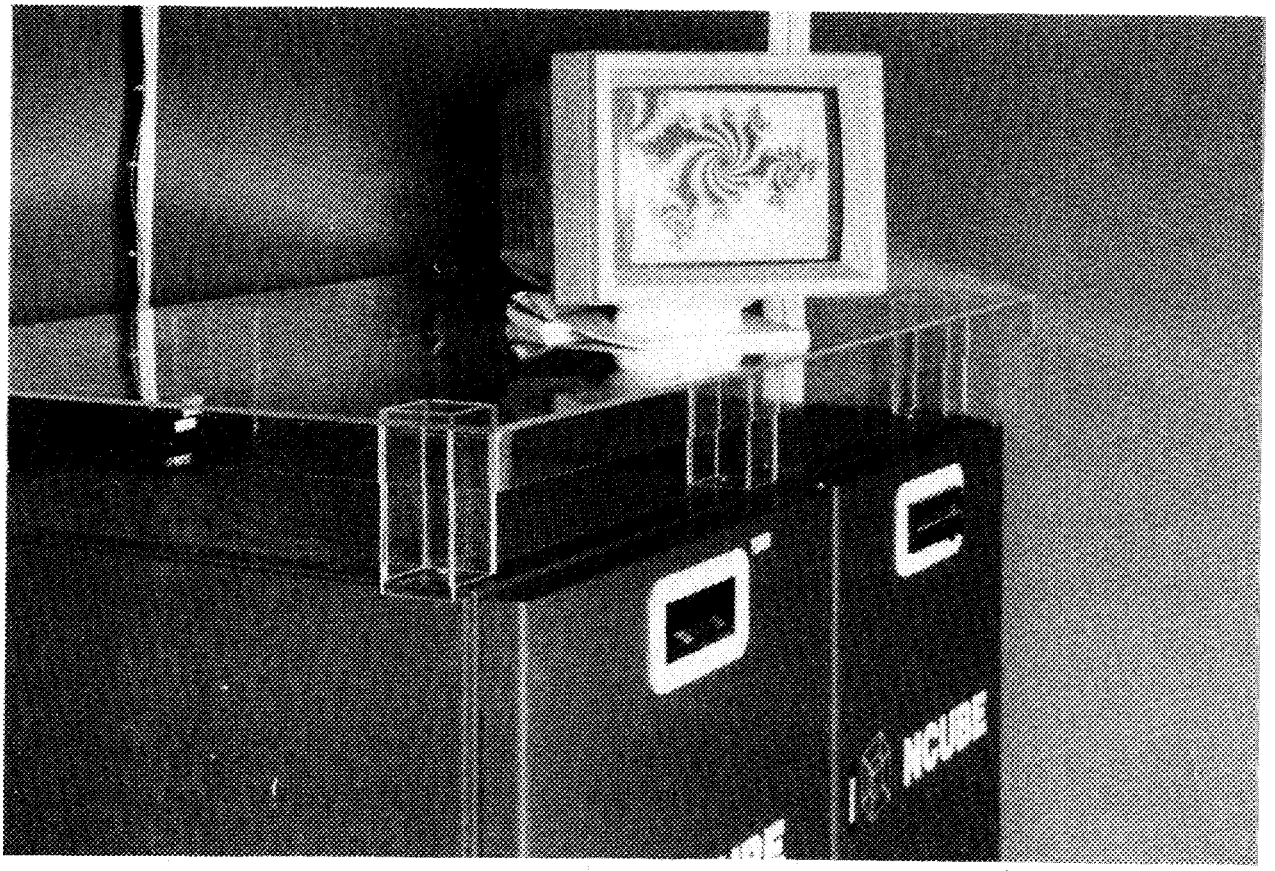
-> Concurrent

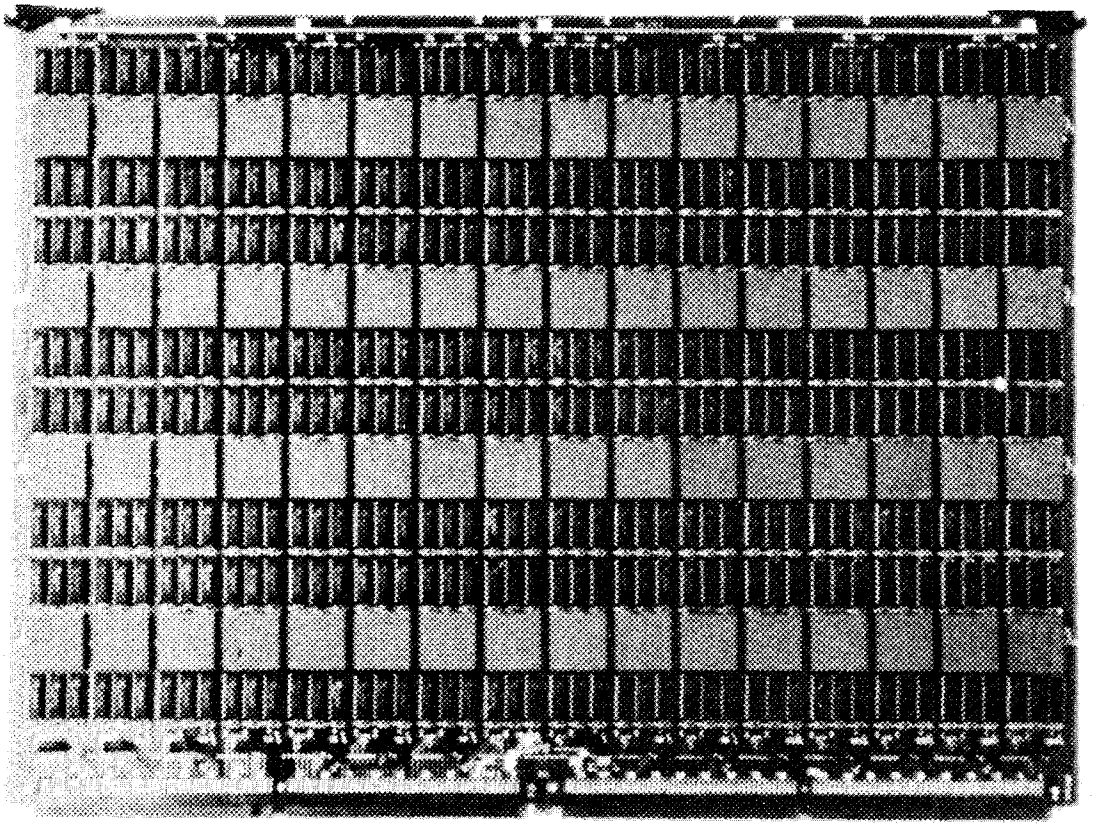
-> Multicomputers

-> lg(N) network diameter

-> O(N lg N) communications channels


COMMUNICATION LINK
MICROPROCESSOR (NODE)
0
1
10
11
00
01
d=1
d=2
100
101
000
001
d=3
111
110
010
011
d=4
N = 2^d = 16


NCUBE



ORNL Concurrent Computer Vision System

Generic Issues:

- Performance
- Minimum communication (M)
- Balanced computation (B)

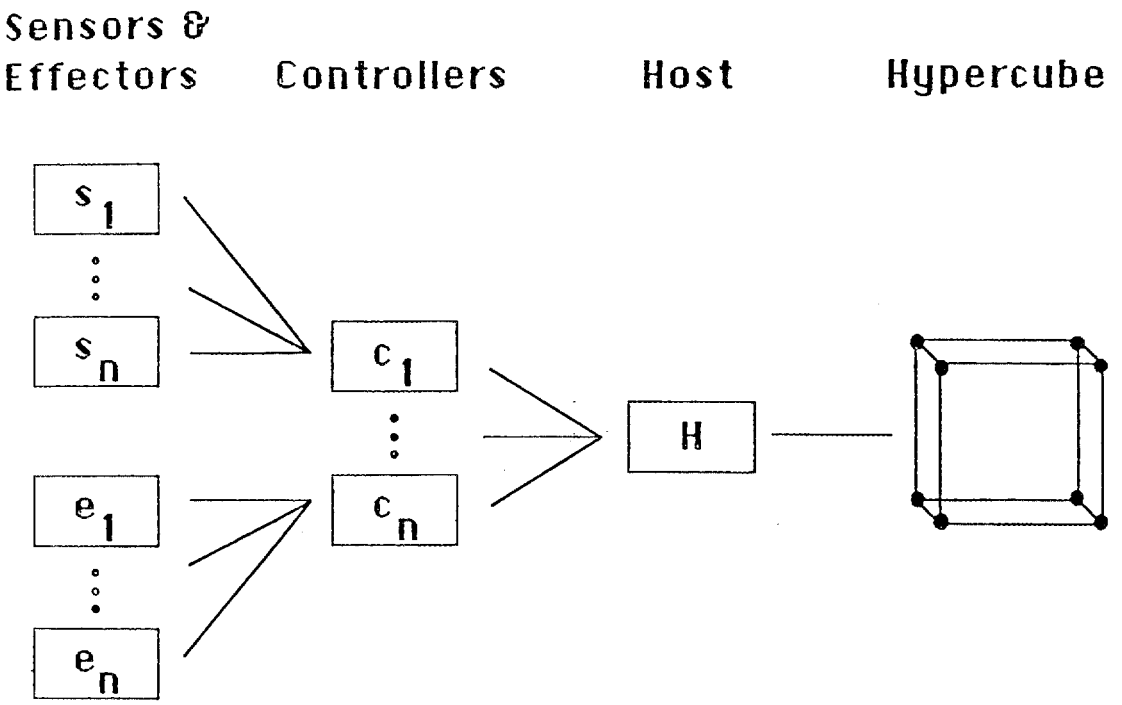
Typical Problem:

Perform a calculation a->b on a hypercube of arbitrary dimension, satisfying M & B

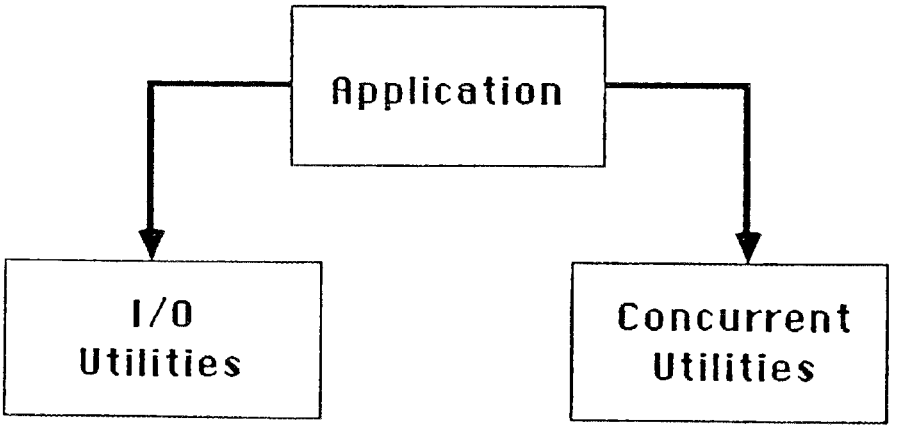
General Objective:

- -> Feasibile system
- -> Generality (Support)
- -> Programmability

CUBIX-like I/O Subsystem



Virtual Sequential Processor

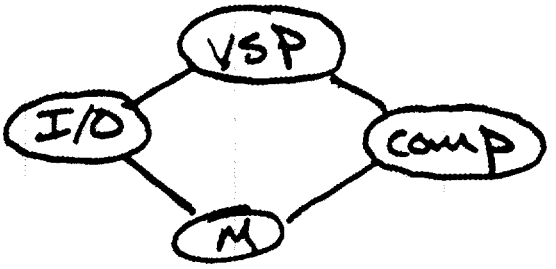


Three Level Abstraction Heirarchy

① Virtual Sequential Processor
- behaves like an ordinary sequential machine

② Concurrent Utilities
- The existence of distributed memory & communication is known

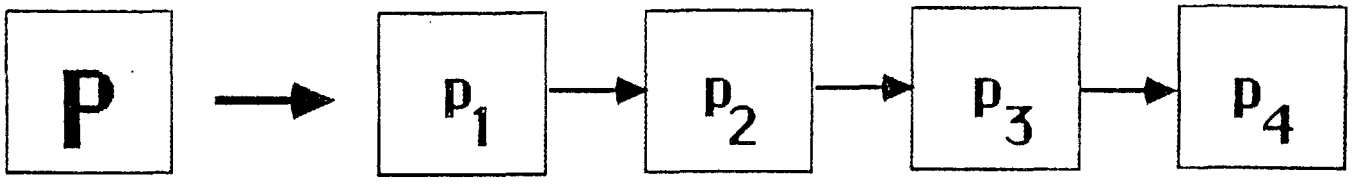
③ Concurrency "Meta-Primitives"
- topology specific



Loosely Synchronous Programming Model

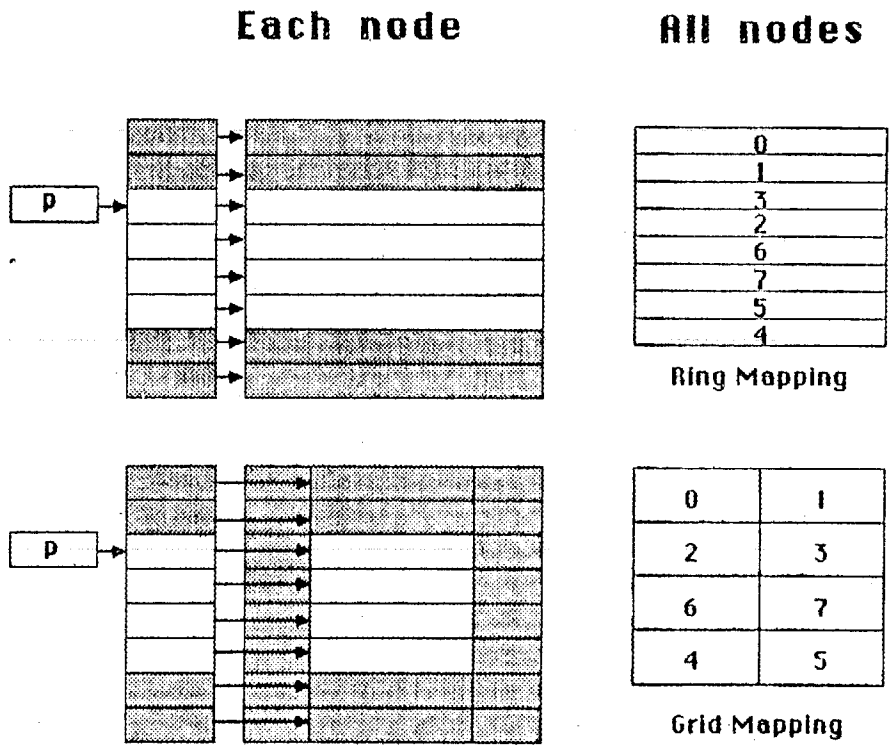
(Caltech)

A (large) problem can be decomposed (in time) into a sequence of sub-problems. These sub-problems can frequently be decomposed (in space) and solved concurrently. Communications impose synchronization.



Domain Decomposition in ORNL Concurrent Computer Vision System

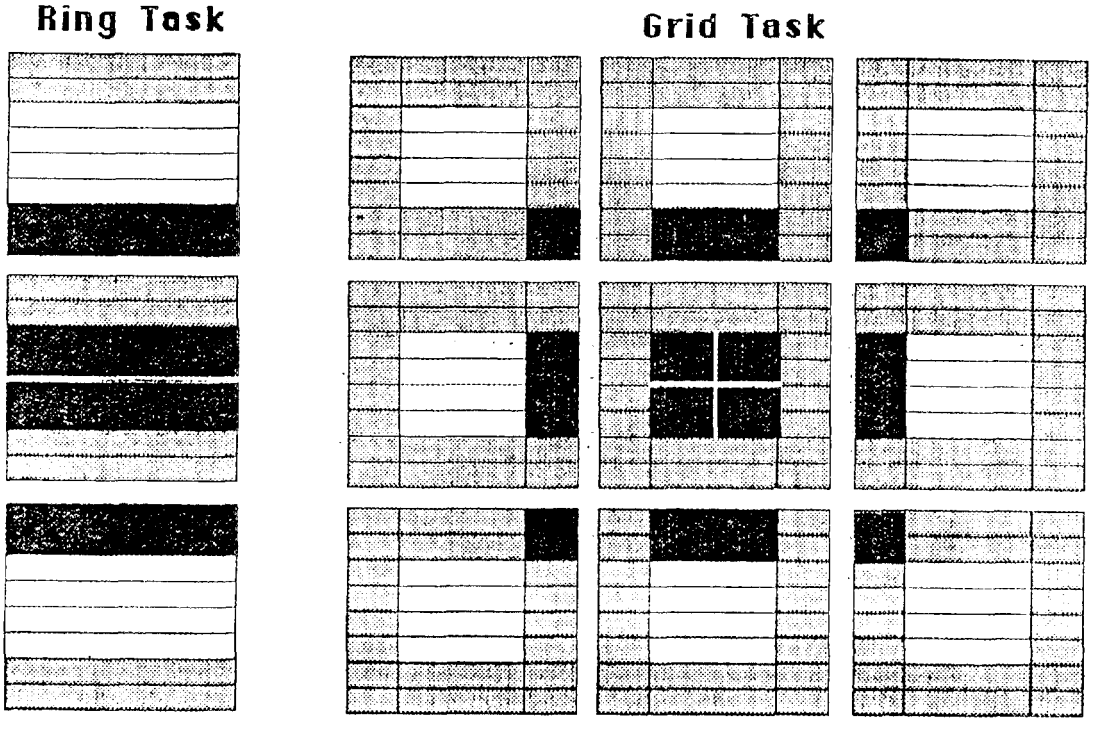
```
struct IMAGE {
    char **p;
    int high,
        wide,
        rast,
        nrows,
        ncols,
        class,
        mapping;
};
```




```
struct IMAGE *imallocc( (int) high, wide, rast, char *class)
```

```
(e.g.)   in_pic = imallocc( 256, 256, 1 , "unsigned char" );
```

Low-Level Communication in ORNL Concurrent Computer Vision System



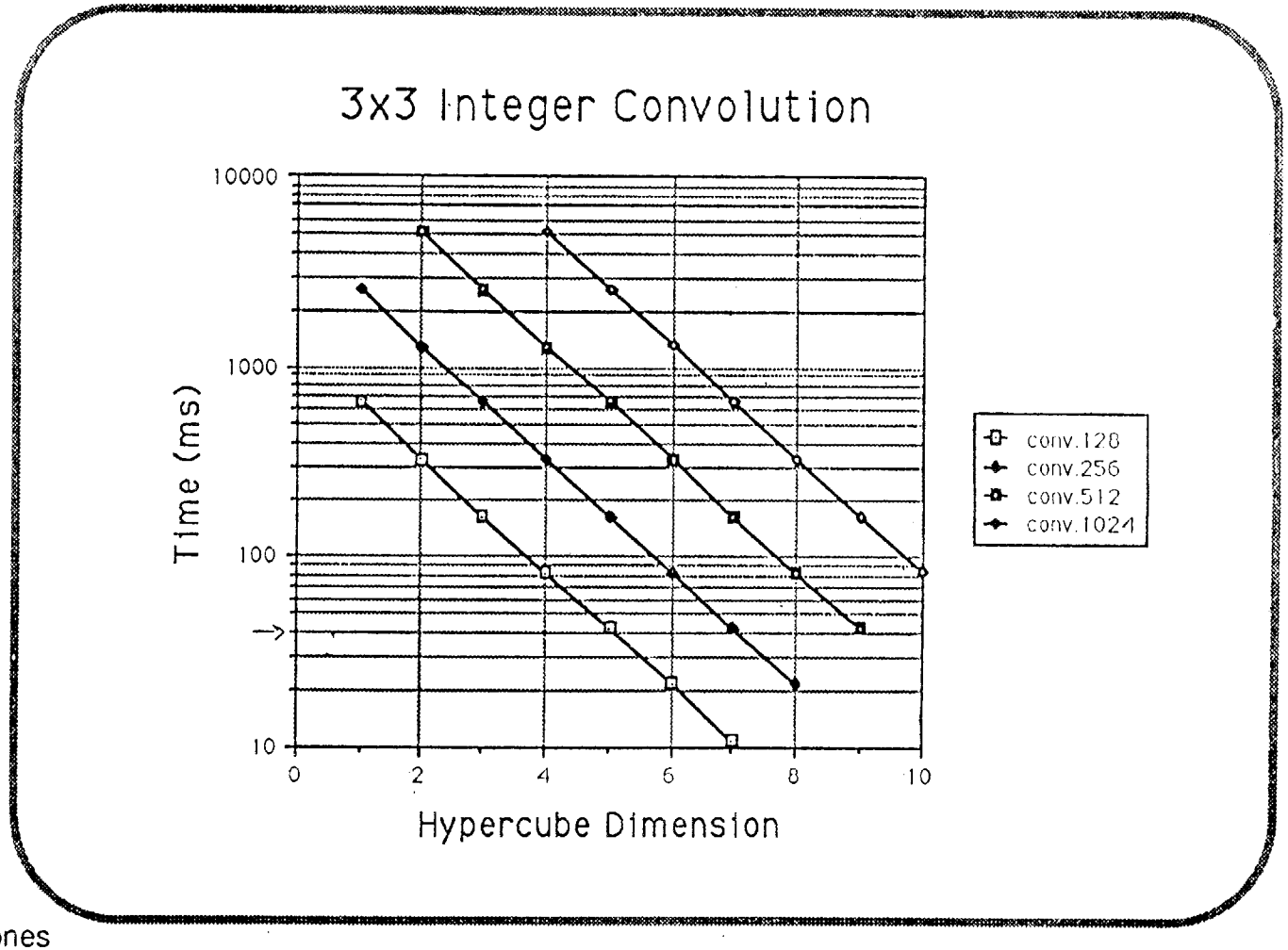
genex(struct IMAGE *pic , int rast)

(e.g.) genex (pic , 1);

Example: 3x3 Convolution

```
conv3( src, dst, n0,n1,n2,n3,n4,n5,n6,n7,n8 )
     struct IMAGE *src,*dst;
     int n0,n1,n2,n3,n4,n5,n6,n7,n8;
{
     unsigned char **src_pic,**dst_pic;
     int conv[9];
     register int sum,i,j,*ptr;

/* exchange */                                  /* dereference */

     genex( src , 1 );                          src_ pic    = src->p;
                                                dst_pic     = dst->p;
/* load convolution buffer */

     ptr = conv;      *ptr++ = n0;     *ptr++ = n1;     *ptr++ = n2;
                      *ptr++ = n3;     *ptr++ = n4;     *ptr++ = n5;
                      *ptr++ = n6;     *ptr++ = n7;     *ptr++ = n8;
/* do it */

     for( i=0; i<src->nrows; i++ ){
     for( j=0; j<src->ncols;j++){
          ptr = conv;
          sum    = *ptr++ * src_pic[i-1 ][j-1 ];
          sum += *ptr++ * src_pic[i-1 ][j    ];
          sum += *ptr++ * src_pic[i-1 ][j+1 ];
          sum += *ptr++ * src_pic[i    ][j    ];
          sum += *ptr++ * src_pic[i    ][j+1 ];
          sum += *ptr++ * src_pic[i+1 ][j-1 ];
          sum += *ptr++ * src_pic[i+1 ][j    ];
          sum += *ptr++ * src_pic[i+1 ][j+1 ];
          dst_pic[i][j] = sum;
     }
     }
}
```

Image Processing on a 1024 node Multicomputer



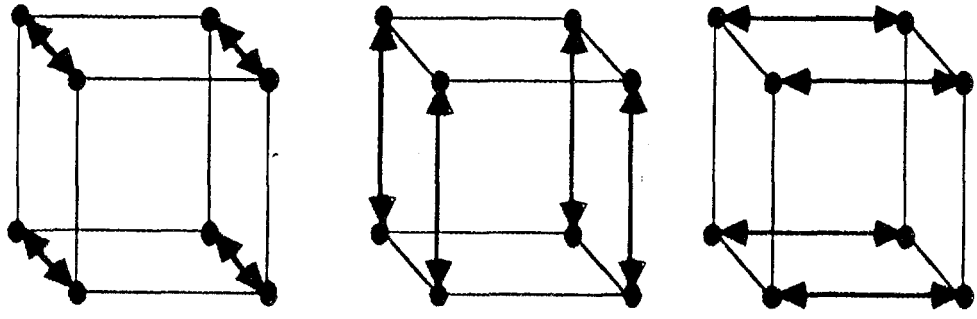
J.P.Jones
Advanced Computing and Integrated Sensor Systems Group
Oak Ridge National Laboratory

Low Level Benchmarks

(in milliseconds)

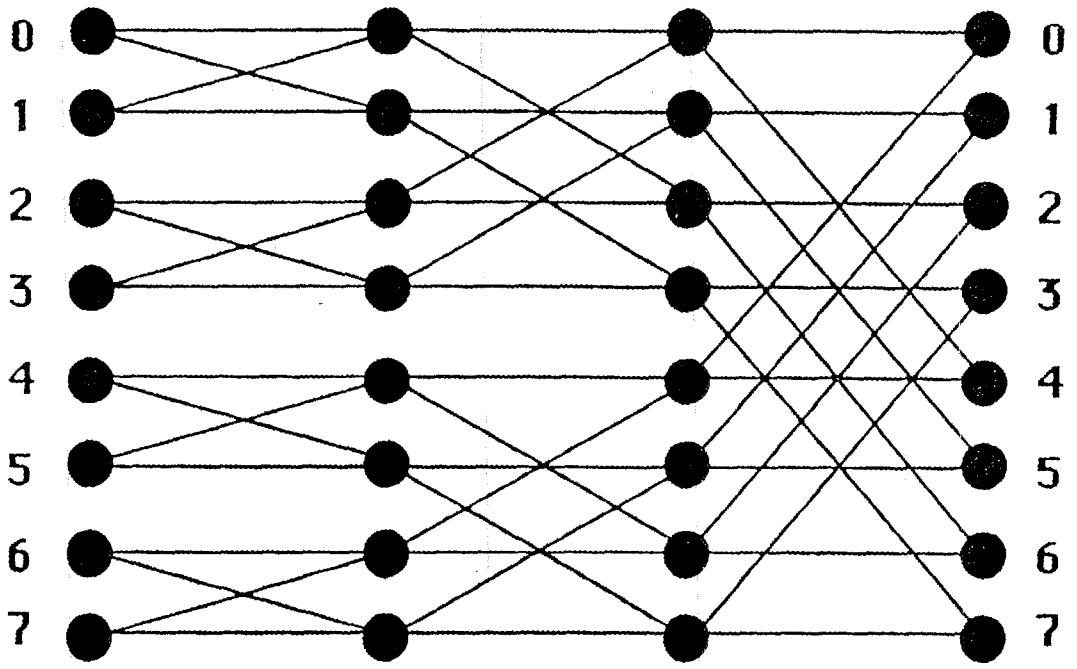
	Hypercube Dimension				
	2	3	4	5	6
Binary Threshold	128	64	32	16	8
Global Average	102	52	28	18	10
Global Histogram	133	75	49	37	33
3x3 Integer Conv.	1477	740	371	187	95
Sobel	2315	1160	581	292	147
3x3 gray. max/min	1989	996	500	252	127
3x3 gray. open/close	3978	1992	1000	503	255
3x3 dilate/erode	1027	515	259	130	67

The Butterfly Accumulator



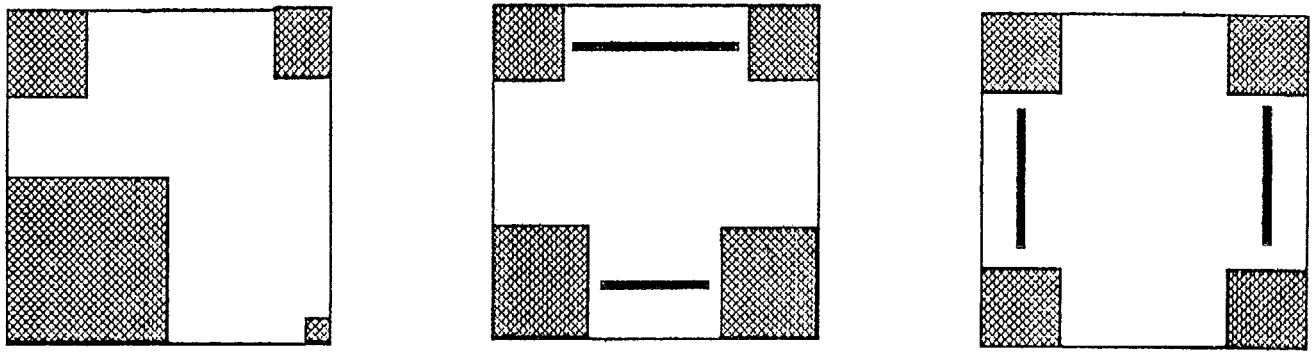
```
for( axis=0; axis<dim; axis++ )
    neighbor = node^(1<<axis)
    nwrite(my_buff,len,neighbor)
    nread(his_buff,len,neighbor)
    resolve buffers
```

Butterfly Network



Communication graph of "Butterfly accumulator" is isomorphic.

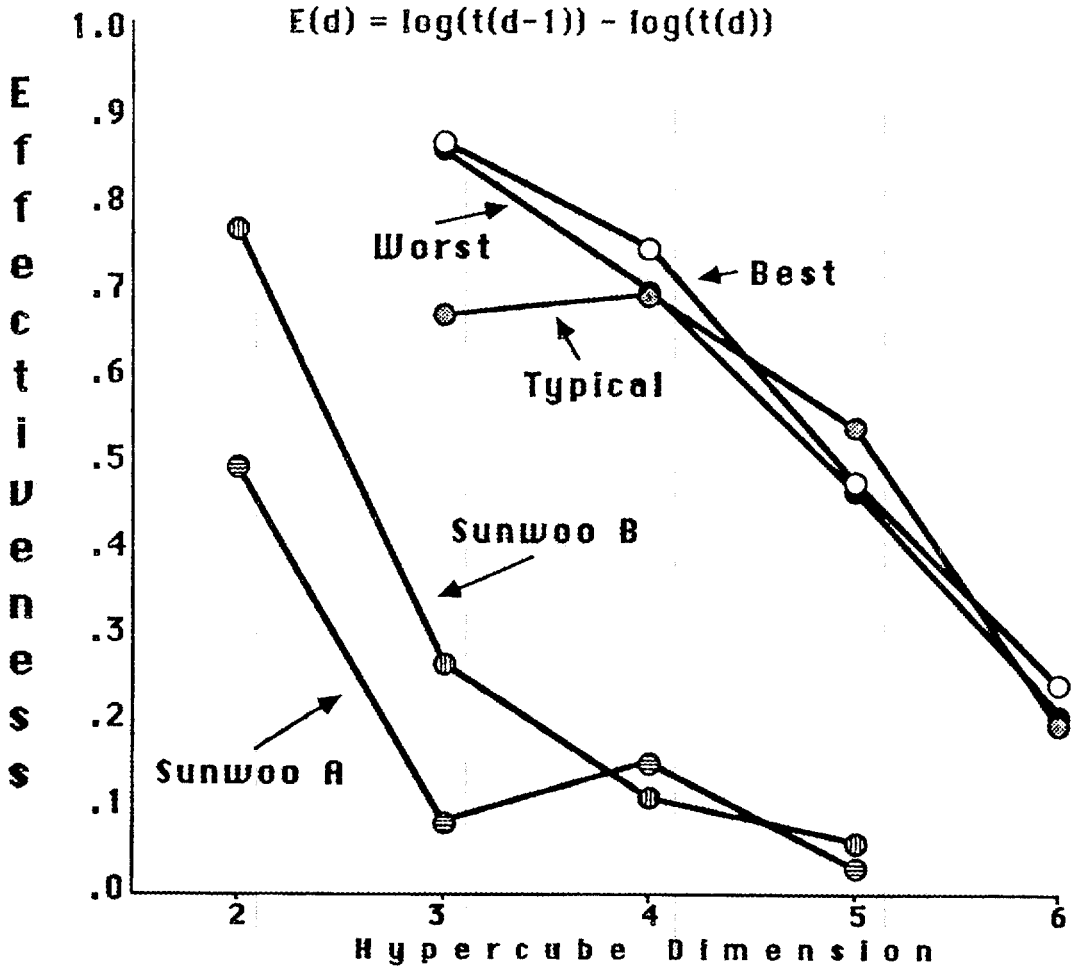
Load Balancing Example



Initial Condition | Iteration 1 | Iteration 2

Residual imbalance = Hamming distance

Present Algorithm Vs. Sunwoo, et. al. Effectiveness



	Hypercube Dimension				
	2	3	4	5	6
Worst Case		.87	.71	.47	.21
Typical		.68	.70	.55	.20
Best Case		.88	.76	.49	.25
Sunwoo A	.50	.09	.15	.03	
Sunwoo B	.78	.26	.11	.07	

USAGE

Robot Vision
- Model-based navigation
- Object recognition
- Model-based manipulation

Image Reconstruction
- Stochastic relaxation
- Mean-Field annealing

Motion Detection & Prediction

Multi-resolution methods

Concurrent Algorithms

Range image segmentation

**Neural Networks

Related Work

Groom, S.L., Maser, A.S., & Lee, M. (1988) "Design and implementation of a concurrent image processing workstation based on the Mark III hypercube." Proc. 3rd Conference on Hypercube Concurrent Computers and Applications, in press.

Hummel, R.A. (1986) "Connected component labeling in image processing with MIMD architectures." In: Intermediate Level Image Processing, M.J.B. Duff, ed. Academic Press, Bonas, France.

Jones, J.P. (1988) "A concurrent on-board vision system for a mobile robot." Proc. 3rd Conference on Hypercube Concurrent Computers and Applications, in press.

Jones, J.P., & Mann, R.C. (1988) Concurrent algorithms for a mobilr robot vision system." Proc. SPIE (937) Applications of Artificial Intelligence VI, M. Trivedi, ed., pp. 495-504.

Lee, S.Y., & Aggarwal, J.K. (1987) "Exploitation of image parallelism via the hypercube." In: Hypercube Multiprocessors 1987, M.T. Heath, ed. SIAM, Philadelphia.

Lee, S.Y. & Aggarwal, J.K. (1987) "Image processing on multiprocessor systems." Proc. 1987 IEEE Conference on Systems, Man, and Cybernetics.

Miller, R., & Stout, Q.F. (1987) "Some graph- and image-processing algorithms for the hypercube." In: Hypercube Multiprocessors 1987, M.T. Heath, Ed. SIAM, Philadelphia.

Miller, R., & Miller, S.E. (1988) "Image processing on hypercube multiprocessors." Proc. SPIE (939) Hybrid Image and Signal Processing.

Mudge, T.N., & Abdel-Rahman, T.S. (1987) "Vision algorithms for hypercube machines." J. Parallel and Distributed Computing (4) 79-94.

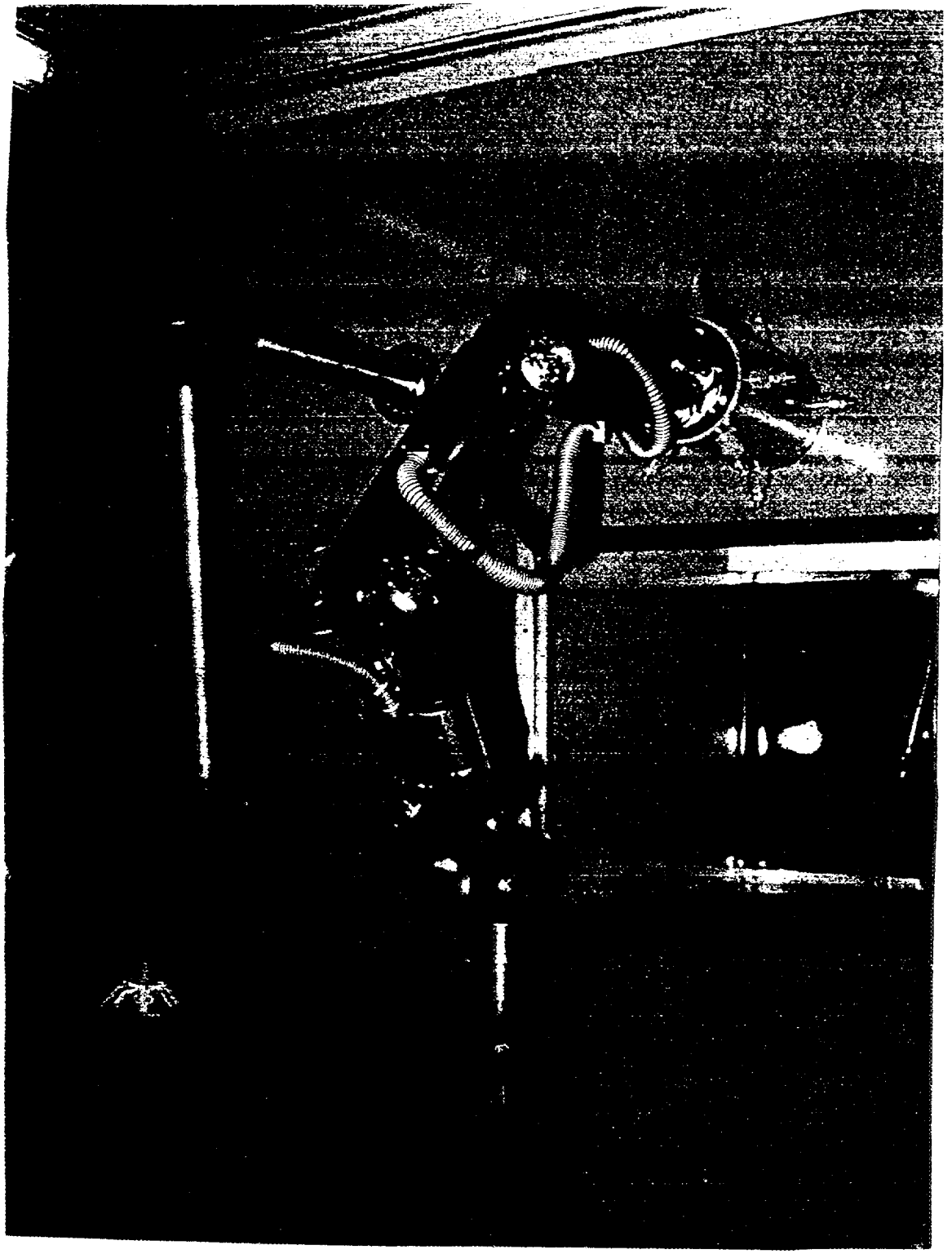
Sunwoo, M.H., Baroody, B.S., & Aggarwal, J.K. (1987) A parallel algorithm for region labeling." Proc. IEEE Workshop on Computer Architecture for Pattern Analysis and Machine Intelligence.

Stout, Q.F. (1986) "Hypercubes and pyramids." In: Pyramidal Systems for Image Processing and Computer Vision, V. Cantoni & S. Levialdi, ed.s NATO ASI Series ARW, Springer-Verlag, New York.

DEMT/SYST/MRS

FIRST EXPERIMENT WITH A 3D SENSOR

G. ERMONT D. GALLEY



DEMT/SYST/MRS

FIRST EXPERIMENT WITH A 3D SENSOR

Operator help during the phase of trajectory learning

GOALS :

- **By matching 3D information from the camera and geometrical shapes, using a friendly Man Machine Interface, compute the trajectory points and send them for execution by the robot controller.**

PROBLEMS :

- **Uncertainty on data from the 3D camera**
- **Artefacts**

SOLUTION :

- **Use a proximetry sensor for real time correction**
- **Software processing**
- **Mix sensor and operator information**

DEMT/SYST/MRS

FIRST EXPERIMENT WITH A 3D SENSOR

3D Camera from SAGEM / CEA/D.LETI

SENSOR CARACTERISTICS :

. Conical sweeping using a He-Ne Laser

. Acquisition by Position Sensitive Detector

. 256 * 256 measurements in 800 ms

CONTROL SYSTEM

. VME bus

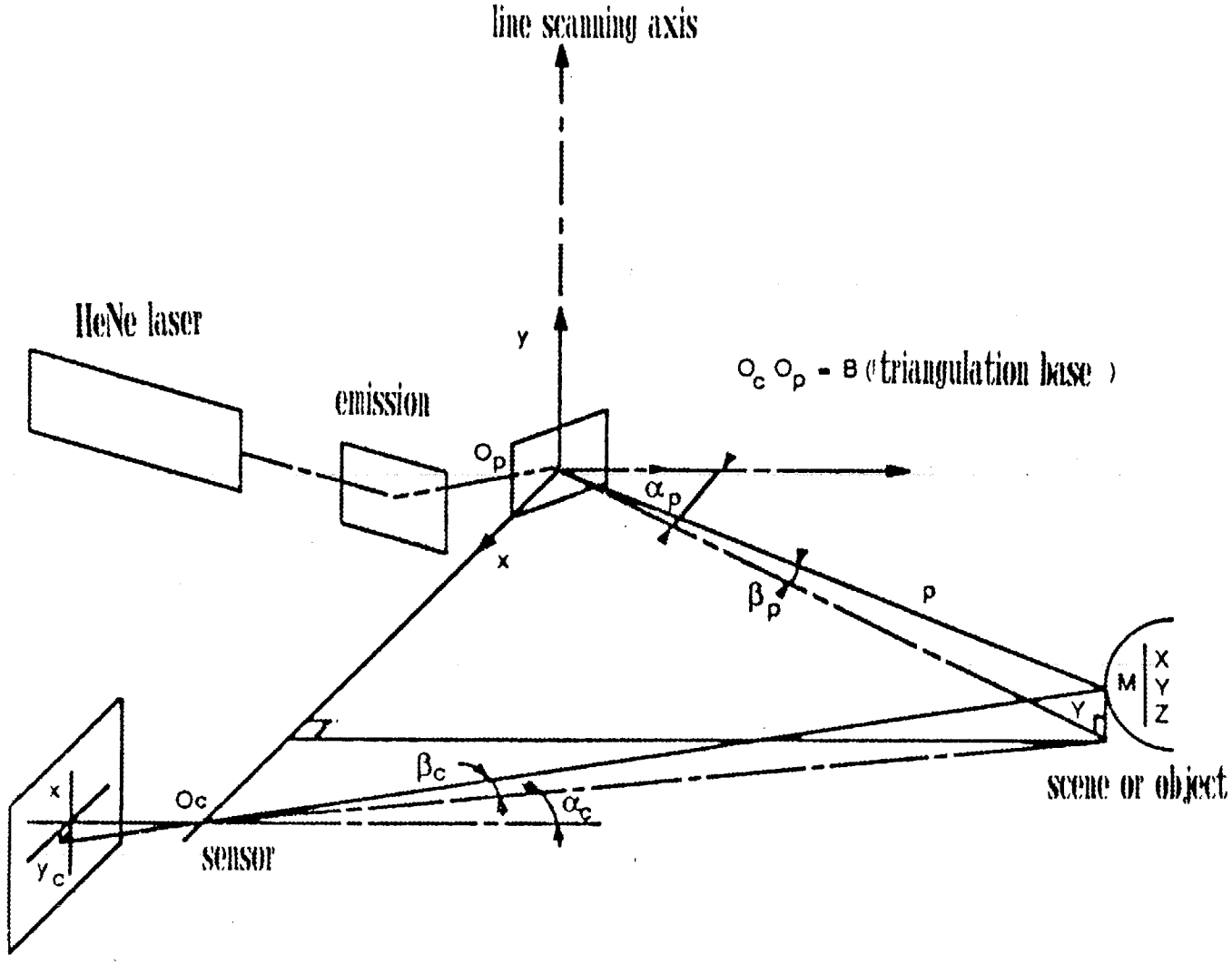
. Specialised control process card

. 1 Megabyte RAM for aquisition

. Triangulation co-processor

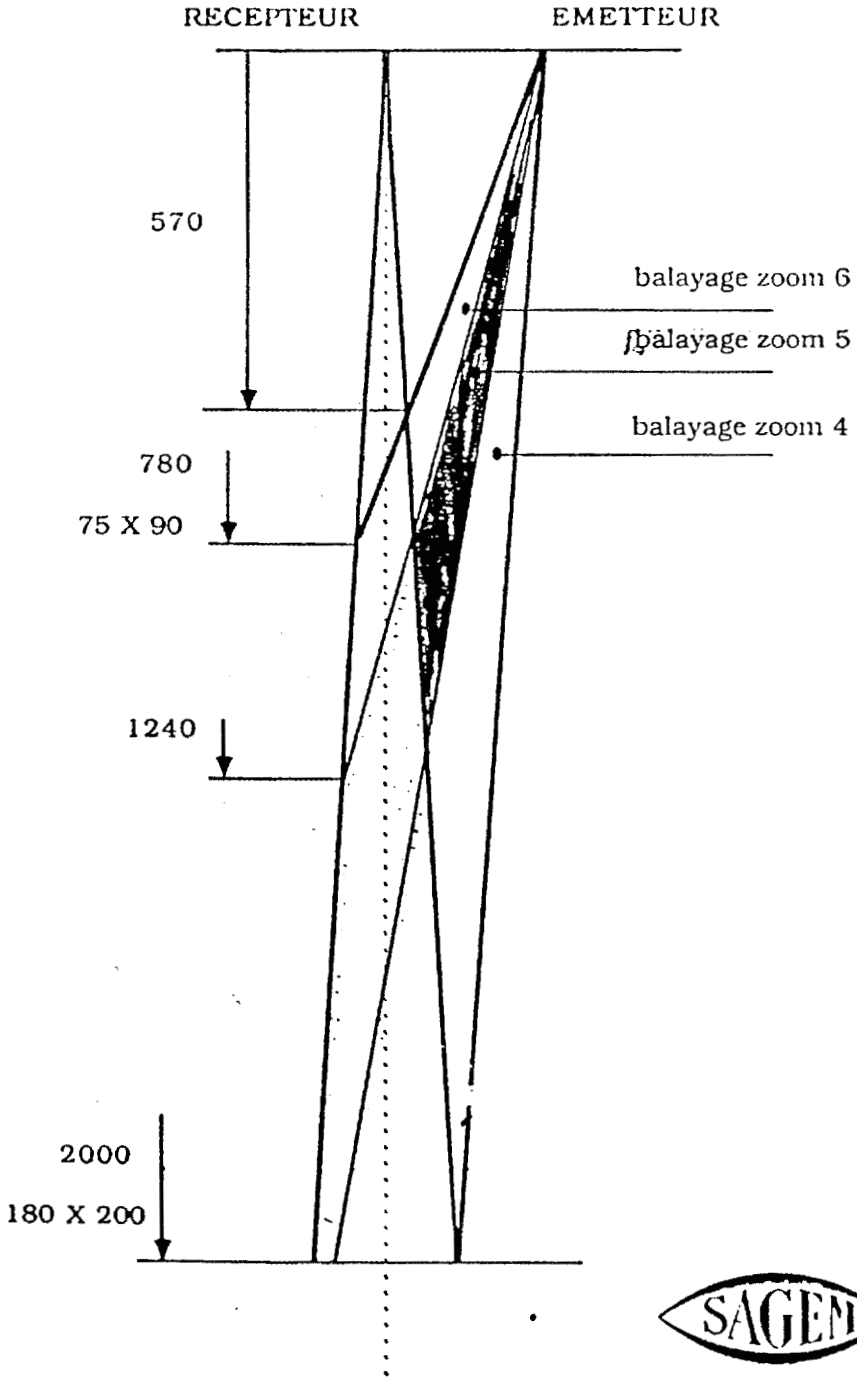
. 68010 processor

.IEEE communication card with PC

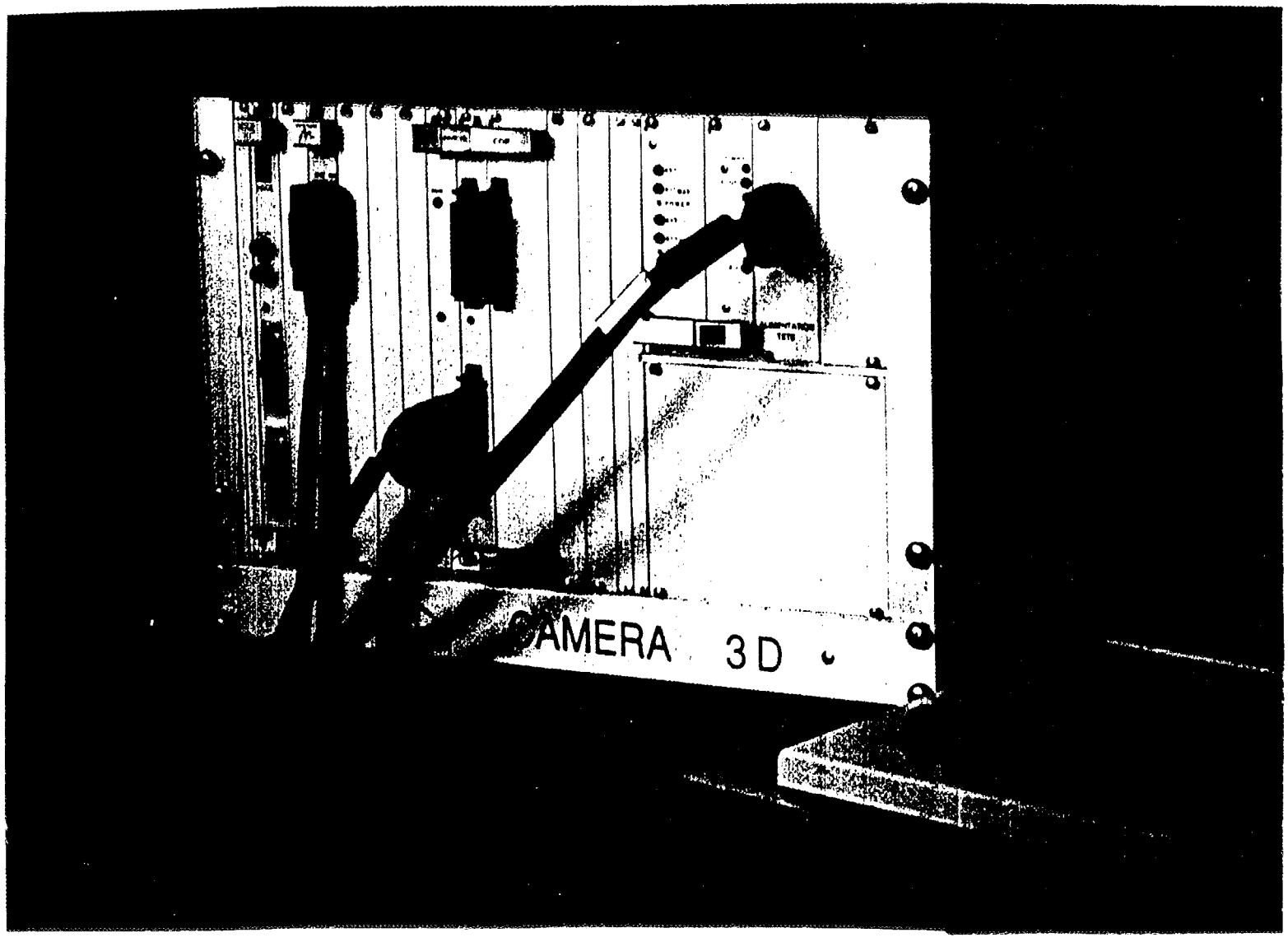


PRINCIPLE 3D CAMERA

DOMAINE DE MESURE CAMERA 3D VERSION 3D-2000



SAGEM


AMERA 3D

DEMT/SYST/MRS

FIRST EXPERIMENT WITH A 3D SENSOR

3D Camera from SAGEM / CEA/D.LETI

DATA TRANSMITED TO THE PC

. Intensity, X, Y, Z

. on 16 bits for 256*256 points

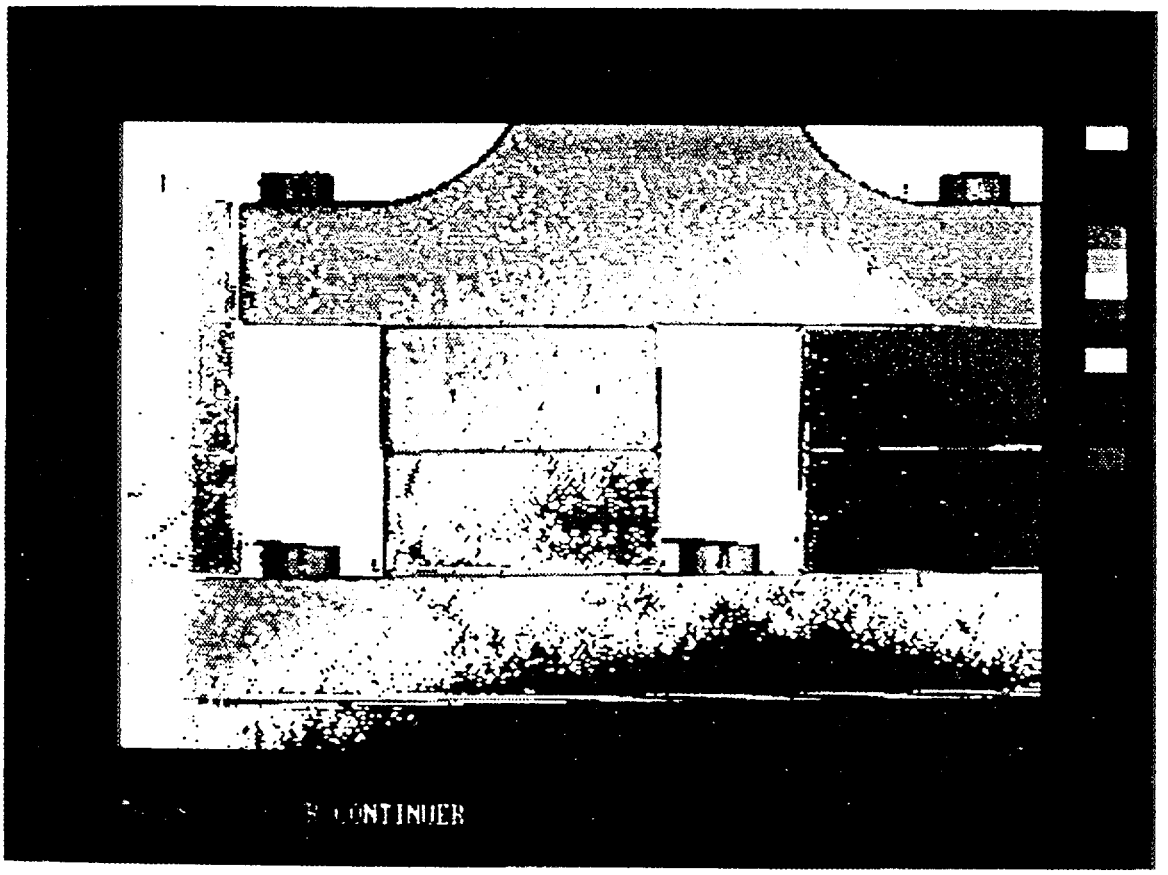
PROCESSING

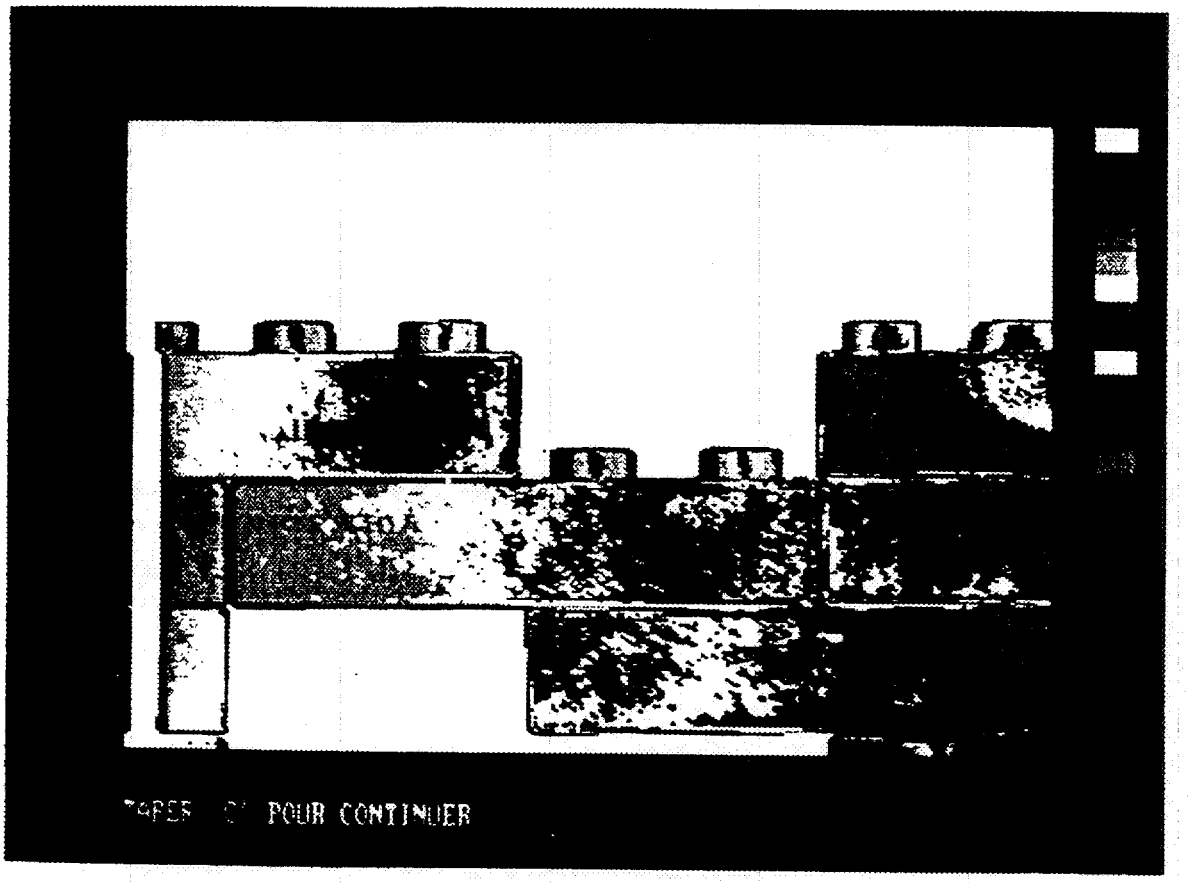
. Reconstruction of "depth image"

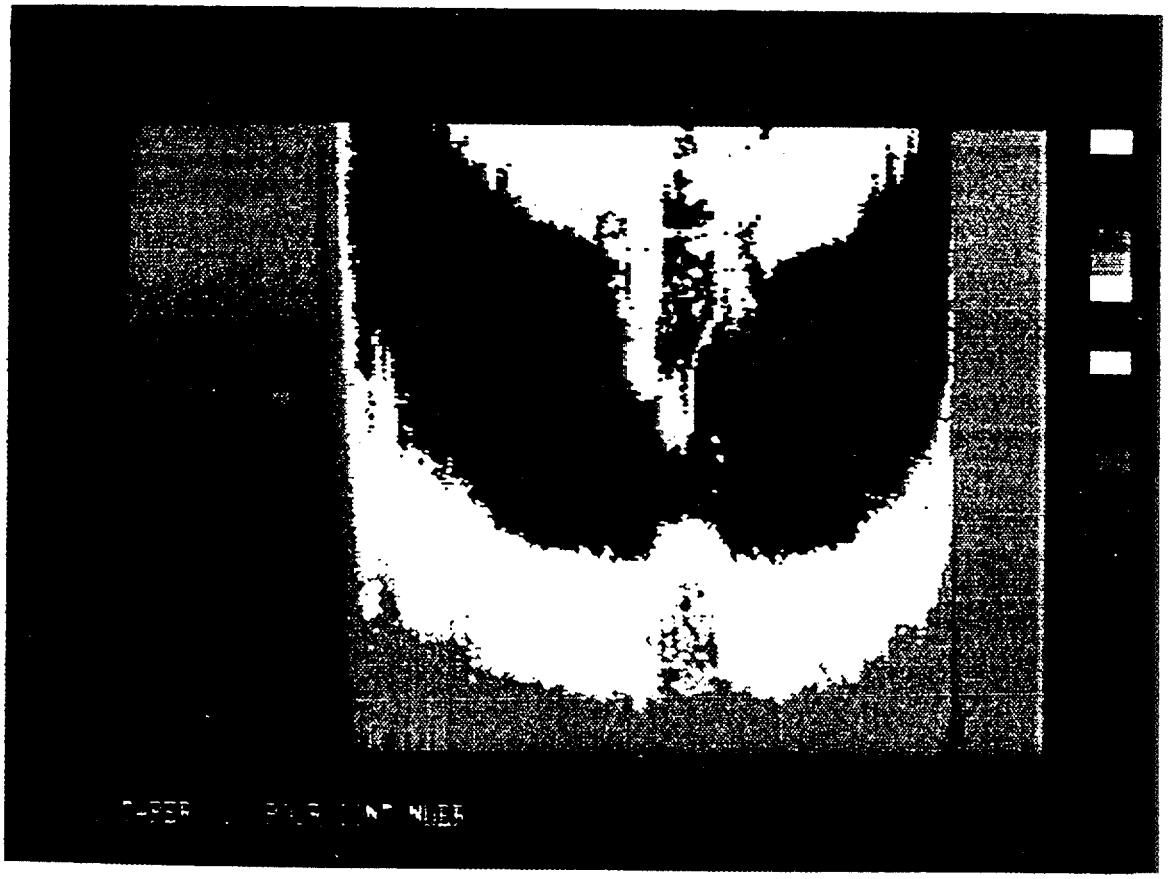
. Use of Intensity to eliminate points with large uncertainties

. Zooming

. Return of X Y Z from mouse pointer


CONTINUER


POUR CONTINUER

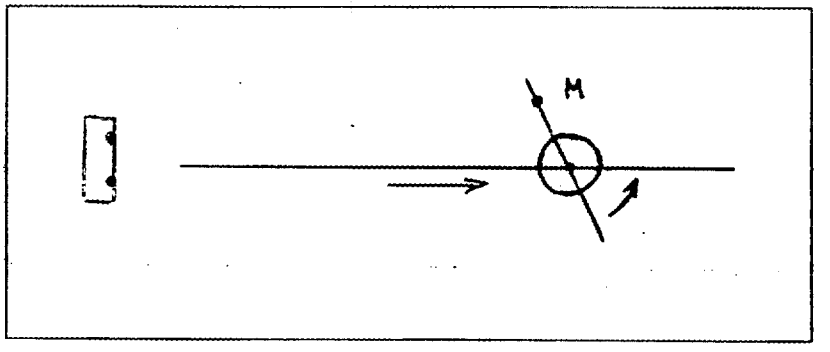


DEMT/SYST/MRS

FIRST EXPERIMENT WITH A 3D SENSOR

Measurements of Depth uncertainty

. Vertical plane , homogeneous reflectance :



Precision on absolute X Y Z better than 1 mm in all range

. Complex objects with holes (LEGO)

Problems occur as soon as :

- large heterogeneity of reflexion
- multiple reflexions

Machine Vision Research Using a Laser Range Camera

Frank Sweeney

Research Plan

1. Determine device characteristics (calibration)

   - limits of operation (range and reflectance)
   - resolution
   - noise characteristics
   - image distortions

2. Model the imaging process (camera model)

   - geometric image distortion

3. Develop low level processing techniques

   - filtering
   - image restoration
   - edge detection
   - segmentation

4. Image representation

   - superquadrics
   - splines
   - polyhedral surfaces

5. Iterate between 2, 3 and 4

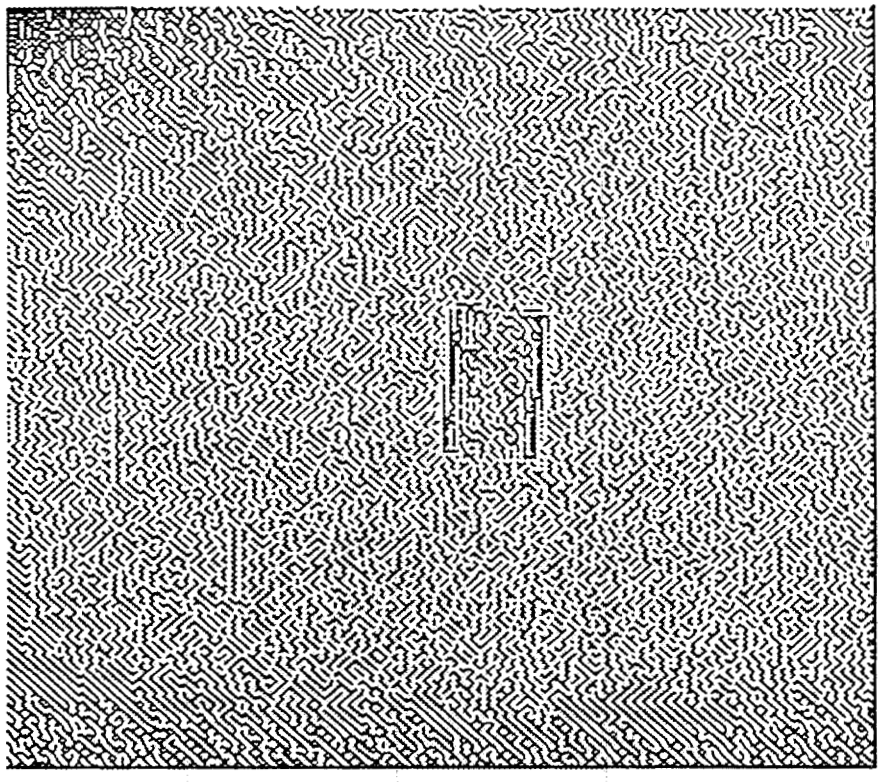
Odetics Laser Range Camera

Specifications:

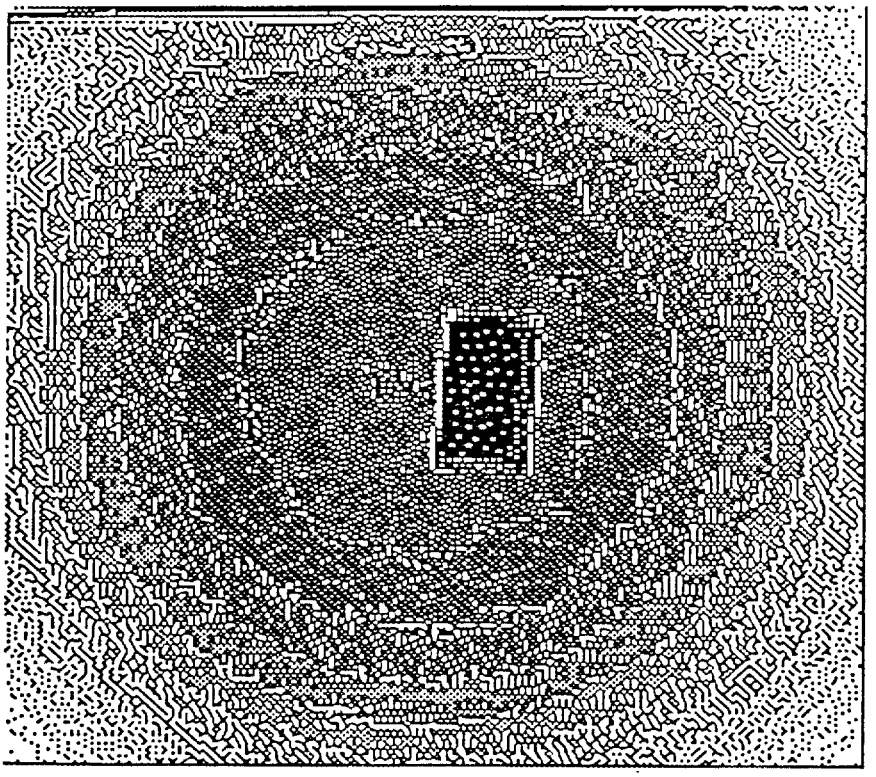
image size	128 x 128 pixels
data format	8 or 9 bits range 8 or 7 bits reflectance
ranging method	phase shift
frame rate	~ 1 sec.
laser device	CW diode
laser wavelength	820 nm
laser power	50 mW
modulation freq.	16 MHz square wave
Head unit size	23 x 23 x 24 cm
head unit weight	1.4 Kg
power supply weight	12.7 Kg
power requirements	50 watts

Measured Performance

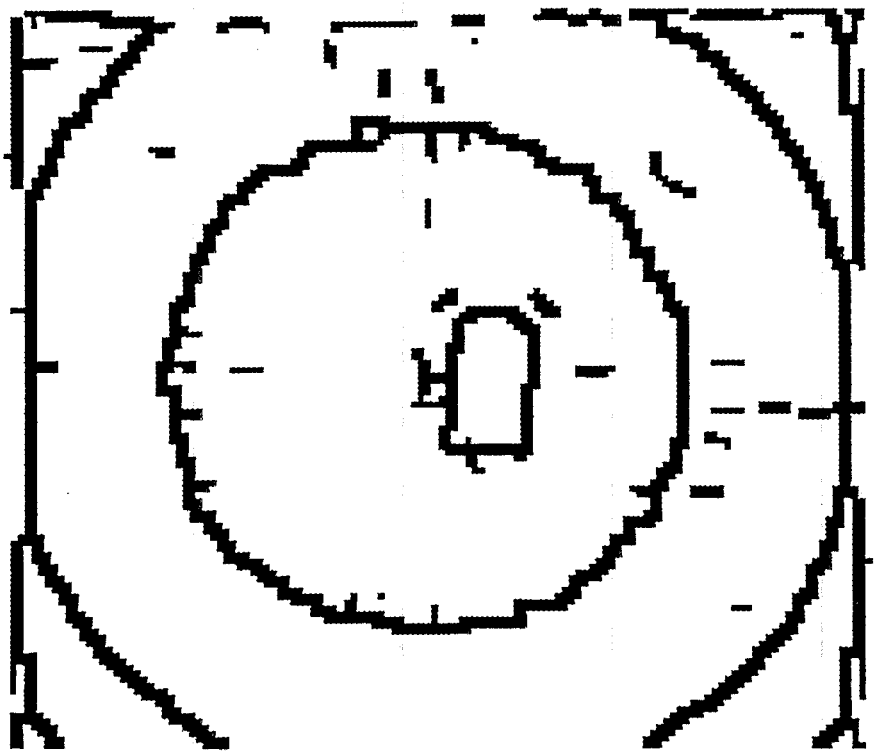
	Manufacturer	Measured
range resolution	3.66 cm	3.92 cm
angular resolution		
vertical	0.5 deg	0.51 deg
horizontal	0.5 deg	0.45 deg
field of view		
vertical	60 deg	64.8 deg
horizontal	60 deg	57.6 deg
reflectance limits of operation	—	>2% diffuse
image center	—	-3 deg vertical



Reflectivity Image



Normalized
Range
Image



Edgemap

Application Problems

- camera model (geometric distortion)
- range dependance on reflectance at transitions
- invalid ranges for low reflectances
- low reflectances caused by
  - -- highly absorptive surfaces (matte black)
  - -- specular reflections
  - -- high angle surface normals
- noise
- image dynamic range
- edge detection and image segmentation dependance on low level processing

Conclusions

- the laser range camera is a unique device
- it does <u>not</u> automatically solve problems normally encountered in machine vision
- it presents challenging new problems in low level image processing, scene representation, and sensor fusion

Session III: Environmental Modeling

Sensor-Based Mapping

M. Beckerman
Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory

CESAR/CEA Workshop on Autonomous Mobile Robots
Oak Ridge, Tennessee
May 30 - June 1, 1989

Sensor-Based Mapping

- Data structure for combining sensor information from different measurements
- Can do geometric reasoning about static features in the environment
- Useful for identifying and reducing systematic errors
- Interesting class of world models

Publications

- "Treatment of systematic errors in the processing of wide angle sonar sensor data for robotic navigation," M. Beckerman and E. M. Oblow, IEEE Transactions on Robotics and Automation
- "Spatial reasoning in the treatment of systematic sensor errors," M. Beckerman, J. P. Jones, R. C. Mann, L. A. Farkas and S. E. Johnston, Advances in Intelligent Robotic Systems, Cambridge, 1988
- "World modelling and multisensor integration for a mobile robot," M. Beckerman, L. A. Farkas, J. P. Jones, R. C. Mann and C. W. Glover, Third Topical Meeting on Robotics and Remote Systems, Charleston, 1989

Systematic Errors
Ultrasound

- Dependences:
  (i) Radiated power and sensing thresholds
  (ii) Beam width (resolution)
  (iii) Object properties
  and environmental geometry

- Error classes:
  (i) Distortions in size, orientation
  and location of object surfaces
  (ii) Specular reflections and complete
  absorption

Multi-Sensor Integration

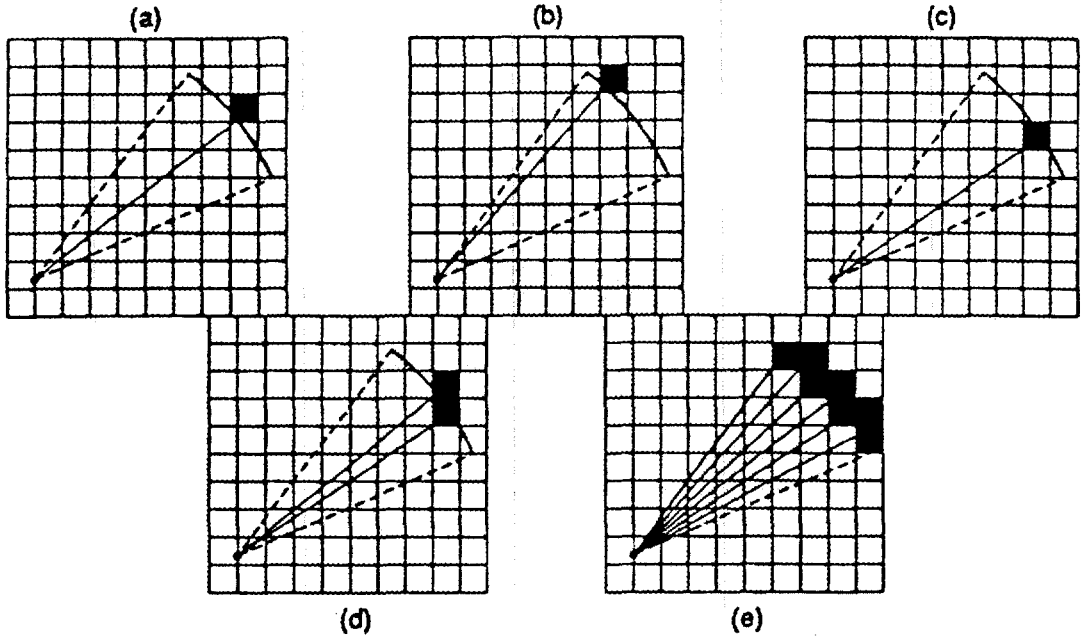
- Complementary sensor domain
- More than one sensing position
- Interdict at low-level
- Use range information from sonar domain in vision data processing
- use vision edge information to refine sonar world model

Spatial Reasoning
Fusion of Ultrasonic Sensor Data

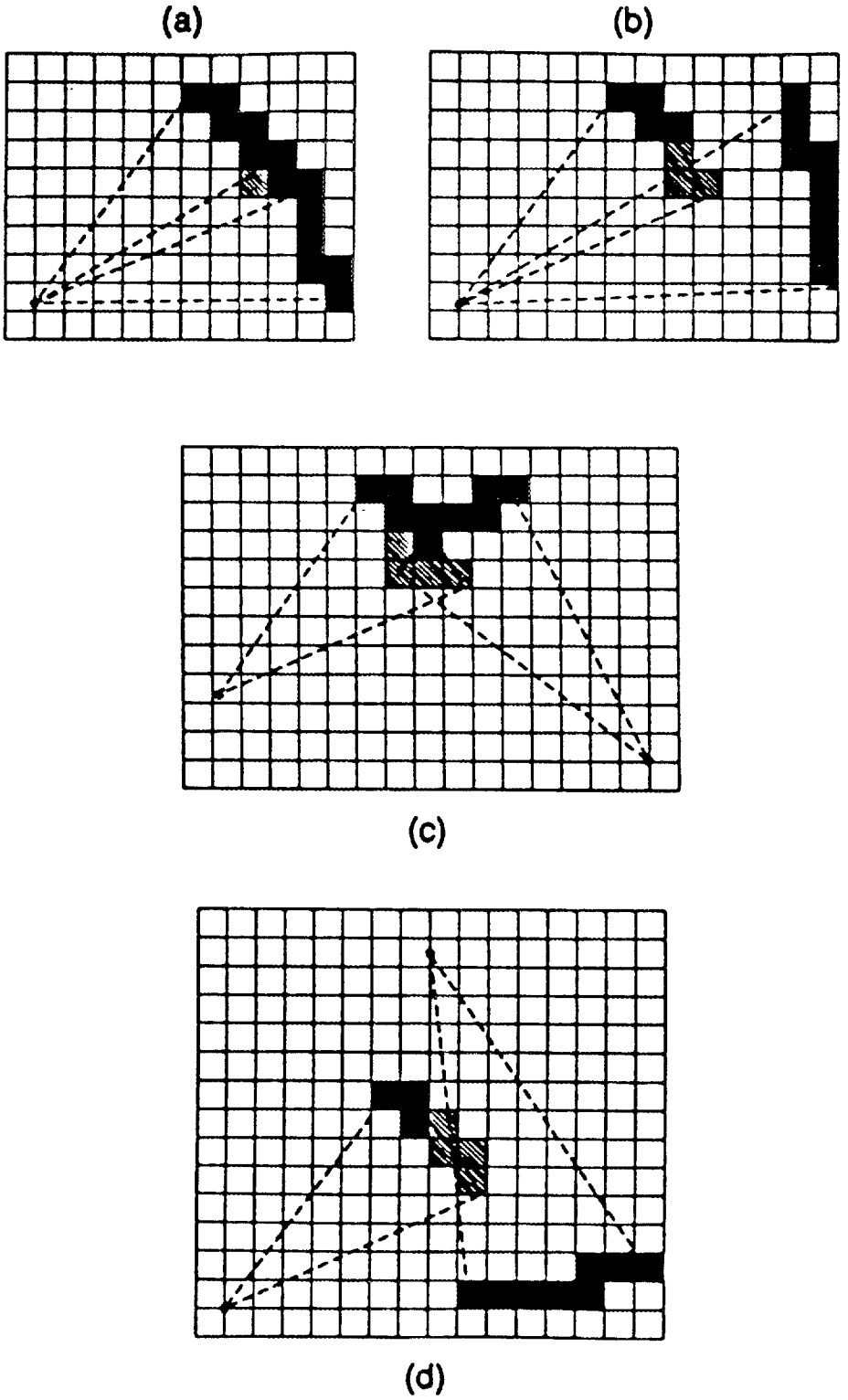
- A real sonar beam cannot pass through a real object
- There are no point scattering sources

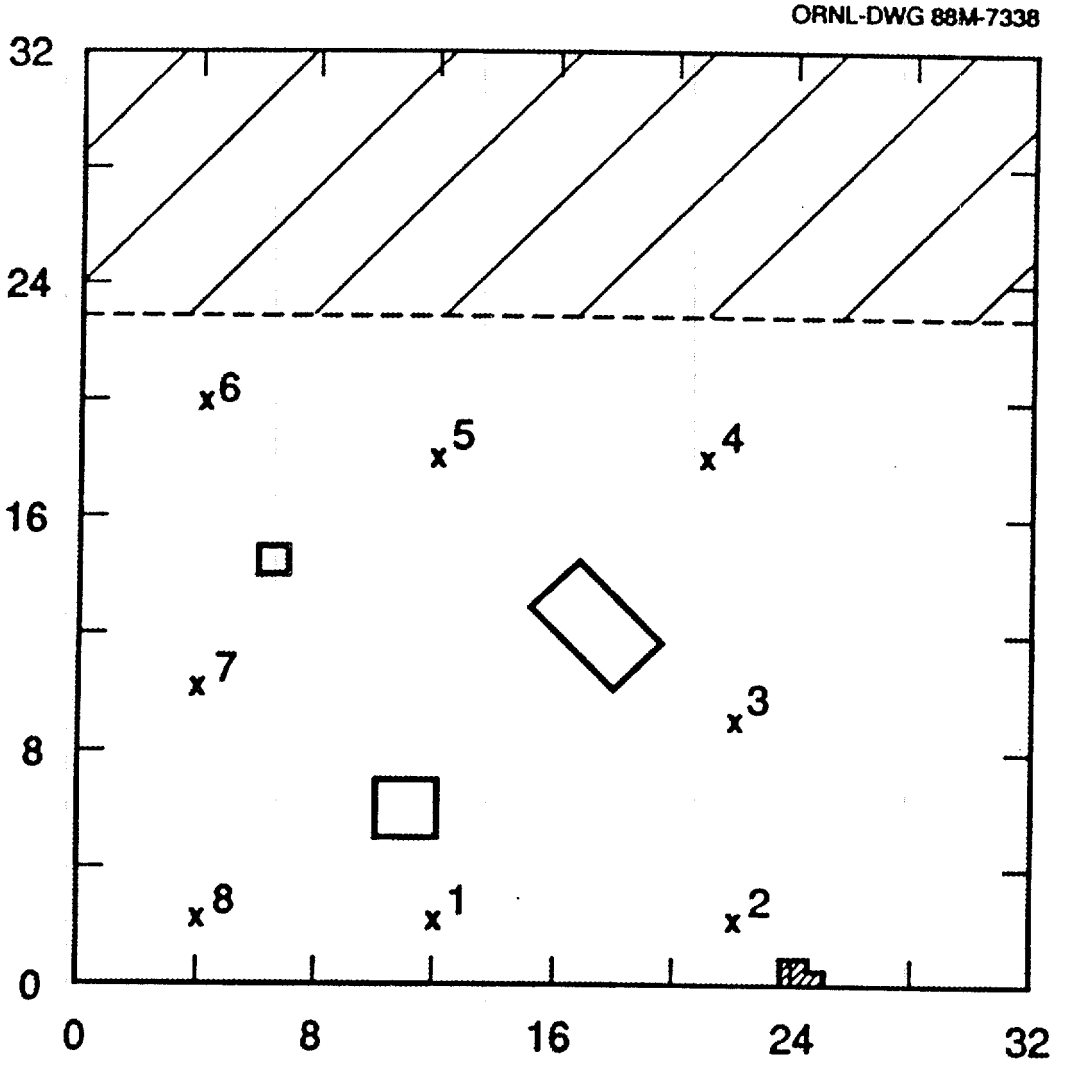
- Pattern analysis
- Consistent-labelling

ORNL-DWG 88M-7341



ORNL-DWG 88M-7340C

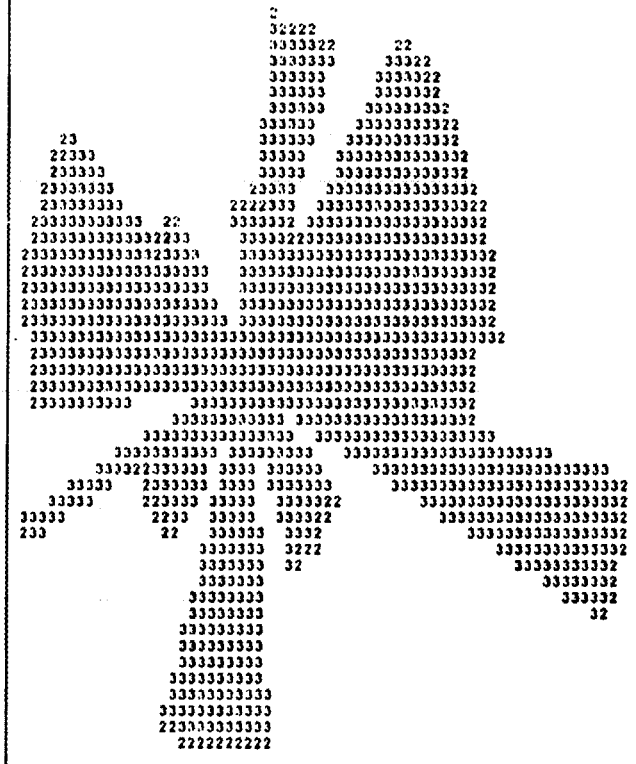



ORNL-DWG 88M-7338
32
24
16
8
0
0
8
16
24
32
x6
x5
x4
x7
x3
x8
x1
x2

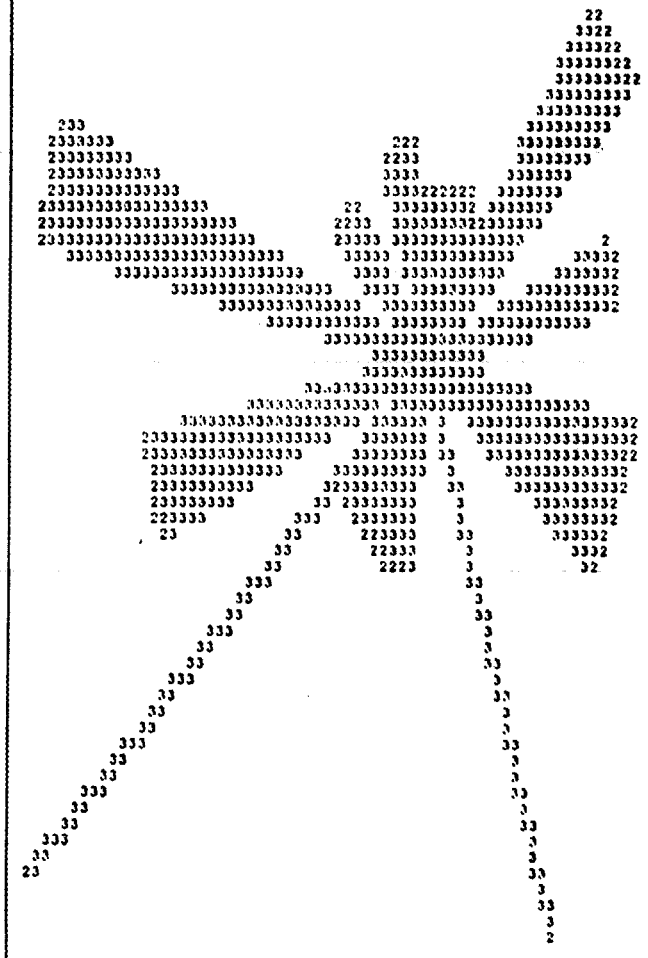
									10	10
		62	62					10	10	
29	29		62	62	54		15	10		
		29	29	62	62	29	29	29	41	41	41
				54				15			
				62				10	15		
				62	62			15	15		
				5	62	5	5	15	10		
								15			
								10	15		
									15	15	
										15	
											15

88-15186

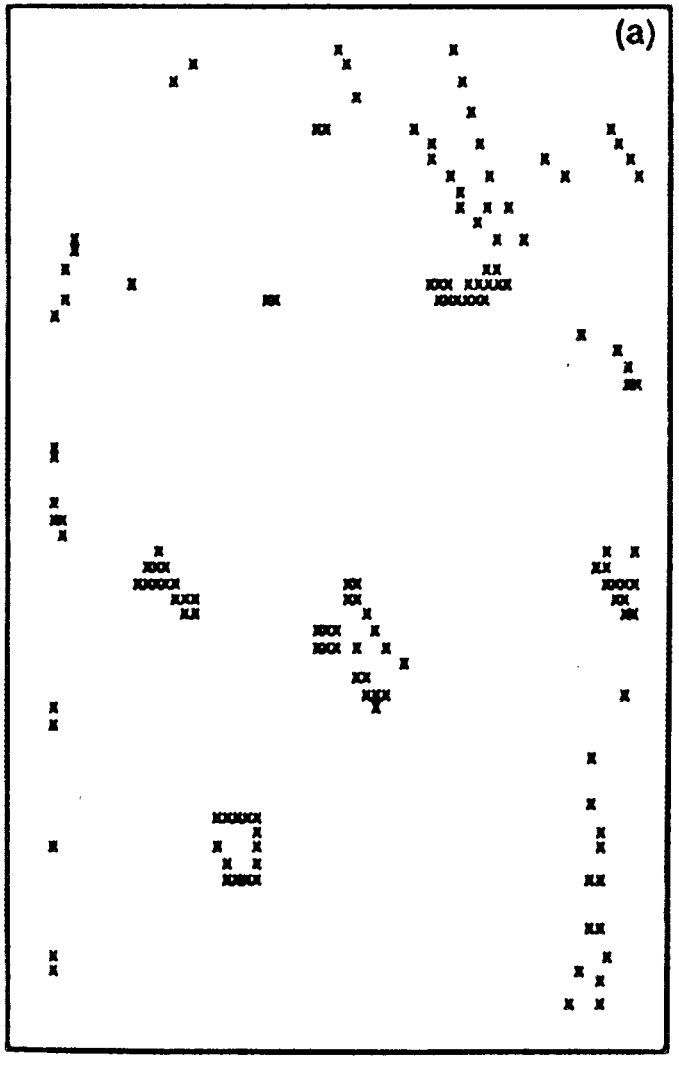
(a)

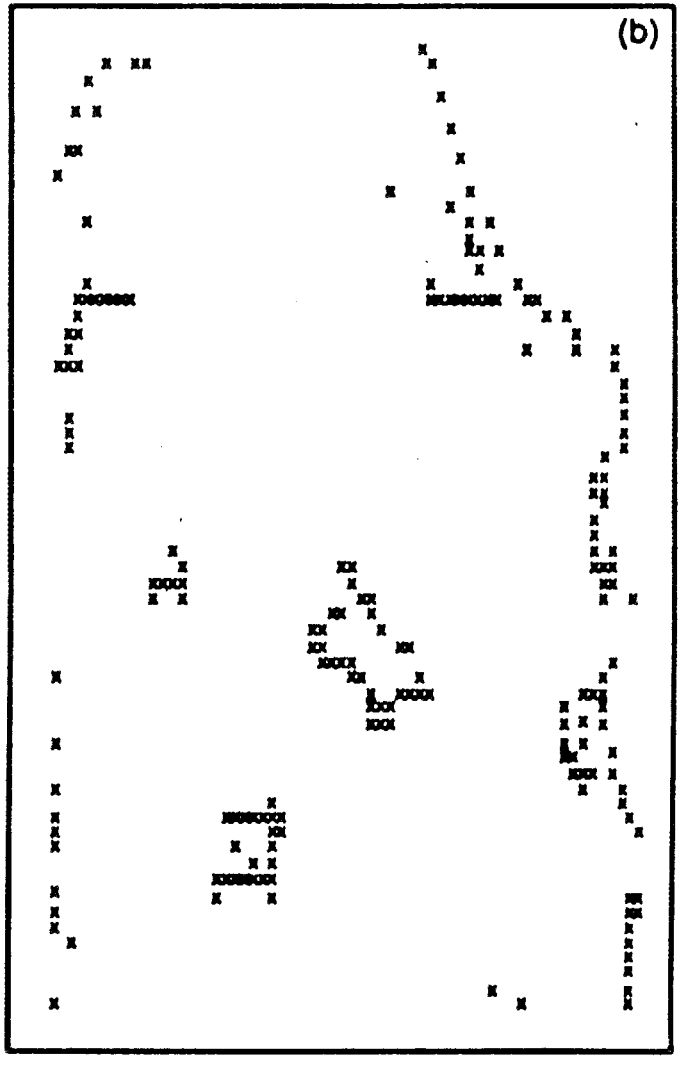


(b)



ORNL-DWG 88M-7347



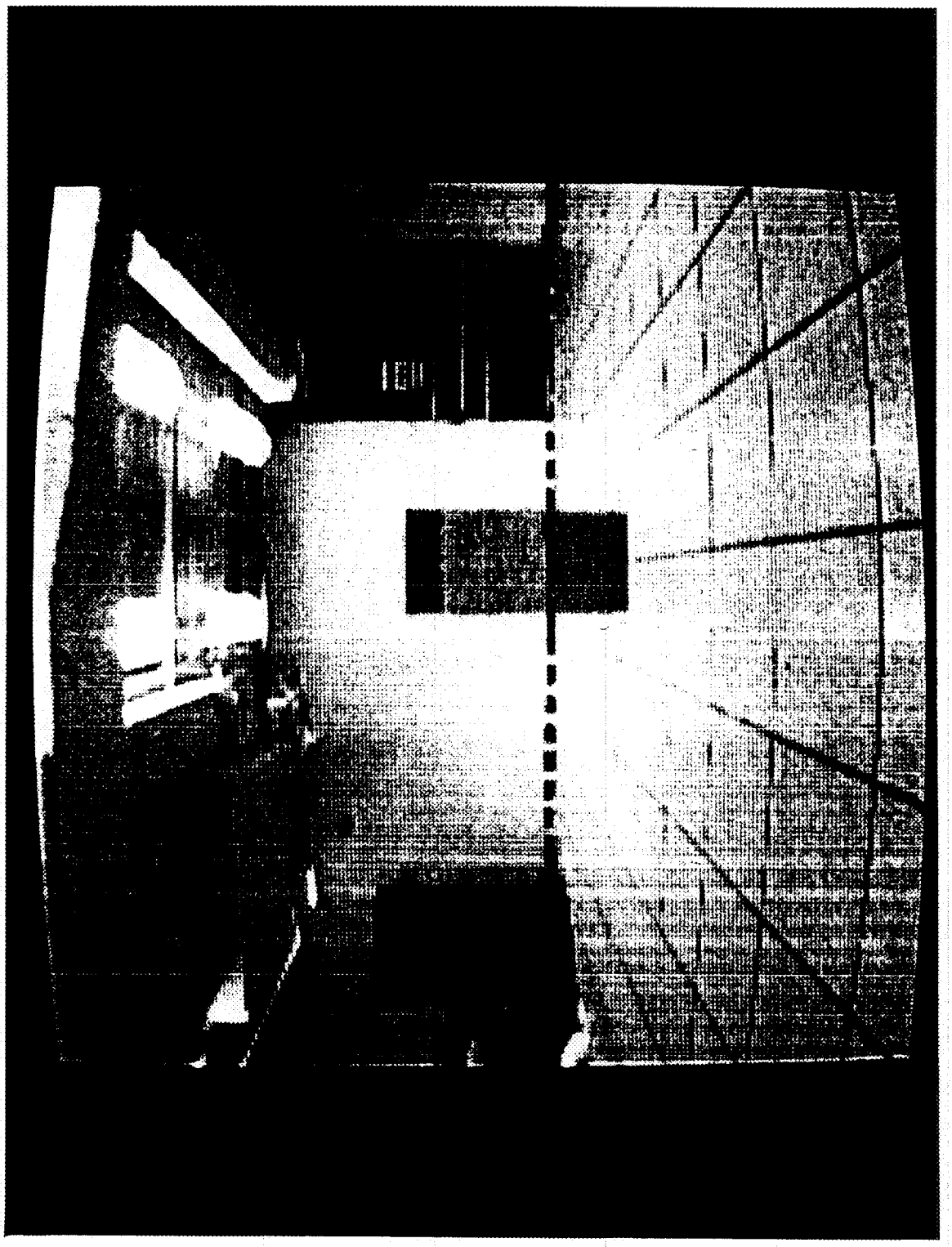


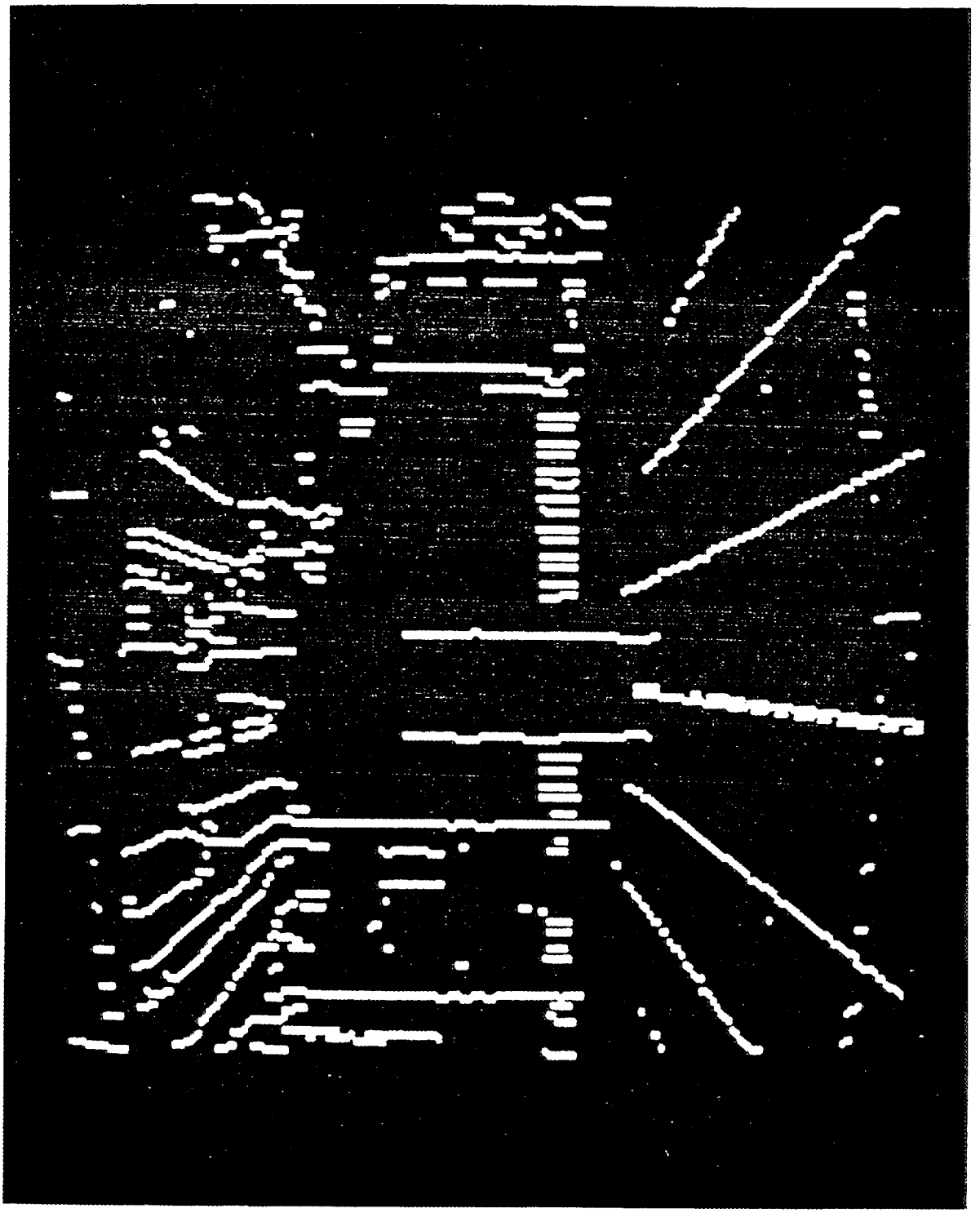
Systematic Errors

- Errors in interpretation arise due to incomplete information
- sonar scan
  - distortions in size and orientation
  - false echoes
- visual scene
  - unable to distinguish between depth discontinuities and (i) variations in local intensity (ii) background details

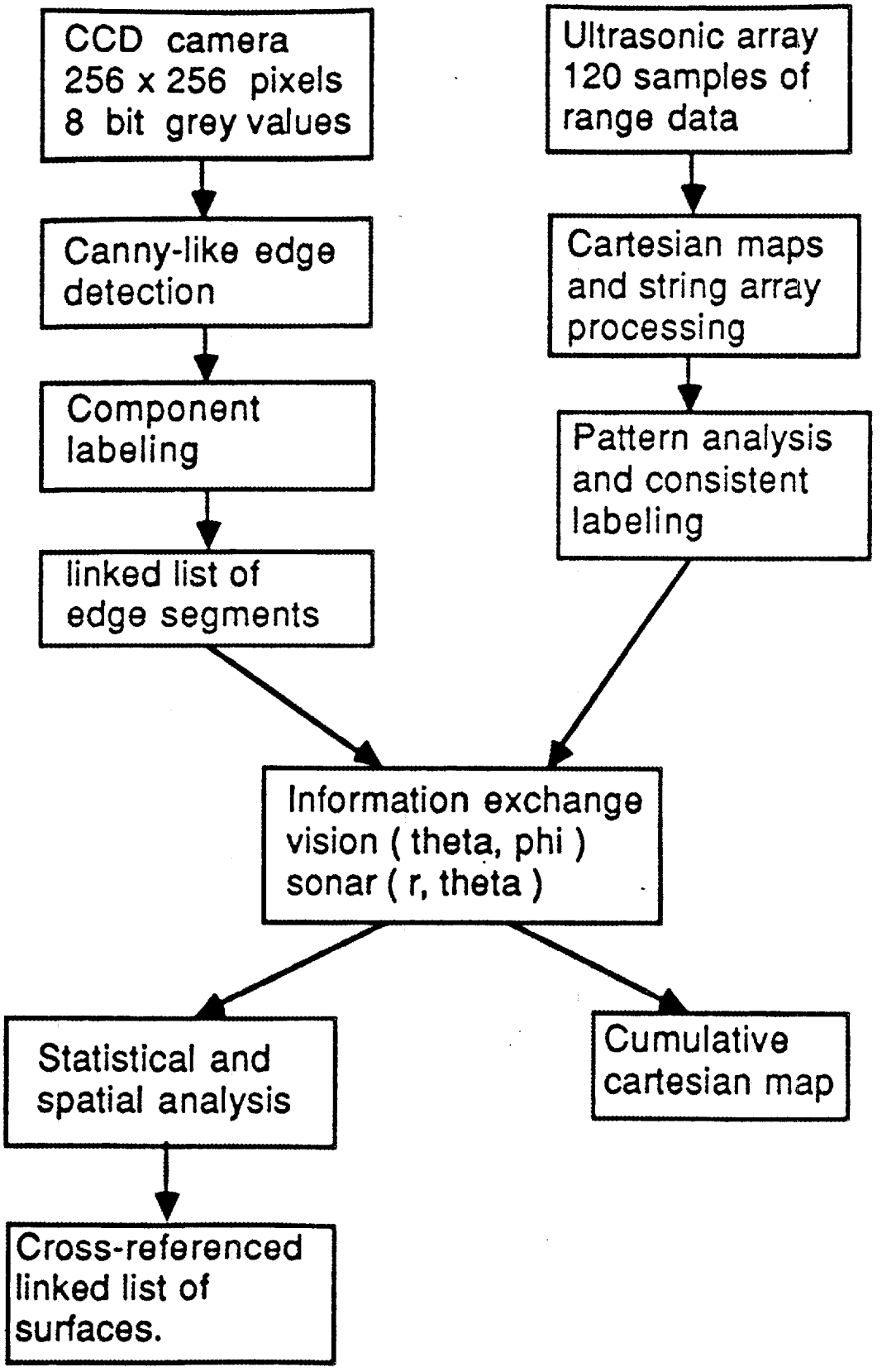
Sensor Fusion
correct erroneous interpretations

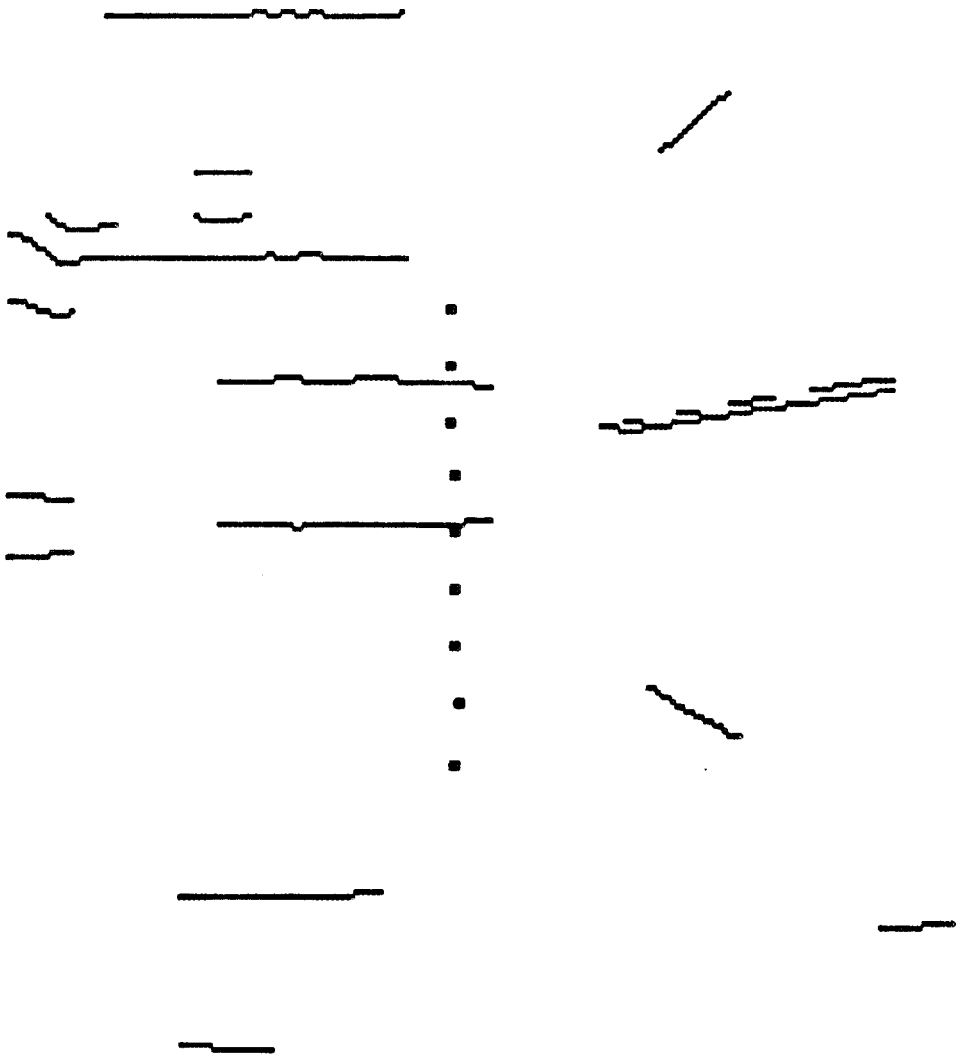
- ultrasonic sensor domain
  - -examine results from more than one scan
  - -use physical (spatial) reasoning
- visual sensor domain
  - -establish correspondence between sonar strings and visual edges;
  - -use sonar range information, spatial reasoning
- sonar range information is more reliable than precise size, orientation of surfaces; return edge information to sonar domain





4325-97


CCD camera
256 x 256 pixels
8 bit grey values
Canny-like edge detection
Component labeling
linked list of edge segments
Ultrasonic array
120 samples of range data
Cartesian maps and string array processing
Pattern analysis and consistent labeling
Information exchange
vision (theta, phi)
sonar (r, theta)
Statistical and spatial analysis
Cumulative cartesian map
Cross-referenced linked list of surfaces.



Conclusions

- We have developed an MSI strategy for the reduction of systematic errors
- exchanged information both within and across sensor domains to achieve internal consistency; used spatial reasoning
- used multiple world models -
  -2D maps in the sonar domain
  -cross referenced, linked lists of geometric features in the 2D vision domain

Conclusions (cont'd)

- mobile robots encounter broad dynamic ranges of sensing conditions
- must evaluate methodology:
  - examine how dense the scenes can be to establish breakdown point
  - examine how low can one set thresholds
  - examine for near and far distances

Environment Modeling for Robotics Simulation

G. Dejonghe, A. Cossic, D. Chaigne
Centre d'Etudes Nucléaires de Saclay
Département des Etudes Mécaniques et Thermiques
91191 Gif sur Yvette Cedex, France

Abstract - *Geometric modeling is a key issue for simulation systems applied to robotics. The different needs for environment modeling are of four kinds: solid modeling, graphic representation for animation and visualization, physics modeling - mechanical, photometric properties for instance, i.e physical attributes -, and last, link modeling for kinematic and dynamic evolution of the world. The basic approach choosen for CAD simulation is the following one: (1) a full constructive solid geometry tree for solid modeling, (2) a derived boundary polyhedral representation for the graphic model, (3) physical properties such as volume, center of mass, matrices of inertia obtained by the polyhedral model and physical attributes for the physical model, and (4) a general graph of the world whose nodes are the solid components and edges joints of any kind between components. Such an environment is intended to provide a geometric, graphic and physical representation for any arbitrary complex kinematic structure. CSG tree and polyhedral representation may also be considered for environment reconstruction from sensor data, in order to build an internal model useful for decision making, geometric reasoning, object and pattern recognition.*

I. Introduction

The various needs for environment modeling applied to robotics simulation are of four kinds: solid modeling, graphical representation, physics modeling and world modeling.

Constructive Solid Geometry (CSG) is becoming an essential feature of almost every solid modeling systems. Complex solid objects are built from boolean operations - intersection, union and difference -, among primitive objects such as box, cylinder, sphere or torus. The resultant objects are reused as building blocks for more complex solids. Objects need not to be convex and may contain holes.

A CSG object is represented as a tree whose leaves are primitive solids and node CSG operations. Such a representation, though not providing a direct description of the resultant object's boundary, is suitable for (1) powerful and natural description from programming, (2) complex solids building facility and (3) accurate intersection calculations between a half straight line and a solid.

On the other hand, an explicit boundary representation consisting of nonintersecting planar polygonal faces is useful for (1) 3D graphic visualization,

(2) physical properties computation (center of mass, matrices of inertia...), (3) topological properties determination, collision detection and drawing near distance calculation. The regularized boundary of an object can be obtained from its CSG tree.

A complex solid object is given a CSG attribute but it is not the only one. Physical attributes may be necessary for robotics simulation. We think for instance to mechanical and photometric properties: modulus of friction, modulus of elasticity, roughness, brightness and so on. All these attributes are needed for algorithm exploiting object movements, coherence, and recognition.

The environmental world may be seen as a collection of arbitrary complex solid objects, that is a generalized graph whose nodes are solid components and edges joints of any kind (from zero to six degrees of freedom) between solid components. In a first stage, we restrict our study to holonomic joints.

All the computer programs are written in Ada language. The graphic library used is the ANSI standardized PHIGS package. The programs run on Sun4 series graphic workstations.

II. CSG representation[1]

The CSG representation of a complex object has a tree structure where the leaf nodes are primitive objects and non-leaf nodes are primitive operations. The set of primitive objects commonly used are block, cylinder, sphere, cone and torus. The primitive operations are union, intersection and difference. With a sufficient set of primitive objects and the three primitive operations, arbitrary complex objects may be defined.

The description of CSG objects is made in textual quotation: the CSG tree input data are provided within an Ada program which is linked to utility packages such as Ares_Geometry package containing definitions and specifications of geometric utility routines and Ares_Csg package containing data structures implementation for CSG elements.

This manner to introduce input data is an easy and natural way to build CSG objects. The following program (Fig. 16) gives an idea of programming the construction of two models of tables: the first model is the one of a rectangular table, the second one leads to a model of a circular table. The function Table returns one of the two model types; the default one returned is the circular table model. Note that basic operator overloading (Ada facility) is used to make it easily understable: the or operator is used for union, the and operator is used for intersection and the "-" operator for difference. Position is given by applying the "*" overloaded operator. In the example, position is given by function T which returns a 4x4 homogeneous matrix, resultant matrix of a translation. Box and Cylinder functions are two utilities routines of Ares_Csg packages which build respectively a CSG element box (3 parameters are needed: length, width and height) and a CSG element cylinder (2 parameters are needed: radius and height). Fig. 17 shows the two CSG models in wire-frame representation.

The use of an Ada program for CSG object definition has three advantages: (1) no specific input data language is needed: the language used for data structure creation is the same as the language used for Ares developments (uniformity and integration enhancement), (2) all the software engineering features of Ada can be used: lisibility, reusability, abstraction (strong typing), generic units, packaging, and (3) we take advantage of all the benefits of types verification and checks performed by the compiler which guarantee the coherence and completness of all the structures.

III. Boundary polyhedral representation

III.a Polyhedral object data structure

The data structure of a polyhedral representation are organized into vertices, edges, contours, faces and solids. Hereafter follow a brief definition and description of the geometric features characterizing a polyhedral representation.

A vertex is a three coordinates point in 3D space with vertex tolerance for accuracy. Its data structure contains a list of all edges to which the vertex is connected.

An edge joints two vertices. Its data structure contains pointers to its starting and ending vertices and a list of all contours in which the edge is located and the corresponding location.

A contour is a single oriented closed planar polygonal curve. Its data structure contains pointers to the edges located within it and their directions (direct or reverse sense).

A face is a two-dimensional finite set of contours. Its structure is characterized by a list of contours, the normal and the distance from the origin and a face tolerance for accuracy. A contour whose direction is the opposite of the normal to the face is a hole.

A solid is a collection of boundaring faces, whose normals point away from its interior.

III.b Derived CSG polyhedral representation[2]

The boundaries of polyhedral objects are partitionned into nonintersecting parts. The algorithm bases its operation (intersection, union or difference) on removing intersections from pairs of faces which can have any arbitrary number of contours of arbitrary complexity.

Enclosing bounding boxes are used to determine whether two objects interfere with each other. If so, each face of each object is checked against one from the other. This process is applied to a collection of objects and is highly recursive. At the lowest level of the recursion, intersection from a particular pair of faces is processed. This intersection may be empty, may lead to collinear line segments for transversal faces (see Fig. 18) or to coplanar polygonal regions for coplanar faces.

All intersection points of each face's edge with the other's plane are combined and sorted into an endpoint list. Membership information is gathered during face cutting and used when CSG operations are applied. The three distinct boolean operations can be performed simultaneously.

Vertex tolerance is supplied to determine whether two vertices must be merged. Face tolerance is also used to determine whether a vertex lies within the face or to one of its edges.

III.c Partitioning

The intersection algorithm modifies the contours of two transversal faces so that they no longer intersect. Traversal of the first (main) face with the transversal face's plane is processed. The two faces have they role interchanged and the process is done again.

The segment information obtained is sorted to find the cutting intervals (cuts, notches or slits).

An intersection point is called an endpoint if an edge of the first face (deemed the main face) penetrates the transversal face. An endpoint *e* is said to be an entry or an exit point, depending on the sign of:

$$\text{sign}\ ((u \times n_m).e)$$

where n_m is the normal to the main face, $u = n_m \times n_t$ (nt is the normal to the transversal face). If this sign is +1, the endpoint is an entry point (comes to inside), otherwise it is an exit (comes to outside). Note that $u \times n_m$ is coordinate invariant.

Face membership information of the endpoints is used to determine the

nature of the cutting intervals. The starting endpoint of a cutting interval is an entry, the ending one an exit.

A face is cut if endpoints of the face are present at both the start and the end of the cutting interval.

A slit is created in a face when both start and end of the cutting interval belong to transversal face.

A notch is placed in a face if one of the start or end of the cutting interval belongs to the face while the other point belongs to the transversal face.

III.d Solid modeling

From segmentation information, contours and edges are split, merged or even created. Then contours are labelized and classified by propagation as inside or outside others solids.

Contours are kept or removed, depending on the CSG operation and their normal orientations.

To produce the resultant object, contours and faces must be merged if necessary. All the linked lists and data structures are updated to obtain a new coherent solid object.

III.e Results

Fig. 1 to 5 show union, intersection and difference of two cones.

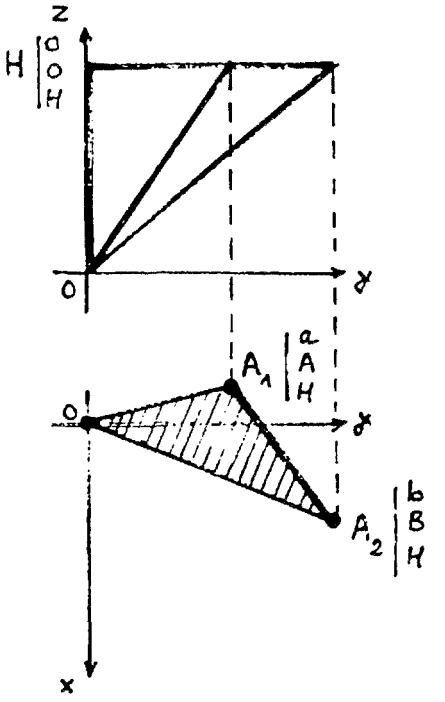
Fig. 6 to 10, CSG operations are applied on a cone and a box.

Fig. 11 to 15, CSG operations are applied on a torus and a cylinder.

IV. Derived properties

The boundary polyhedral representation is convenient for computing the physical properties of an object such as center of mass, solid angle, matrices of inertia.

Let us take the example of computation of moments of inertia. Considering a single face in a reference coordinate system:



The different moments of the tetraedron are given by:

$$I_{x^2+y^2} = H.X.(a^2+ab+b^2+A^2+AB+B^2)$$

$$I_{x^2+z^2} = H.X.(a^2+ab+b^2+6H^2)$$

$$I_{y^2+z^2} = H.X.(A^2+AB+B^2+6H^2)$$

$$I_{xy} = H.X.(2aA+aB+bA+2bB)/2$$

$$I_{xz} = H^2.X.(a+b).2$$

$$I_{yz} = H^2.X.(A+B).2$$

where: $X = (aB-bA)/60$

For each face, the contribution within a given coordinate system is calculated then summed to obtain the moments of inertia. Appropriate changes of coordinate systems are needed and Koenig's theorem must be taken into account.

The accuracy of the result depends on the discretization of the solid object.

The first level of discretization for a sphere is the icosaedron. The second level leads to 80 faces. The third one to 320 and so on. The following table sums up results obtained from computation of moments of inertia of a sphere (diagonal moments):

Level	Accuracy
1 (20 faces)	60%
2 (80 faces)	20%
3 (320 faces)	6%
4 (1280 faces)	1.5%
5 (5120 faces)	0.3%

The error at the level 3 is less than 6% and around 1.5% at level 4 (1280 faces). The level 3 is sufficient to obtain a good approximation of the moments of inertia.

V. World modeling[3]

The environmental model consists of a general graph containing solid components (nodes), refering to a solid CSG boundary representation model, and joints (edges) linking components to each other.

Any kind of joint is intended to be described, from zero (fixed joint)to six degrees of freedom (free joint). An object may have multiple joints. Nature and type of joints are taken into account, that is for instance if it is a rigid joint (fixed one), a non-rigid joint (rotoidal one for example) or a conditional joint (gravity type). This approach leads to a total uniformization of the internal representation of the world, i.e there is no distinction made at the graph level between a manipulator unit and a mobile robot system. Corresponding data structures are the same.

Such a model is intended to determine the time evolution of any articulated kinematic structure with or without kinematic loops, with or without fixed points. A general kinematic model is used.

The description of solids and environments are done by Ada programming, as mentionned above. Data are provided as simple Ada computer programs, using predefined utility packages and basic description procedures. These description primitives allow to assemble CSG objects and to attach them physical attributes. They also allow to define and create links by providing their nature and type, their number of degrees of freedom and so on. The corresponding data structure has two distinct parts: a static part containing parameters such as axis, initial values of articulations, configuration matrices... and a time dependant part: current values of articulations, position matrices and so on.

Algorithm simulation uses action and perception primitives. Action primitives are characterized by the ability to take into account evolution of the world (movements and actions operated by tools): this is for example the grasping of an object. Perception utility routines are used for collision and contact problem calculation. A ray-tracing primitive is an example.

VI. Application to vision techniques

A great number or researchers in computer vision[4,5,6,7] have solved much problems in three-dimensional object recognition. But there is much work still to be done on the problem of setting up links between an object representation scheme and sensors.

Some recent contribution on 3D machine vision techniques[8,9] are based on representation tools, such as geometric modeling, and knowledge-based robotic systems. The purpose is to

take a representation scheme, such as a boundary representation derived from the CSG representation of an object model and to obtain from it information suitable for interpreting and analyzing sensor data. The knowledge-based robotic system is used to store and recall various forms of knowledge: object representations, task plans for manipulation, task monitoring, state of the world and so on. AI techniques are useful to avoid combinatorial explosion when computing correspondences between represented sensor data and data-based models.

This approach has been successfully implemented and validated for 3D object recognition in simple worlds consisting of a small number of objects.

This interesting but difficult field is intended to be studied further on by ARES.

VII. Conclusion

We have emphasized the basic approach used in ARES for geometric modeling applied to robotics simulation. We have seen that a CSG representation was useful for building arbitrary complex objects and a derived polyhedral boundary representation was necessary for providing a direct description of the resultant object's boundary and topology.

We have also discussed the usefulness of merging together CAD information and range data information, using a knowledge-based robotic system for 3D object recognition.

REFERENCES

1- A.A. REQUICHA, "Representations for rigid solids: Theory, methods and systems", ACM Computing surveys, 12, 437-464 (1980).

2- M. SEGAL and S.H. SEQUIN, "Partitioning Polyhedral Objects into Nonintersecting Parts", IEEE Computer Graphics and Applications (Januray 1988).

3- G. DEJONGHE, "Simulation Tools for Benchmarking Robotics Systems: the ARES approach", CEA/ORNL Workshop on Autonomous Mobile Robots, Oak Ridge, Tennessee, 1989.

4- P.J. BESL and R.C. JAIN, "Three-dimensional Object Recognition", ACM Computing Surveys, 12, 75-145 (1986).

5- B.A. BOYTER and J.K. AGGARWAL, "Recognition of Polyhedra from Range Data", IEEE expert, 1, 47-59 (1986).

6- O. FAUGERAS and al., "Object Representation, Identification, and Positioning from Range Data", Robotics Research, The First International Symposium, pp 425-446 (1984).

7- T. LOZANO-PEREZ and al., "Handey: A Robot System that Recognizes, Plans and Manipulates", proceedings of the 1987 International Conference on Robotics and Automation.

8- R.J.P de FIGUEIREDO and N. KEHTARNAVAZ, "Model-Based Orientation-Independent 3D Machine Vision Techniques", IEEE Transaction on Aerospace and Electronic Systems, 24, 597-607 (1988).

9- A.C. KAK and al. "Knowledge-based Robotics", Int. J. Prod. Res., 26, 707-734 (1988).

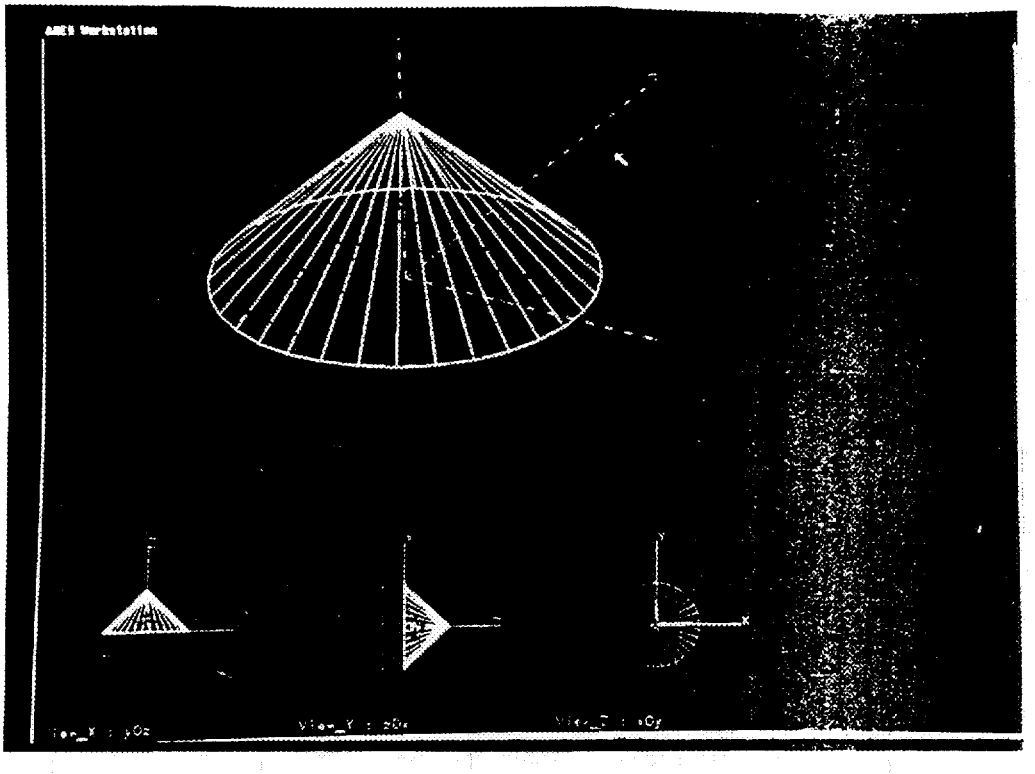


Fig. 1 Two cones

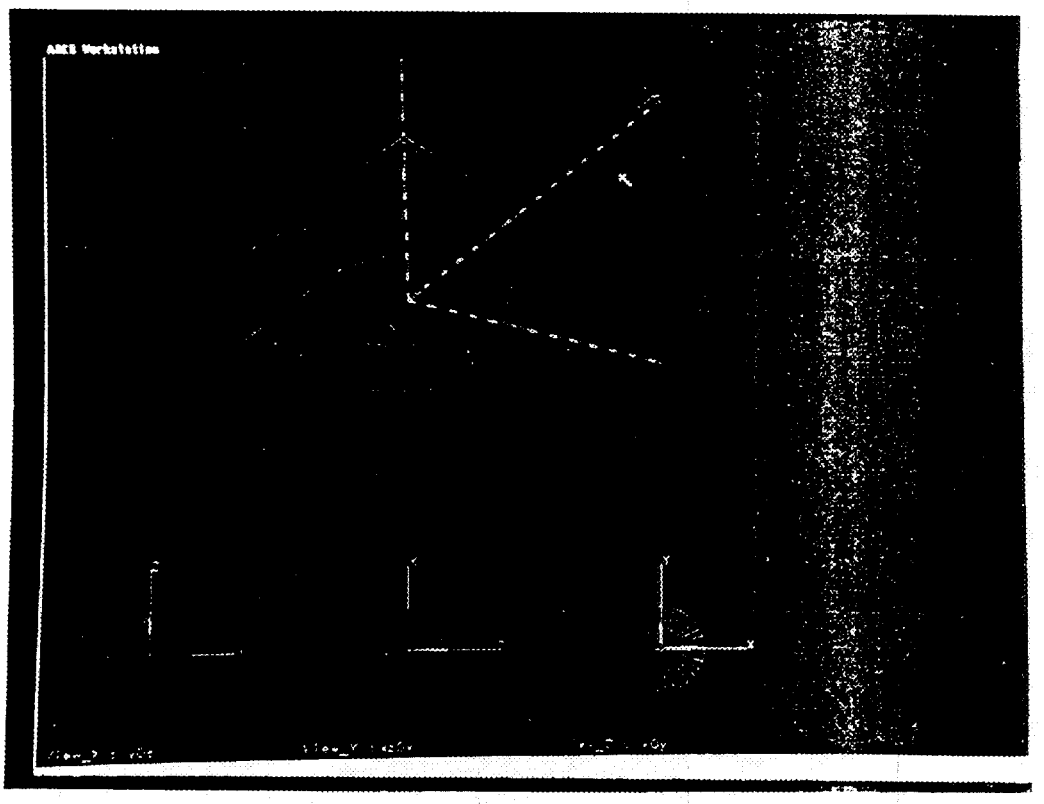


Fig. 2 Union of the two cones

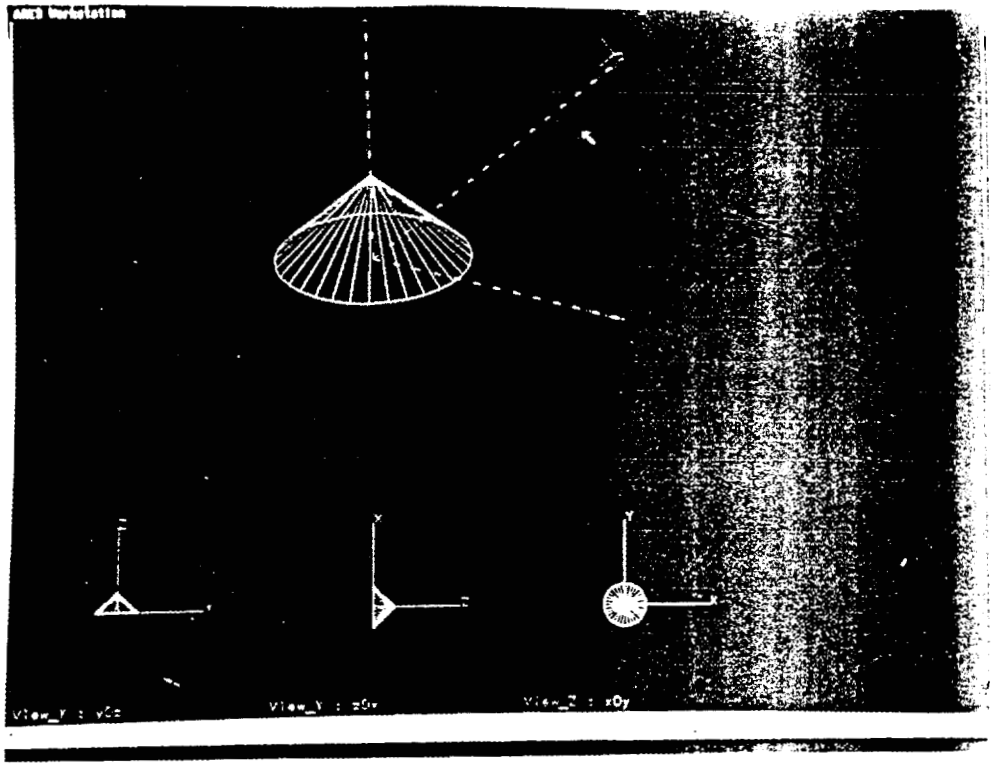


Fig. 3 Intersection of the two cones

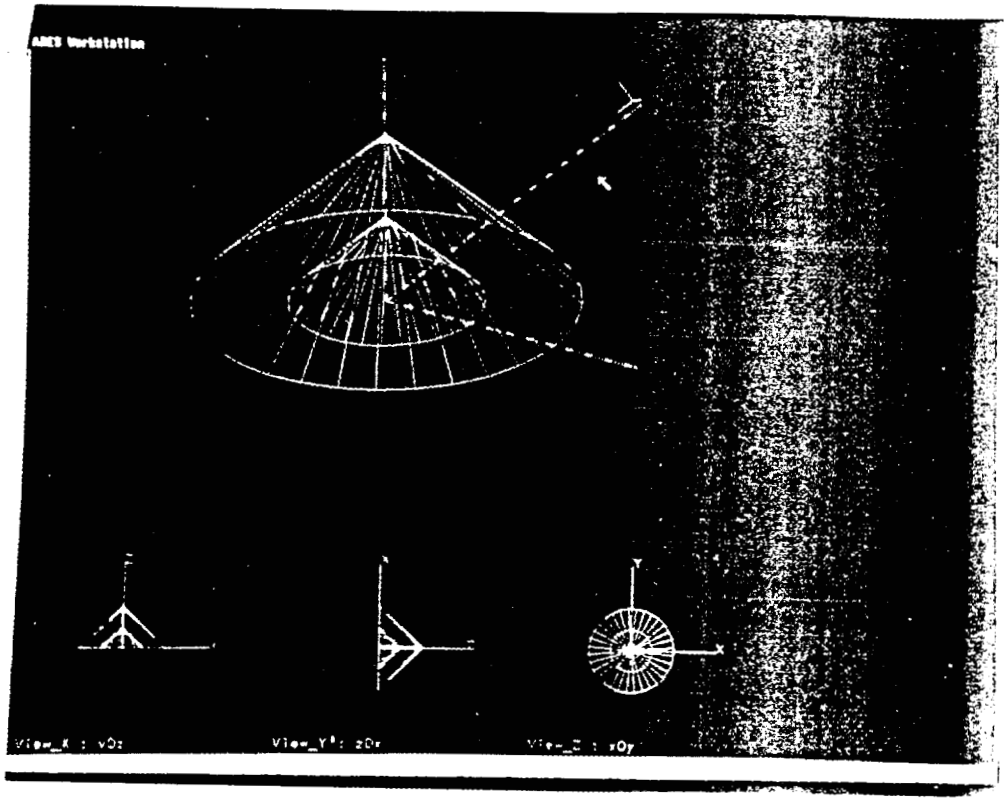


Fig. 4 Cone A \ Cone B

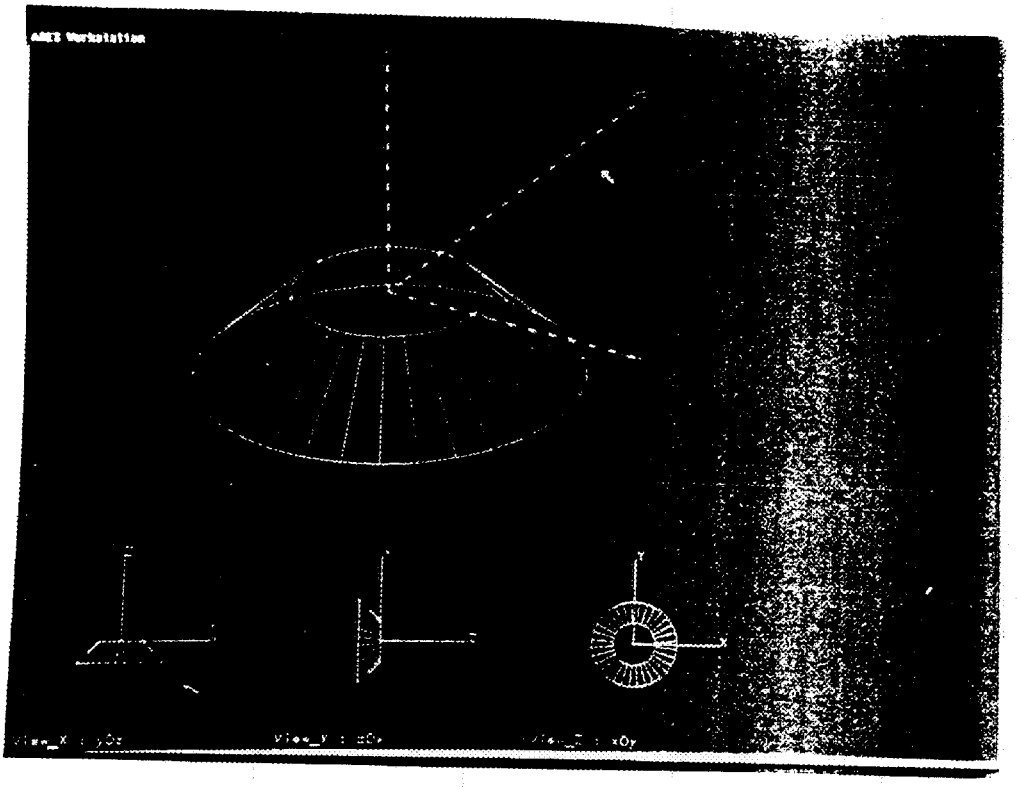


Fig. 5 Cone B \ Cone A

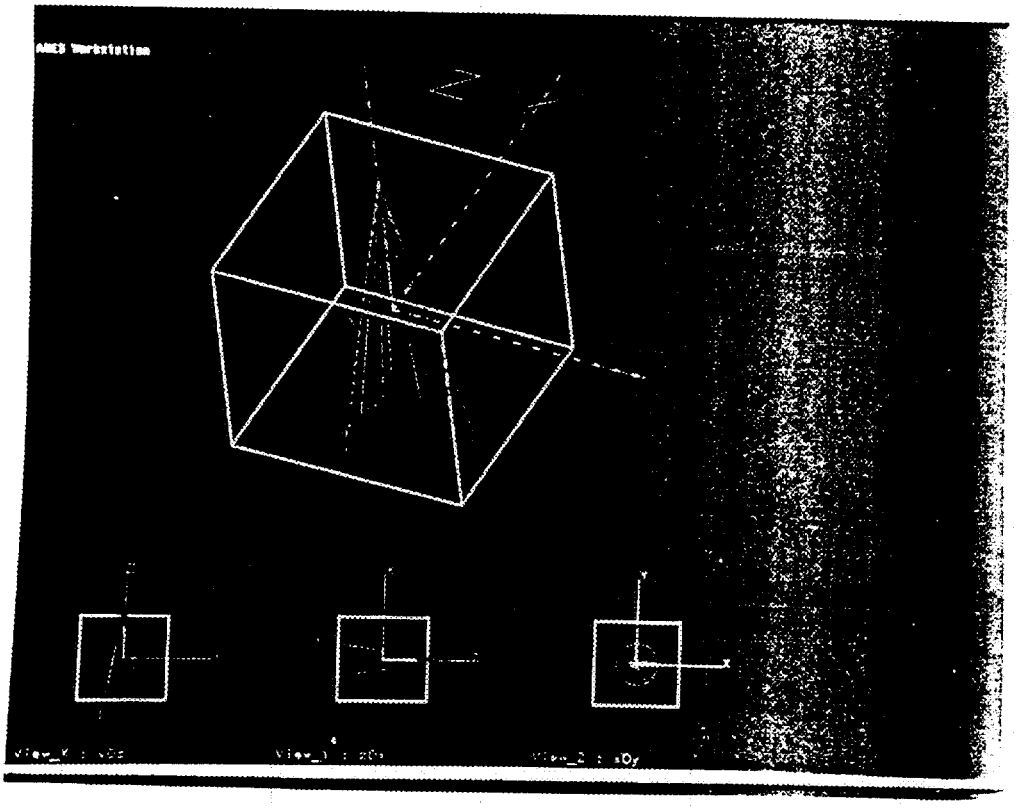


Fig. 6 A cube and a cone

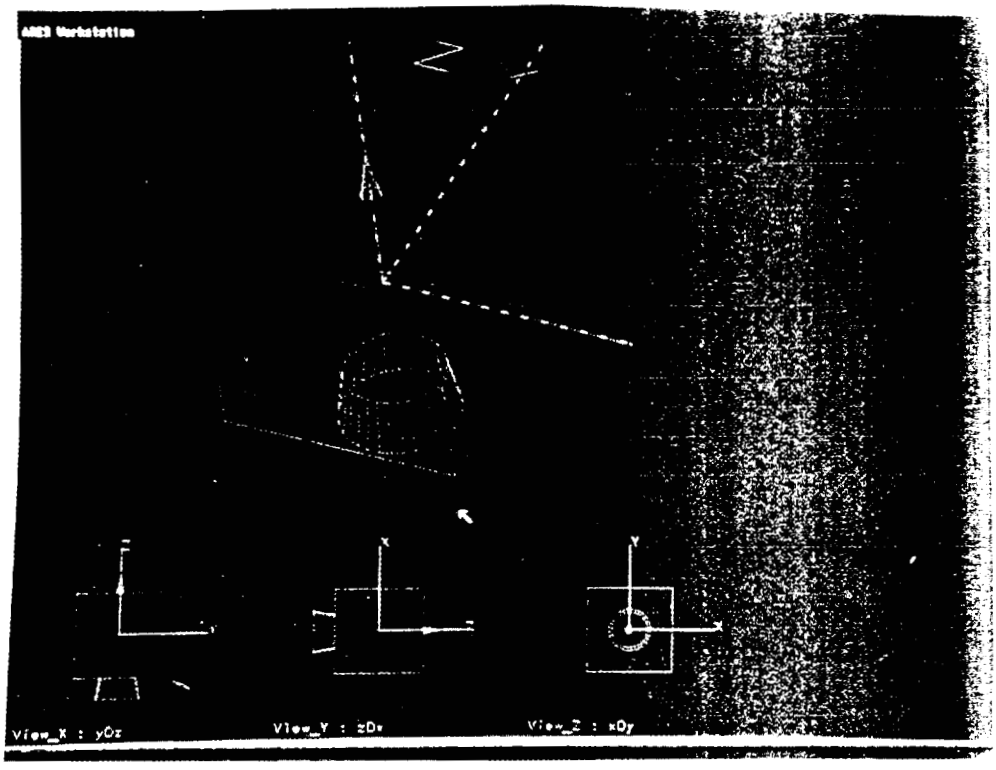


Fig. 7 Union of the cube and of the cone

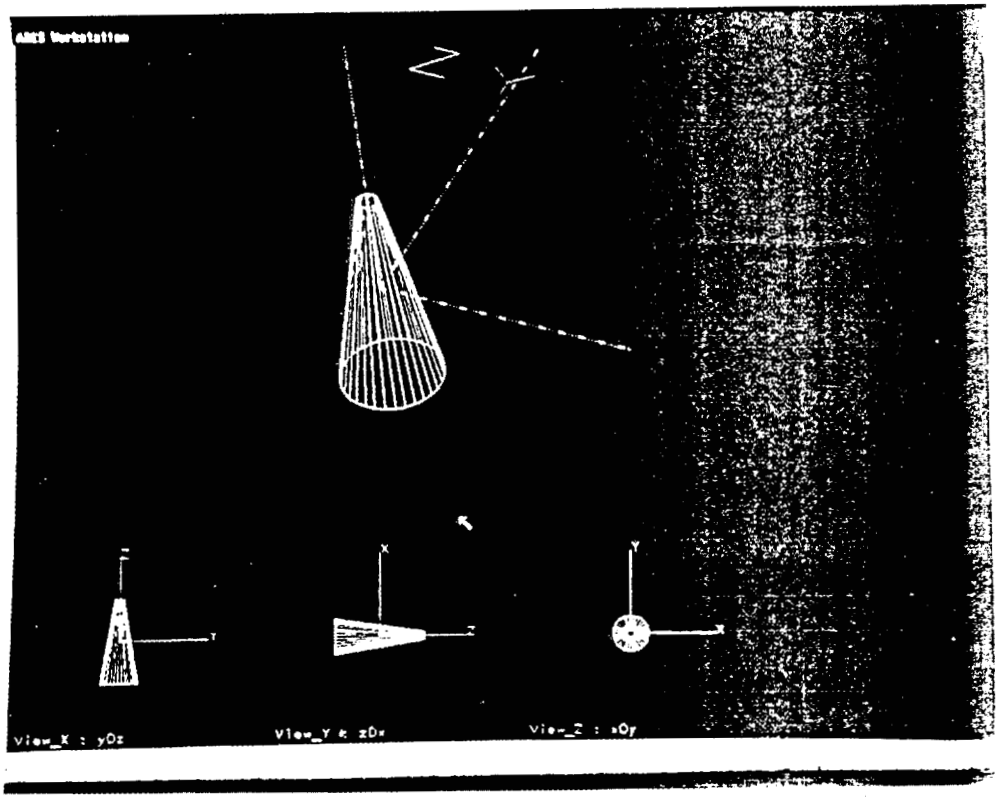


Fig. 8 Intersection of the cube and of the cone

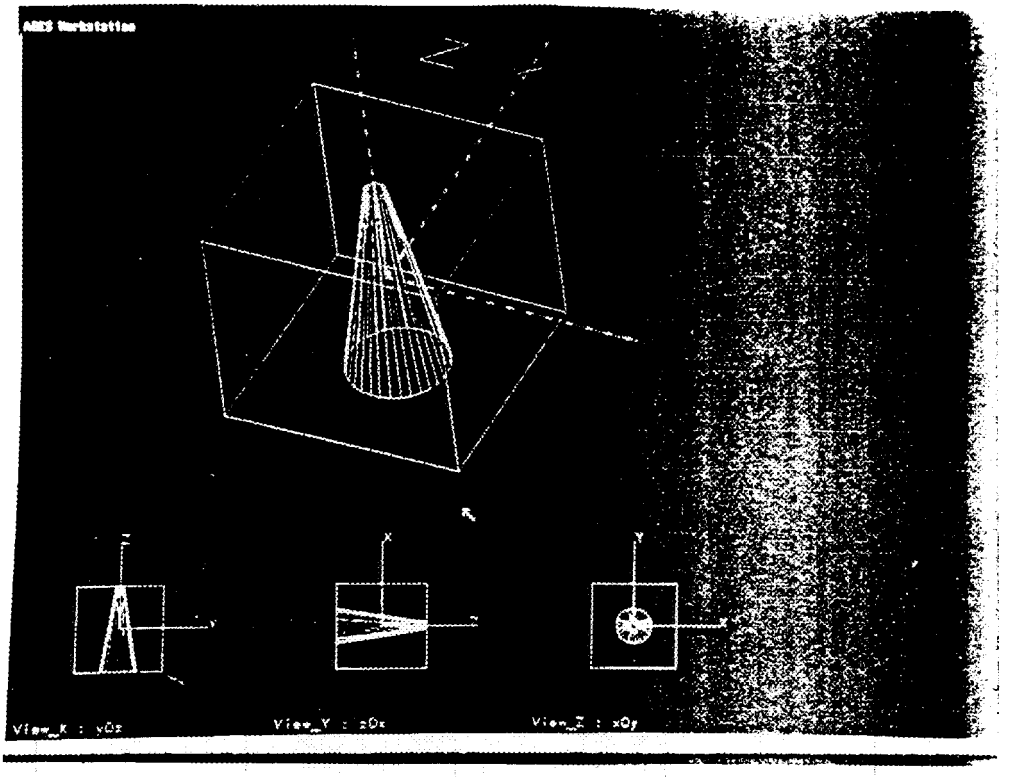


Fig. 9 Cube \ cone

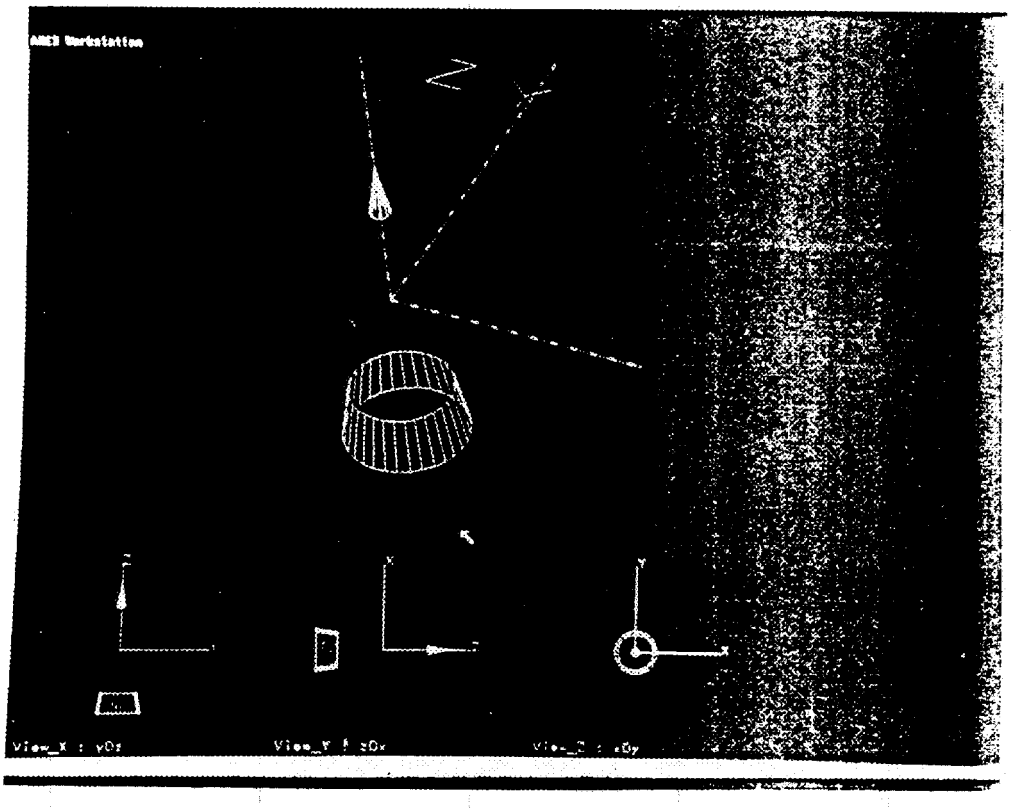


Fig. 10 Cone \ cube

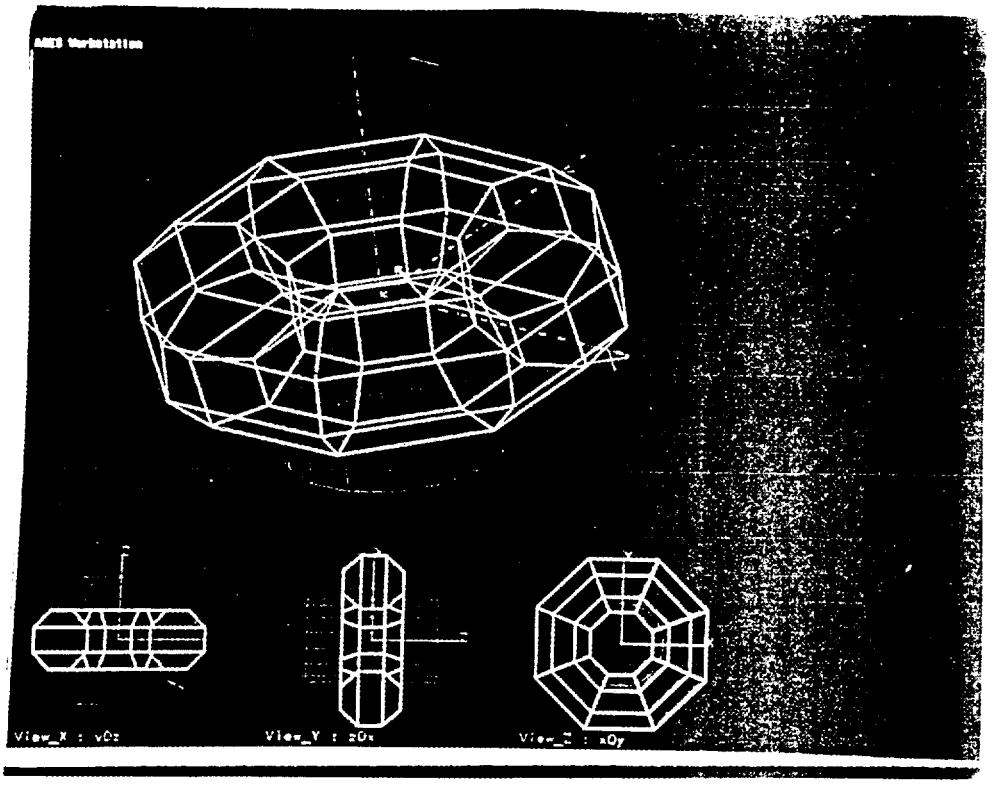


Fig. 11 A cylinder and a torus

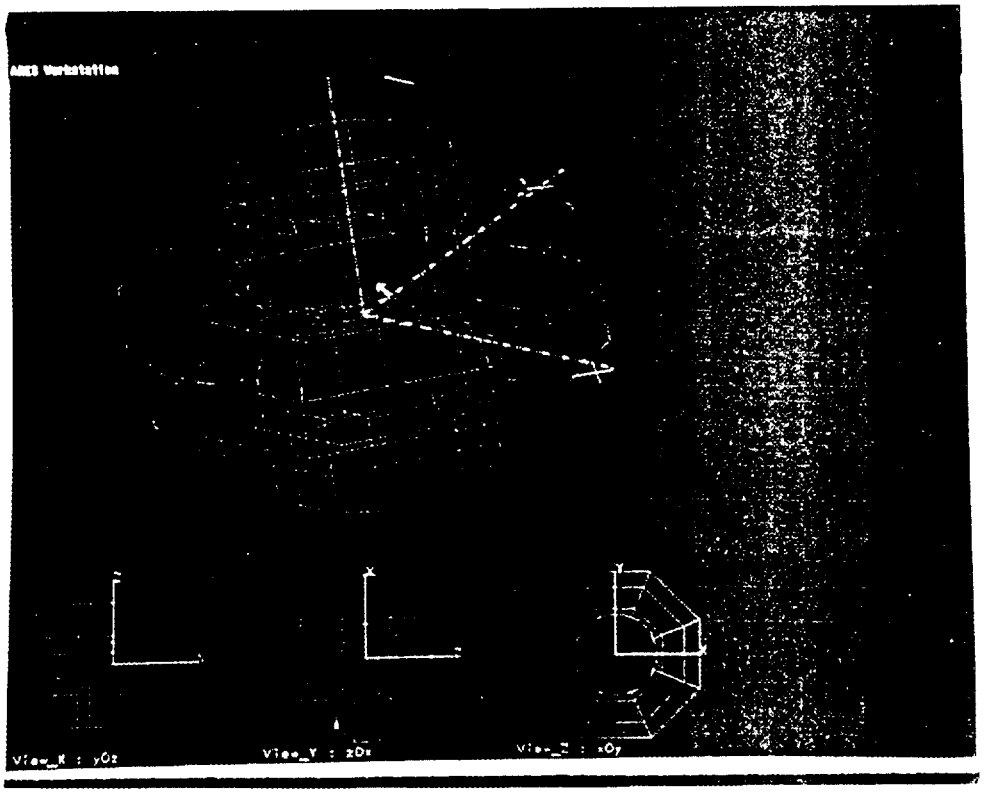


Fig. 12 Union of the cylinder and the torus

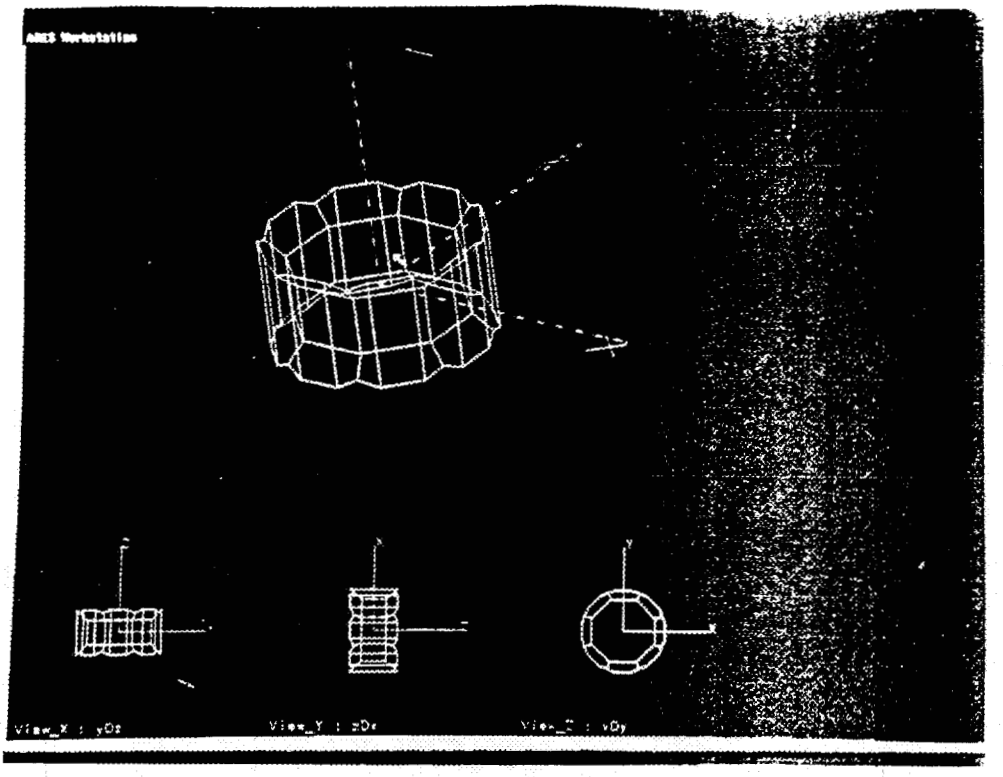


Fig. 13 Intersection of the cylinder and the torus

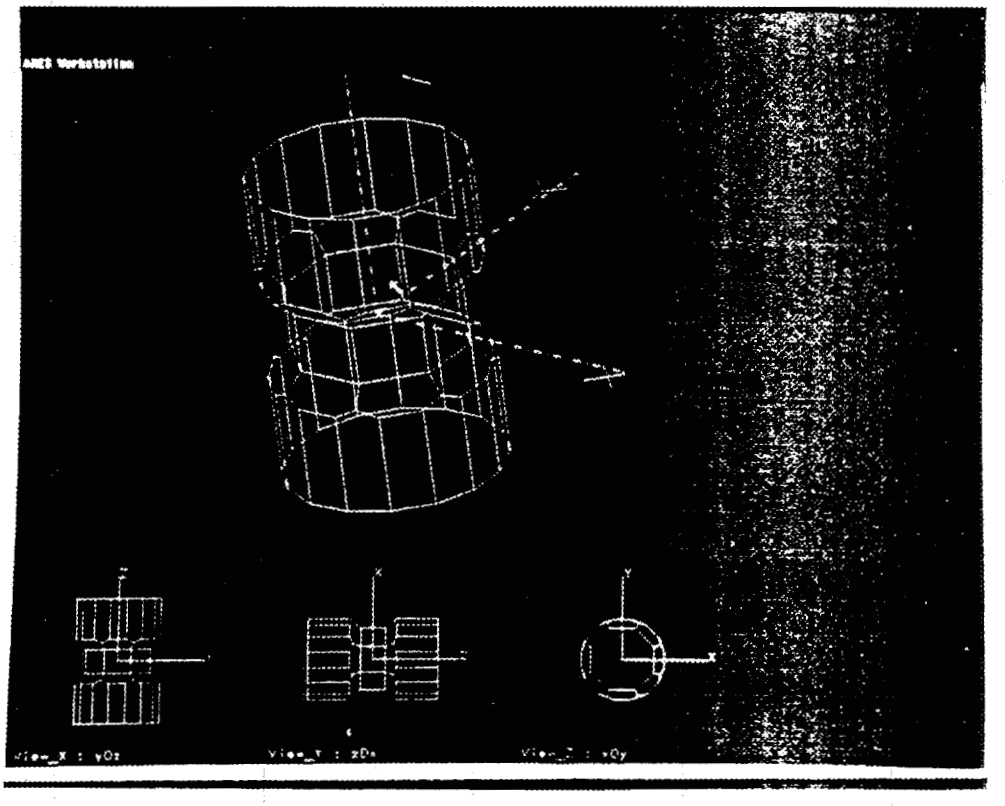


Fig. 14 Cylinder \ torus

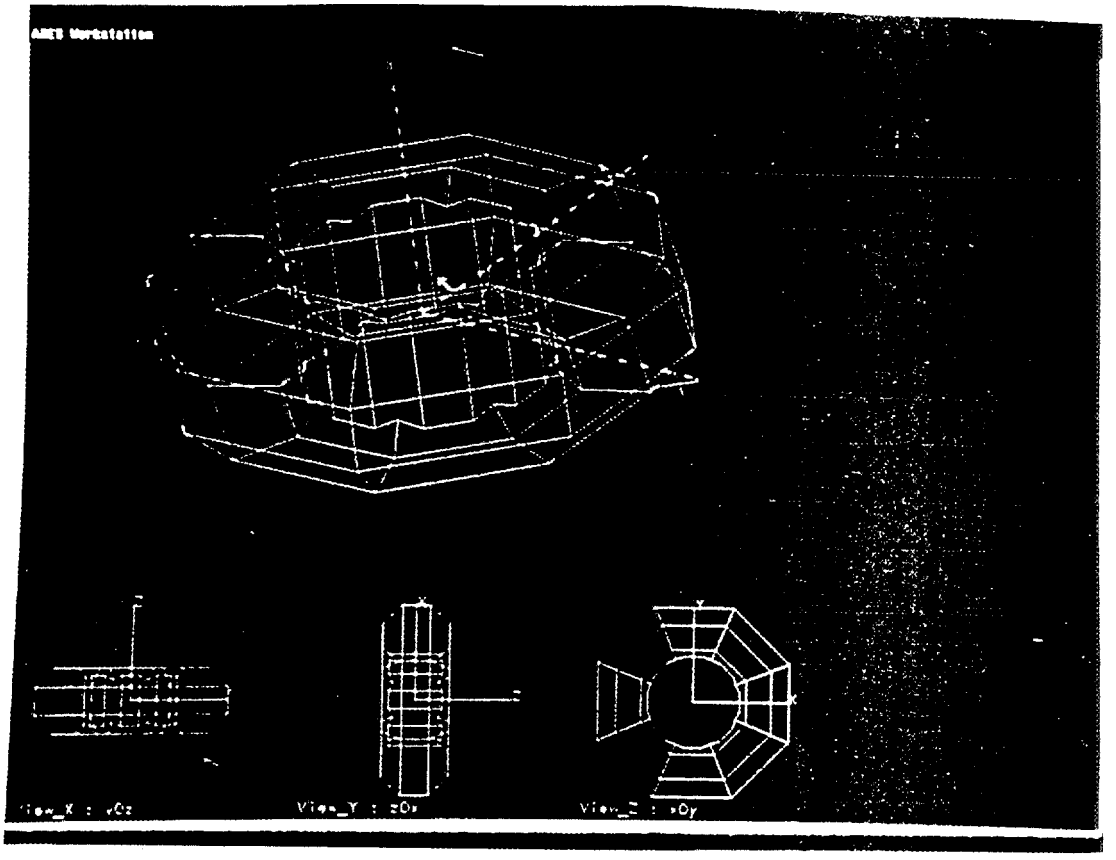


Fig. 15 Torus \ Cylinder

```
with ARES_GEOMETRY;
use  ARES_GEOMETRY;
with ARES_CSG;
use  ARES_CSG;

procedure BUILD_TWO_TABLES is

   type TYPE_OF_TABLE is (CIRCULAR,RECTANGULAR);
   CIRCULAR_TABLE        : CSG_ELEMENT;
   RECTANGULAR_TABLE     : CSG_ELEMENT;

   function T (V : VECTOR_3D)        return HOMOGENEOUS_MATRIX
                                     renames TRANSLATION;

   function TABLE (MODEL : TYPE_OF_TABLE := CIRCULAR)
                                     return CSG_ELEMENT is

      PLATE,LEG : CSG_ELEMENT;
      LEFT_FRONT_LEG, RIGHT_FRONT_LEG      : CSG_ELEMENT;
      LEFT_BACK_LEG, RIGHT_BACK_LEG        : CSG_ELEMENT;

   begin

      LEG := BOX (LENGTH => 5.0, WIDTH => 5.0, HEIGHT => 80.0);

      if MODEL = RECTANGULAR then

         LEG          := BOX (LENGTH => 5.0, WIDTH => 5.0, HEIGHT => 80.0);
         PLATE        := BOX(LENGTH => 150.0, WIDTH => 100.0, HEIGHT => 5.0) ;
         LEFT_FRONT_LEG    := T(V => (-70.0, 45.0,0.0))              *LEG ;
         RIGHT_FRONT_LEG   := T(V => ( 70.0, 45.0,0.0))              *LEG;
         LEFT_BACK_LEG     := T(V => (-70.0,-45.0,0.0))              *LEG;
         RIGHT_BACK_LEG    := T(V => ( 70.0,-45.0,0.0))              *LEG;

         return T(V => (0.0,0.0,40.0))*PLATE
                or LEFT_FRONT_LEG or RIGHT_FRONT_LEG
                or LEFT_BACK_LEG or RIGHT_BACK_LEG;

      else

         LEG          := CYLINDER (RADIUS => 7.5, HEIGHT => 80.0);
         PLATE        := CYLINDER (RADIUS => 40. HEIGHT => 0,5.0) ;
         return T(V => (0.0,0.0,40.0))*PLATE or LEG;

      end if;

   end TABLE;


begin

   -- creation of two models of table

   RECTANGULAR_TABLE     := TABLE (MODEL => RECTANGULAR);
   CIRCULAR_TABLE        := TABLE;

   PUT (RECTANGULAR_TABLE);
   PUT (CIRCULAR_TABLE);

end BUILD_TWO_TABLES;
```

Fig. 16 Build_Two_Tables Ada program

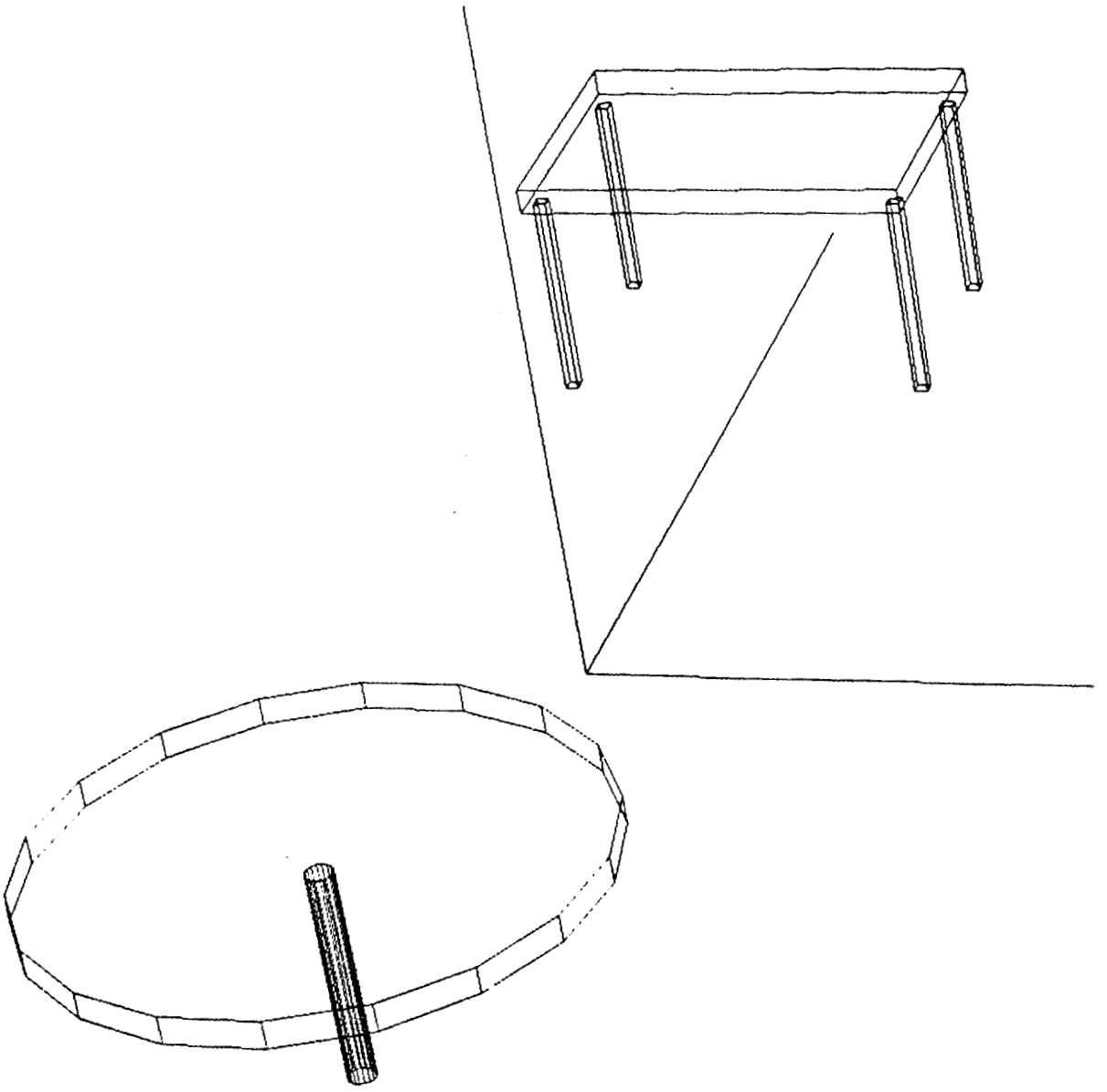


Fig. 17 Graphical result

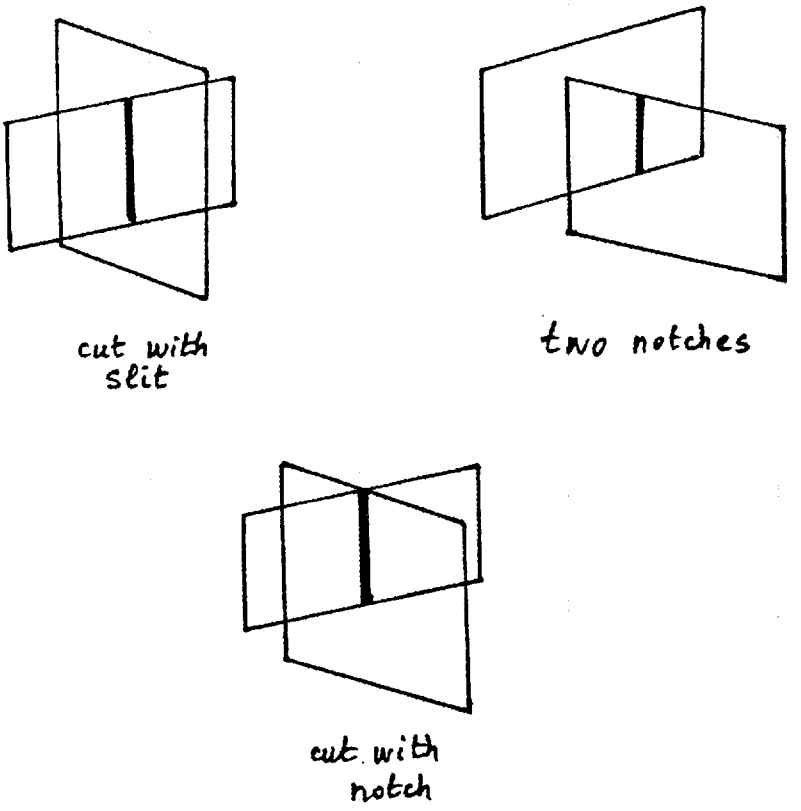


Fig. 18 Cut, slit and notch

ENVIRONMENT MODELING

- NEEDS
- CSG REPRESENTATION
- BOUNDARY POLYHEDRAL REPRESENTATION
- DERIVED PROPERTIES
- WORLD MODELING
- APPLICATION TO VISION TECHNIQUES

NEEDS

- **FOUR KINDS**

Solid modeling

CSG representation
Derived boundary representation

Graphical representation

Wire-frame representation
Polygonal representation

Physics modeling

Intrisic properties
Physical attributes

World modeling

Graph structure
Data structure

CSG REPRESENTATION

- FUNDAMENTALS

Leave nodes (primitives)

Box
Cylinder
Cone
Torus
Sphere

Non-leaf nodes (operations)

Union
Intersection
Difference

Description

Geometric package
CSG package
Ada program

CSG REPRESENTATION

- CSG OBJECT CONSTRUCTION

Ada facilities

Overloading
Packaging
Lisibility

Advantages

No specific input data language to develop
Software engineering features of Ada
Coherence and completness

Example

Construction of two models of table:
* rectangular model
* circular model

BOUNDARY POLYHEDRAL REPRESENTATION

- DATA STRUCTURES

Vertex

3 coordinates point
Pointer to list of edges
Accuracy

Edge

Two vertices joint
Pointer to list of contours of faces

Contour

Single oriented closed planar polygonal curve
Circular linked list of pointers to edges

Face

Two-dimensional finite set of contours
Normal and distance from the origin
Tolerance

Solid

Collection of boundaring faces

BOUNDARY POLYHEDRAL REPRESENTATION

- **DERIVED CSG BPR**

Fundamentals

Partitionning into nonintersecting parts
Checking faces against one another
Removing intersections from pairs of faces
Recursive process
CSG operations applied simultaneously

Sorting lists

Combination of intersecting points, sorting
Cutting interval determination
Membership information gathered

Tolerances

Vertex tolerance
Face tolerance

BOUNDARY POLYHEDRAL REPRESENTATION

- **PARTITIONING**

  Intersecting

  Main face, transversal face
  Endpoint determination

  Endpoint list determination

  Entry endpoint
  Exit endpoint
  Finding the cutting interval

  Membership information use

  Nature of the cutting interval
  Slit, notch, cut

BOUNDARY POLYHEDRAL REPRESENTATION

- SOLID MODELING

  Solid = homogeneous

  Segment information use

  Splitting edges and contours
  Merging edges and contours

  Classification

  Keeping or removing contours
  Mark propagation

  Resultant object

  Merging contours and faces
  Updating linked lists and data structures

BOUNDARY POLYHEDRAL REPRESENTATION

- EXAMPLE 1

  Two cones

  Union
  Intersection
  Difference

- EXAMPLE 2

  A cone and a box

  Union
  Intersection
  Difference

- EXAMPLE 3

  A cylinder and a torus

  Union
  Intersection
  Difference

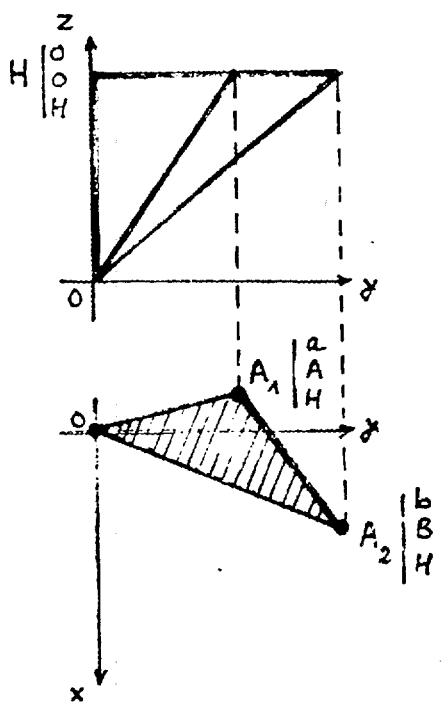
DERIVED PROPERTIES

- COMPUTATION FROM BPR

  Center of mass

  Solid angle

  Matrices of inertia



The different moments of the tetraedron are given by:

$$I_{x^2+y^2} = H.X.(a^2+ab+b^2+A^2+AB+B^2)$$

$$I_{x^2+z^2} = H.X.(a^2+ab+b^2+6H^2)$$

$$I_{y^2+z^2} = H.X.(A^2+AB+B^2+6H^2)$$

$$I_{xy} = H.X.(2aA+aB+bA+2bB)/2$$

$$I_{xz} = H^2.X.(a+b).2$$

$$I_{yz} = H^2.X.(A+B).2$$

where: $X = (aB-bA)/60$

DERIVED PROPERTIES

■ MATRICES OF INERTIA

Accuracy

Level	Accuracy
1 (20 faces)	60%
2 (80 faces)	20%
3 (320 faces)	6%
4 (1280 faces)	1.5%
5 (5120 faces)	0.3%

WORLD MODELING

- ARCHITECTURE

Graph

Solid components
Joints

Joints

From 0 to 6 DOF
Data structure

Description

Ada program

Action primitives

Perception primitives

VISION TECHNIQUES

- APPLICATIONS

3D object recognition

Object representation scheme (solid modeling)
Sensor-tuned representation

Geometric modeling

CSG representation
Derived boundary representation

AI approach

Knowledge-based robotics systems
Object representations
Task plans (manipulation of ORs)
Task monitoring
State of the world

Session IV: Planning and Navigation

Action Planning
applied to
execution of high level controls

Dominique SCHMIT
Commissariat à l'Energie Atomique
Division LETI
Département d'Electronique et d'Instrumentation Nucléaire

Control Levels in Robotics

- **Mission**
  - ▲ *Autonomous Robot*
- **Step**
  - ▲ *Semi-autonomous Robot (Supervised)*
- **Action**
  - ▲ *Industrial Robot*
- **Actuator**
  - ▲ *First Robots*

From Steps to Actions

- **Step :**
  - ***State (of robot & world) to reach***

    described with sufficient generality to be able to
    - express it in advance (off line)
    - leave sufficient initiative to robot as to prevent failure of an (unnecessary) detailed plan
- **Action :**
  - ***Detailed description of what the robot has to execute in the very near future***
    - precision as needed by low level executive

From Steps to Actions (2)

- **Our planner**
  - ***Translates a Step (High level control) into a sequence of actions (Low Level control)***
    - Adapting it to current state of world & robot
    - Respecting behaviour constraints
  - ***Does mainly symbolic processings, so***
    - Uses (or helps) a path planner to find motion actions
    - Uses perception, and when in doubt planifies future perception

XGP : An Action Planner

- **Features :**
  - *produces linear (ordered) plans*
  - *non hierarchical representations*
  - *short life plan*
- **Principle of resolution :**
  - *backward chaining & depth first search (goal oriented)*
  - *reduction*
    - try not to destroy any goal already reached
  - *ordered goals*
- **Written in PROLOG & C**

Basic Ideas

- **Ability to add heuristics in the description of actions**
  - ***about the choice of the action depending on***
    - desired effect
    - situation (context) in which the action is considered
  - ***about the order to follow in the realisation of its preconditions***
- **Ability to model actions having different consequences depending on its use**

Facts

- **Described by**
  - class
  - name
  - list of attributes
- **Classified**
  - ***Facts set by actions (goals)***
  - ***Facts imposed (no means to act on)***
  - ***Logical conditions***
  - ***Facts relevant to the planification process***
  - ***Facts whose value is given by other processes (e.g. path planner, perception)***

Modelling of Actions

- **One action may have several versions**
- **Each version described by 5 lists of facts**
  - ***Desired effects***
    - allow to choose the action
    - caracterise the reason why the action was chosen
  - ***Consequences***
    - add list
    - delete list
    - ▲ *Remark : Desired effects are part of Add List or possibly of Delete List*

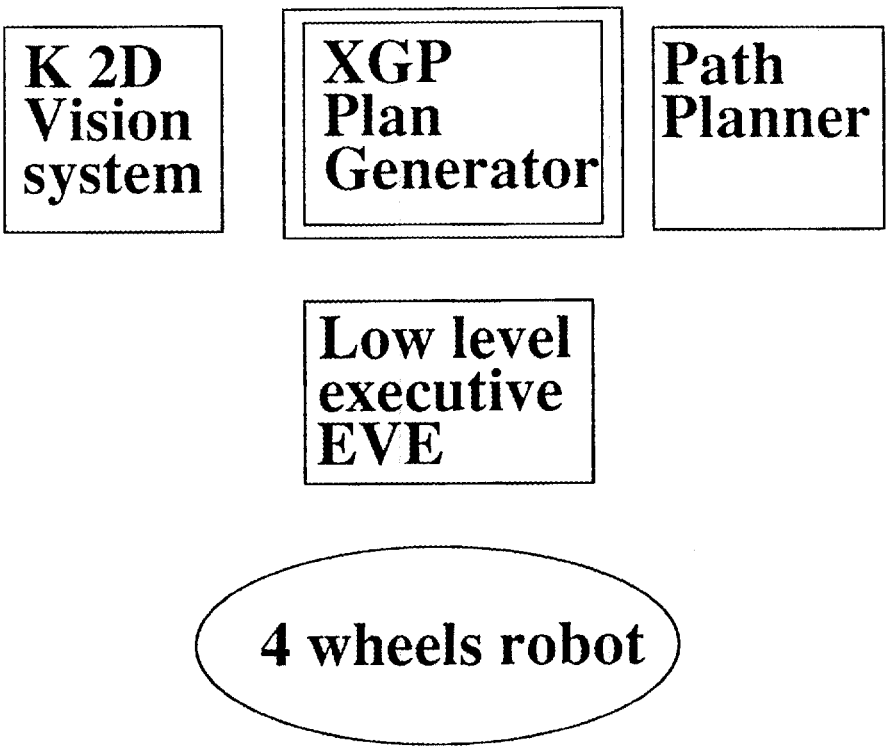
Modelling of Actions (2)

- ***Contextual preconditions***
  - facts to be established when the action is considered
  - never cause a plan to be generated
- ***Execution preconditions***
  - facts to realise before the action can be executed
  - ordered depending on the use of the action

Action Choice

- **Having a goal to achieve**
- **Consultation of desired effects lists**
  - leads to a choice of an action
- **Are protected goals destroyed ?**
  - if YES : try to insert action in previous plan
  - if NO : try to extend plan by this action
- **Are contextual preconditions verified ?**
  - if NO : choose another action
- **Do executive preconditions lead to a loop situation ?**
  - if YES : try to find other contextual preconditions verified or another action
- **Find an action plan such that executive preconditions are verified**

Experiment


Operator
K 2D Vision system
XGP Plan Generator
Path Planner
Low level executive EVE
4 wheels robot

Job Planning and Execution Monitoring for a Human-Machine Symbiotic System

Lynne E. Parker

CESAR/CEA Workshop on Autonomous Mobile Robots
May 31, 1989

CESAR

Human-Machine Symbiosis Bridges the Gap Between Manual and Autonomous Systems

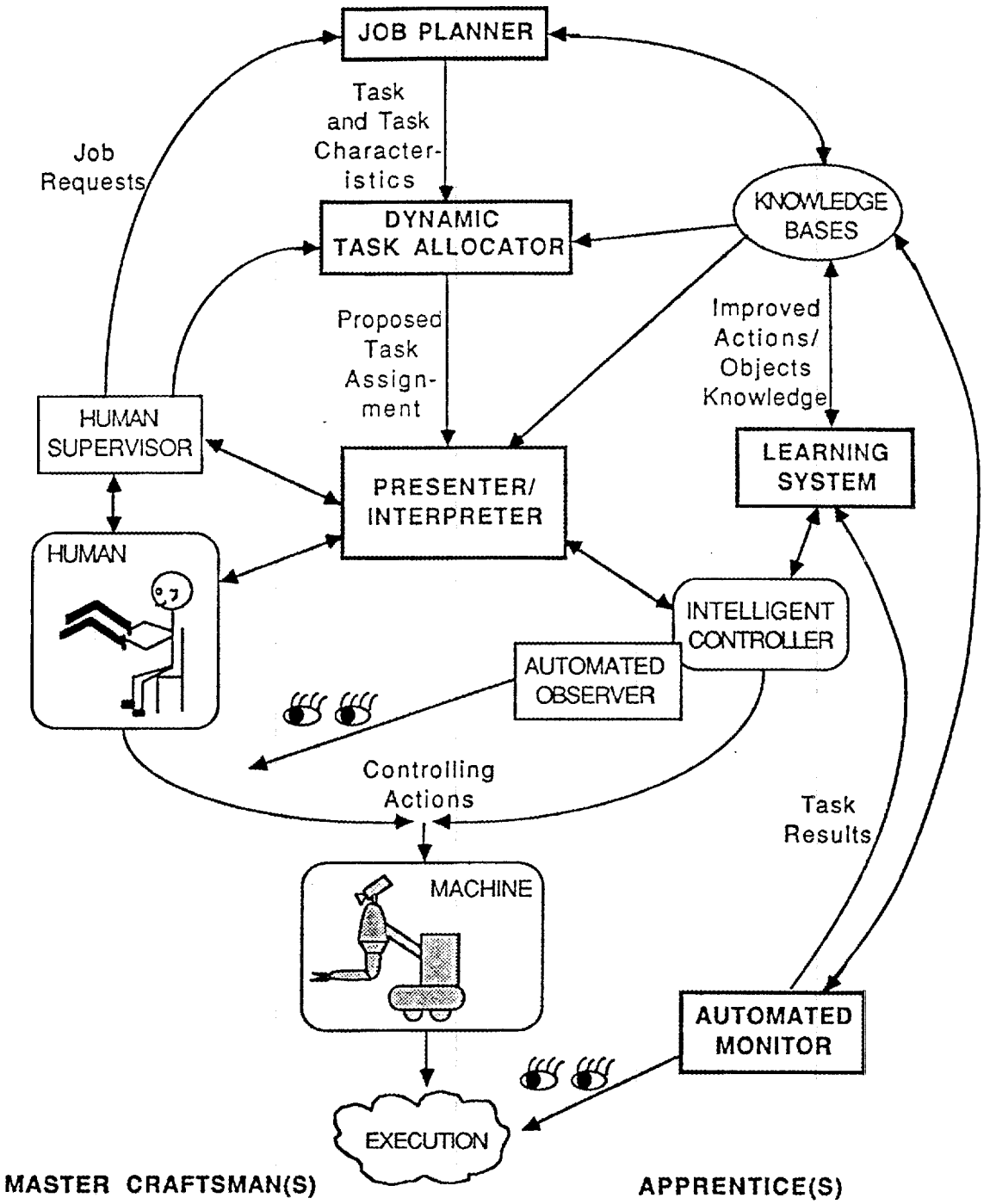
AUTONOMOUS SYSTEMS	HUMAN-CONTROLLED SYSTEMS
• Automation of repetition	• Not optimized for repetition
• Improved speed, accuracy, efficiency	• Human fatiguing slow
• Expertise in narrow task domain	• Wide task domain
• No ability to cope with unexpected events	• Rely on human to cope with unexpected events
• Limited unsupervised learning	• Supervised learning is possible; play-back is rigid

HUMAN-MACHINE SYMBIONT

- Human in the loop for innovative reasoning and decision-making
- Improved speed, accuracy, efficiency
- Increasing replacement of human in repeated tasks
- Increasing expertise in wide task domain
- Capability for supervised and unsupervised learning
- Ability to cope with unexpected events

ornl

Human-Robot Symbiosis Framework



Job Planner Establishes the Actions (Subtasks) Required to Accomplish the Goal

- Job Planner is responsible for planning the primitive task activity sequences that lead to efficient job completion
- Resulting subtask structure should allow rapid reconfiguration due to unexpected events or human interaction
- Subtask structure will be provided to the Dynamic Task Allocator for resource assignment

ornl

Job Planning Methodology is Closely Related to an Action-Object Language

- Define:
  - -- A set of valid actions which can be performed in the environment
  - -- A set of objects which exist in the environment
  - -- The relationships between actions and objects

- These sets will likely vary over time as new actions are learned, or as new objects appear

- Each of the components of the symbiont architecture use this language to accomplish its objectives:
  - -- Job Planner: plans **actions** to be performed on **objects**
  - -- Dynamic Task Allocator: assigns **actions** to be performed to Human or Robot
  - -- Automated Monitor: observes execution of **actions** or states of **objects**
  - -- Learning System: learns new **actions** or **objects**

Job Planner Uses a STRIPS-like Planning Strategy (Theorem-Proving)

GIVEN:

- Initial starting state (modified 1st order predicate calculus statements)
- Goal state (modified 1st order predicate calculus statements)
- Set of operators

FIND:

the sequence of operators which transforms the planning world model from the initial state to the goal state

OPERATOR:

- a description of an action which may be performed by the agent (human or machine)
- contains 3 lists:

  PRECONDITIONS list: conditions which must be true prior to application of operator

  ADD list,
  DELETE list: conditions added to/deleted from the world model subsequent to operator application

ornl

JOB PLANNER -- Example

- o Job planning prototype implemented in "C"
- o Successfully plans sequential tasks

Example manipulation task:

Starting conditions:

At(Hand, Start)	Free(Bolt3)
Handempty	Free(Bolt4)
Connected(Bolt1)	Connected(Tube_jmpr)
Free(Bolt2)	Connected(Elec_conn)

Goal: Free(Bolt1)

Subset of operator rules:

```
Grasp(*object)        PRECONDITIONS: Handempty
                                     Found(*object:Grasp_loc)
                                     At(Hand,*object:Grasp_loc)
                                     END
                         DELETE_LIST:Handempty
                                     END
                            ADD_LIST:Grasped(*object)
                                     END

Move_Arm(*from_loc,  *to_loc)
                      PRECONDITIONS:
                                     Path_planned(*from_loc,*to_loc)
                                     END
                         DELETE_LIST: At(Hand,-)
                                     END
                            ADD_LIST: At(Hand,*to_loc)
                                     END

Extract(#Bolt)        PRECONDITIONS: Grasped(Wrench)
                                     Found(#Bolt:Unscrew_loc)
                                     At(Hand,#Bolt:Unscrew_loc)
                                     END
                         DELETE_LIST: Connected(#Bolt)
                                     END
                            ADD_LIST: Free(#Bolt)
                                     END
```

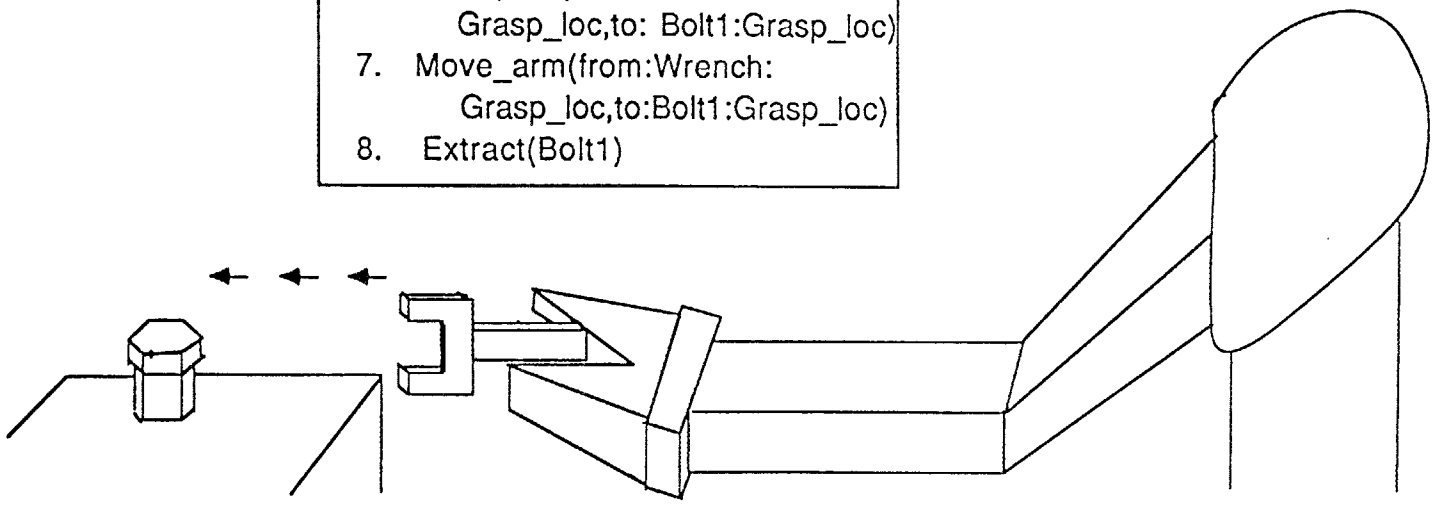
CESAR

Resulting Job Plan

Job Plan

1. Find (Wrench:Grasp_loc)
2. Plan_path(from: start,
   to: Wrench:Grasp_loc)
3. Move_arm(from: start,
   to: Wrench:Grasp_loc)
4. Grasp(Wrench)
5. Find(Bolt1:Grasp_loc)
6. Plan_path(from:Wrench:
   Grasp_loc,to: Bolt1:Grasp_loc)
7. Move_arm(from:Wrench:
   Grasp_loc,to:Bolt1:Grasp_loc)
8. Extract(Bolt1)

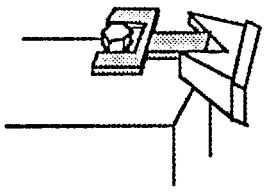
TO TASK ALLOCATOR FOR RESOURCE ASSIGNMENT

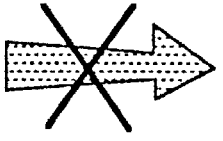


CESAR

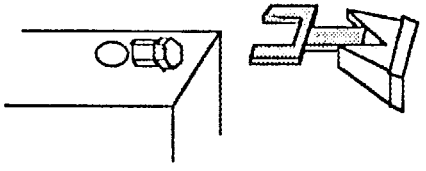
Automated Monitor
Example of Unexpected Event

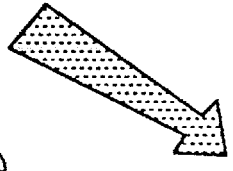
Example: Robot
Unscrewing Bolt

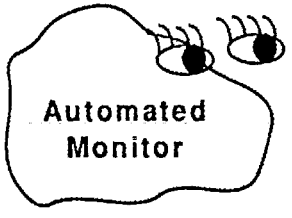




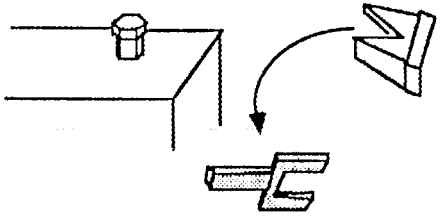
Expected Result:
Bolt Unscrewed







Actual Result:
Wrench Dropped



Actions of Automated Monitor:

- o Inform Job Planner of actual result
  - - Work with Job Planner to derive new plan to fix problem and reach goal
- o Possibly update information on capabilities of robot
  - - Probability of robot successfully "unscrewing Bolt" decreases

CESAR

Automated Monitor, Function 1: Detect "Unexpected" Events

Define:

(1) $S = \{s_1, s_2, \ldots, s_n\}$ (S = available sensor suite)
where n = total # of sensors

(2) $C = \{c_1, c_2, \ldots, c_m\}$ (C = generic types of conditions which describe the environment)
where m = number of types of conditions

(3) $A = \{a_1, a_2, \ldots, a_n\}$ (A = actual sensor readings at a given point in time)

(4) For all c_i, V_i^1, V_i^2, ... = sets of valid sensor readings expected for condition c_i,
where:

$$V_i^1 = \{v_{i1}^1, v_{i2}^1, \ldots, v_{in}^1\}$$

$$V_i^2 = \{v_{i1}^2, v_{i2}^2, \ldots, v_{in}^2\}$$

Then:

For expected condition c_i, if $a_j = v_{ij}$, $j = 1, \ldots, n$, for some set V_i, no problem exists; else, a discrepancy has been detected, and the human will be notified of the potential problem.

But ... Mapping expected conditions to expected sensor readings is not enough . . .

- Detection of discrepancies in sensor readings comes directly from information on conditions which must be true at any given point in time
- Question: How does the Automated Monitor know what conditions should be true at any point in time?
- Information does not directly come from current action (task) being executed.

  example:

  "Move_arm(from,to)" is the same action whether an object is in the gripper or not. But, the sensor readings are not the same.

- Requires knowing detailed information about the overall plan being executed, not just the current action.

Automated Monitor must receive an Execution Monitoring Table (EMT) from the Job Planner

- Since the Job Planner knows the overall intent of the plan being executed, it can provide information on what conditions should be true at any point in the execution of the plan

- EMT is created as the job is planned

- EMT provides input to the AM indicating expected conditions:
  - -- prior to subtask execution ("Preconditions")
  - -- during subtask execution ("Continuing Conditions")

Creation of Execution Monitoring Table

Job Planner uses the following method to plan the job to be executed:

Given:

C: set of conditions available for describing environment

g: goal condition

T: set of actions that can be taken, where each T_i consists of 3 lists:

P_i , list of preconditions

A_i , list of add conditions

D_i , list of delete conditions

where each $p_{ih}, a_{ij}, d_{ik} \in C$

Find:

Sequence of actions $t_0, \ldots, t_n$ such that condition g is true subsequent to action t_n .

EMT consists of the following for each task, t_k :

Preconditions: preconditions, p_k , of action t_k

Continuing conditions: all add_list conditions, A, of previous tasks $t_0, \ldots, t_{k-1}$ which have not yet appeared on the preconditions list, P, of a subsequent task.

Execution Monitoring Table Consists of Preconditions And Continuing Conditions

Preconditions: conditions which must be true prior to subtask execution

Continuing conditions: conditions which must be true during subtask execution

example:

subtask	preconditions	continuing conditions
Find(Casing1:Grasp_loc)	---	---
Move_arm(Starting_loc) Casing1:Grasp_loc)	At(Hand, Starting_loc)	Found(Casing1: Grasp_loc)
Grasp(Casing1)	Handempty Found(Casing1: Grasp_loc) At(Hand,Casing1: Grasp_loc)	---
Find(Jig_lower_center)	---	Grasped(Casing1)
Move_arm(Casing1: Grasp_loc, Jig_lower_center)	At(Hand,Casing1: Grasp_loc)	Grasped(Casing1) Found(Jig_ lower_center)
Place(Casing1, Jig_lower_center)	Grasped(Casing1) Found(Jig_ lower_center) At(Hand,Jig_lower_ center)	---

DERIVED JOB PLAN

Plan Number	Task Description
1	Find(Casing1)
2	Move_Arm(Curr_Loc,Casing1:Hover_pos)
3	Grasp(Casing1)
4	Find(Jig_lower_center)
5	Move_Arm(Curr_Loc,Jig_lower_center>Hover_pos)
6	Place(Casing1,Jig_lower_center)
7	Release(Casing1)
8	Find(Lever)
9	Move_Arm(Curr_Loc,Lever:Hover_pos)
10	Grasp(Lever)
11	Find(Jig_axis)
12	Move_Arm(Curr_Loc,Jig_axis>Hover_pos)
13	Place(Lever,Jig_axis)
14	Release(Lever)
15	Find(Spacer)
16	Move_Arm(Curr_Loc,Spacer:Hover_pos)

Press <RET> to continue or Q to quit:

DERIVED JOB PLAN

Plan Number	Task Description
17	Grasp(Spacer)
18	Find(Casing_1_bottom_edge)
19	Move_Arm(Curr_Loc,Casing_1_bottom_edge>Hover_pos)
20	Place(Spacer,Casing_1_bottom_edge)
21	Release(Spacer)
22	Find(L_pin_1)
23	Move_Arm(Curr_Loc,L_pin_1:Hover_pos)
24	Grasp(L_pin_1)
25	Find(Casing_1_bottom_left_hole>Hover_pos)
26	Move_Arm(Curr_Loc,Casing_1_bottom_left_hole>Hover_pos)
27	Insert(L_pin_1,Casing_1_bottom_left_hole)
28	Release(L_pin_1)
29	Find(L_pin_2)
30	Move_Arm(Curr_Loc,L_pin_2:Hover_pos)
31	Grasp(L_pin_2)
32	Find(Casing_1_bottom_right_hole>Hover_pos)

Press <RET> to continue or Q to quit:

```
Subtask id: Find(Casing1)
          Preconditions:

          Continuing Conditions:



Subtask id: Move_Arm(Curr_Loc,Casing1:Hover_pos)
          Preconditions:

          Continuing Conditions:
               Found(Casing1)



Subtask id: Grasp(Casing1)
          Preconditions:
               Handempty
               Found(Casing1)
               At(Hand,Casing1:Hover_pos)

        Continuing Conditions:



Subtask id: Find(Jig_lower_center)
          Preconditions:

          Continuing Conditions:
               Grasped(Casing1)



Subtask id: Move_Arm(Curr_Loc,Jig_lower_center>Hover_pos)
          Preconditions:

          Continuing Conditions:
               Found(Jig_lower_center)
               Grasped(Casing1)



Subtask id: Place(Casing1,Jig_lower_center)
          Preconditions:
               Grasped(Casing1)
               Found(Jig_lower_center)
               At(Hand,Jig lower_center>Hover_pos)

          Continuing Conditions:
```

```
Subtask id: Release(Casing1)
          Preconditions:
               Grasped(Casing1)

          Continuing Conditions:
               Positioned(Casing1,Jig_lower_center)



Subtask id: Find(Lever)
          Preconditions:

          Continuing Conditions:
               Handempty
               Positioned(Casing1,Jig_lower_center)



Subtask id: Move_Arm(Curr_Loc,Lever:Hover_pos)
          Preconditions:

          Continuing Conditions:
               Found(Lever)
               Handempty
               Positioned(Casing1,Jig_lower_center)



Subtask id: Grasp(Lever)
          Preconditions:
               Handempty
               Found(Lever)
               At(Hand,Lever:Hover_pos)

          Continuing Conditions:
               Positioned(Casing1,Jig_lower_center)



Subtask id: Find(Jig_axis)
          Preconditions:

          Continuing Conditions:
               Grasped(Lever)
               Positioned(Casing1,Jig_lower_center)



Subtask id: Move_Arm(Curr_Loc,Jig_axis>Hover_pos)
          Preconditions:

          Continuing Conditions:
               Found(Jig_axis)
               Grasped(Lever)
               Positioned(Casing1,Jig_lower_center)
```

JOB PLANNER -- Continuing Work

- Incorporate hierarchical job planning

- Examine the concept of optimal job plans, where optimal can be in terms of:
  - -- logic: the most "sensible" path
  - -- cost: the path with the fewest number of operators
  - -- time: the path which is the quickest to execute

- Incorporate planning with time constraints

CESAR

DEMT/SYST/MRS

3-D simulation of a world modeling and navigation method

J P. Vouiné

DEMT/SYST/MRS

2 papers:

* 3-D world modeling based on combinatorial geometry for autonomous robot navigation

* 3-D world modeling with updating capability based on combinatorial geometry

M. Goldstein
F.G. Lin
G. de Saussure
C.R. Weisbin

DEMT/SYST/MRS

1. Recall / summary of the papers

2. Simulation tools

3. Implementation

4. Conclusion

DEMT/SYST/MRS

1. Approach summary

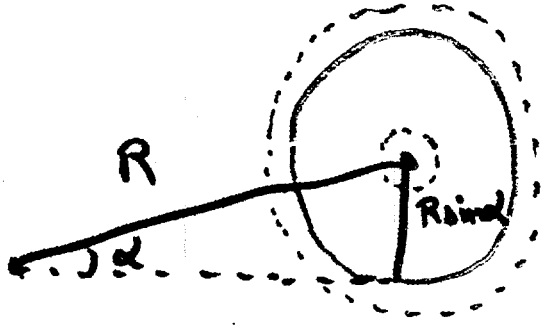
- need of 3D envt modeling for mobile robots
- interest of range finder vs camera (range map vs intensity image)
- build a boundary rep. from range data (apparent shape of obstacles)

external world ⟶ internal model for decision making

DEMT / SYST / MRS

1. Approach summary

- sensor "impacts" surrounded by spheres (compact data), taking neighborhood relationships into account:



DEMT / SYST / MRS

1. Approach summary

- contiguity property ⟶ gather overlapping spheres into zones representing connex parts of obstacles.

- possibility to perform efficient distance computations to the obstacles (zones - spheres hierarchy)

DEMT/SYST/MRS

1. Approach summary

updating capabilities (2nd paper)

- merge / update data w.r.t successive scans
- accuracy ↑ with proximity to obstacles (R ↓, so do the spheres radii)
- check pre-learned data vs acquired data
- moving obstacles

DEMT/SYST/MRS

1. Approach summary

navigation (heuristic)

- scan - world modeling (zones)
- the robot looks for discontinuities between zones
- tests possibility to go through discontinuities
- chooses nearest discontinuity (w.r.t the goal)
- proceeds a fraction of the free way
- iterates if needed

DEMT / SYST / MRS

Purpose of simulation

- help understanding how an alg. works by visualizing it while running
  (algorithm animation)
- possibly compare algorithms

- heuristics → test different situations

DEMT/SYST/MRS

2. Simulation tools

- ARES basic modeler
  simplified CSG (union)
  ↳ environment "externae" modeling

- package of intersection computation routines
  ↳ sensor data acquisition simulation

DEMT/SYST/MRS

2. Simulation tools

- utility graphics routines
  - mobile robot visualization
  - acquired / processed sensor data visualization
- simulation output routines

DEMT/SYST/MRS

2. Simulation tools

- user interface
  [graphics WS peripheral devices]
  - turnbuttons
    ↳ image manipulation
  - key panel
    ↳ graphical / functional actions trigger
    * display / undisplay objects
    * hold / continue / restart / stop simulation
  - tablet + cursor
    ↳ point inputs

DEMT/SYST/MRS

2. Simulation tools

- robot
  - parallelepiped
  - sensor implantation
  - allowed to perform straight moves or turns [consistent with the capabilities of a two driving but not steering wheel platform - not optimal]

DEMT/SYST/MRS

2. Simulation tools

- range finder

  one intersection computation for each ray

  possibility to define the limits of the scanned field, a maximum range.

- proximity sensors or "whiskers"

DEMT/SYST/MRS

2. Simulation tools

- limitations

  programming interface (F77)

  no moving obstacles modelized

- computation time

  1 intersection ≃ 1 ms [geometrice]

  graphics processing
  algorithm processing

  reasonably fast animation

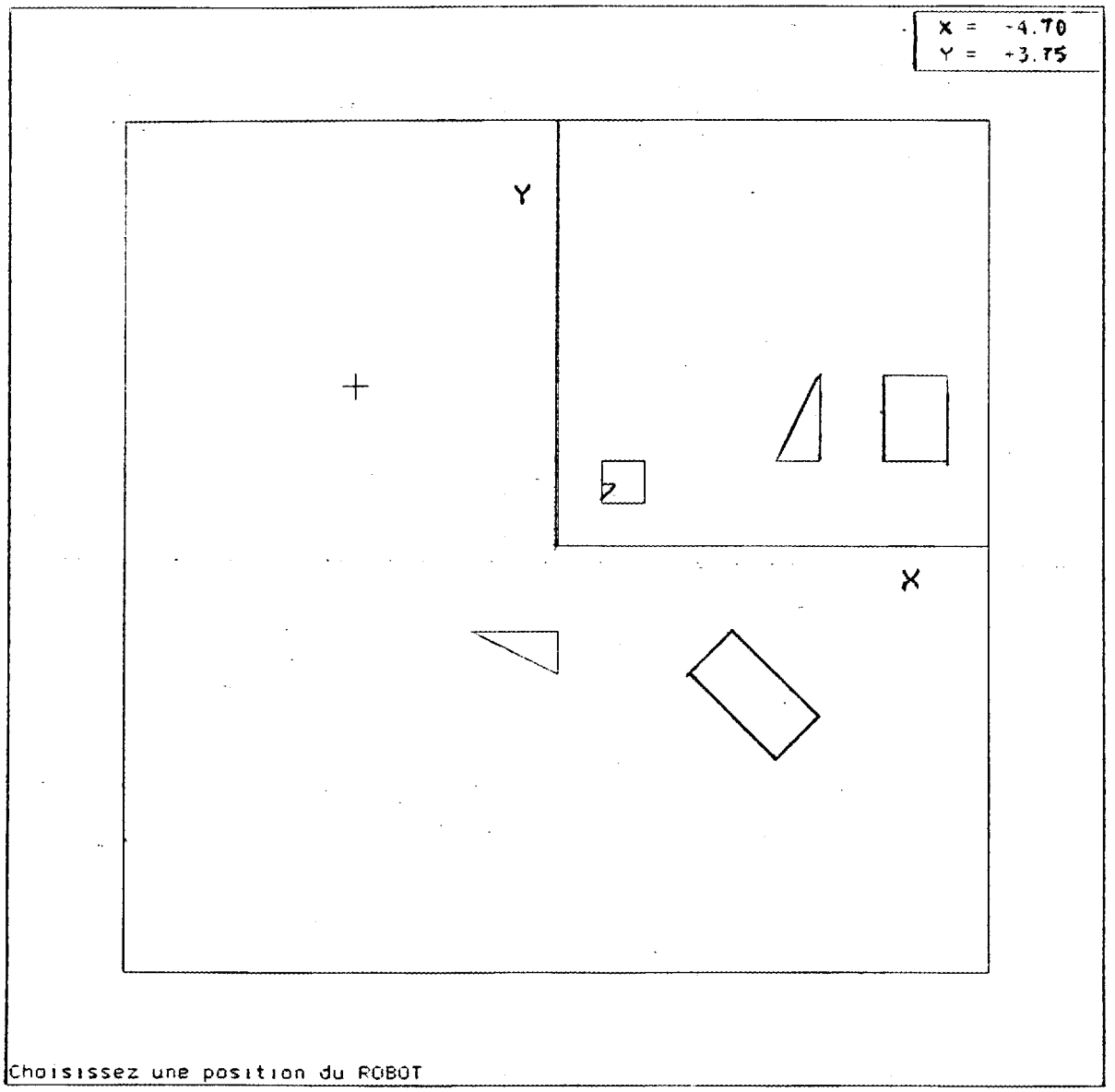
3. Implementation

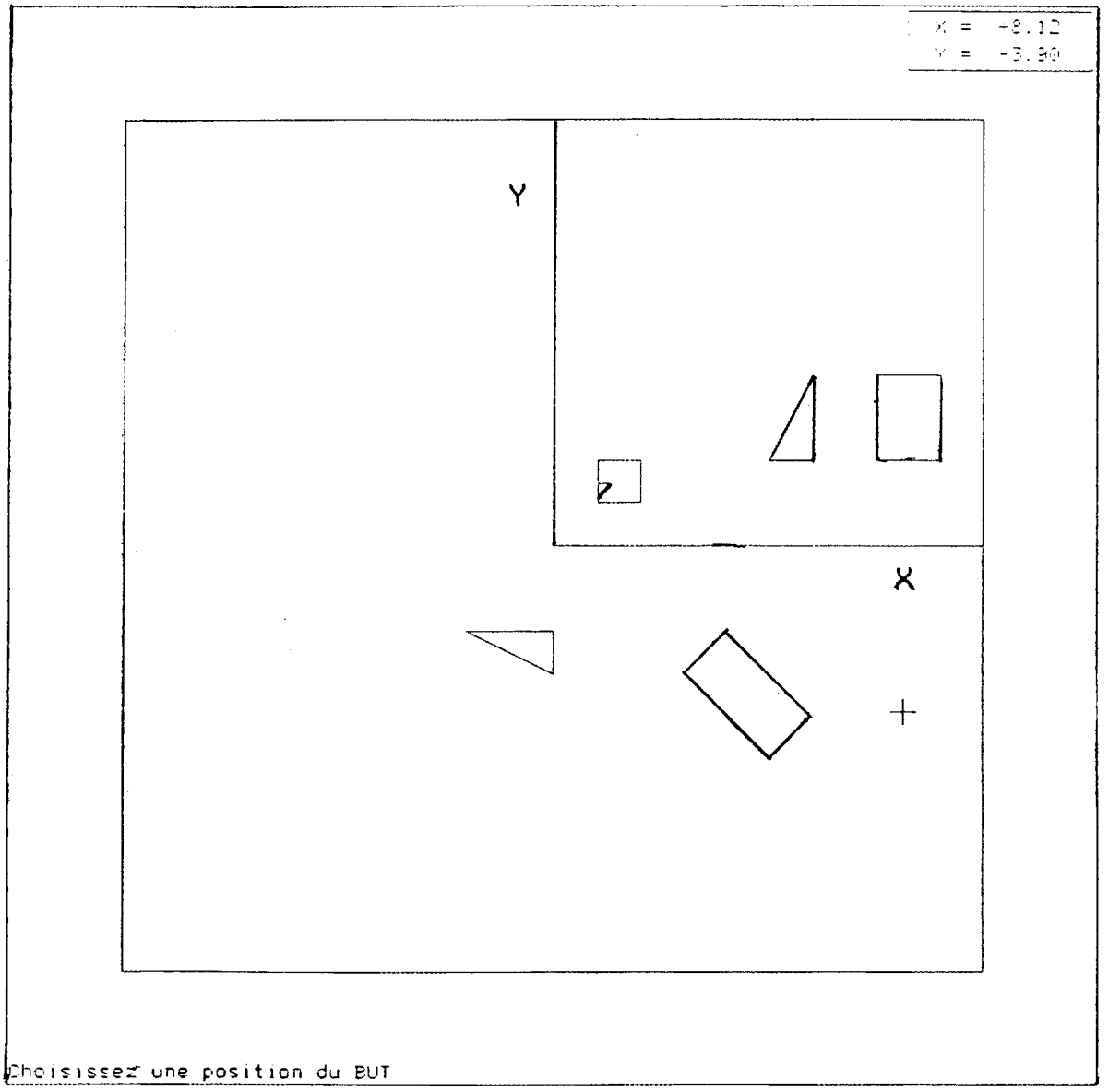
simplifications for this particular case in order to speed up computations

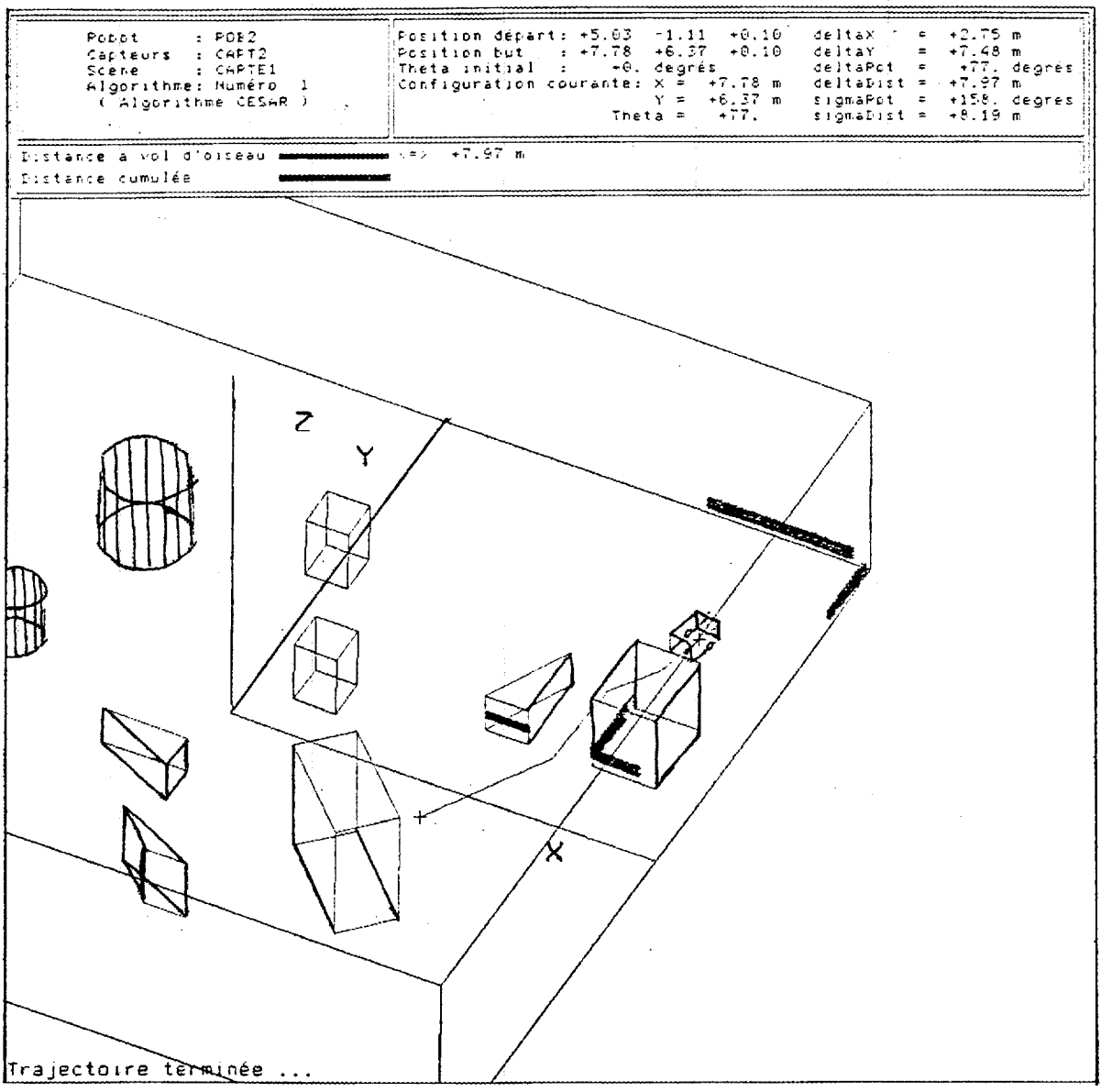
- moving in a plane
- shape invariance along horizontal sections of the robot

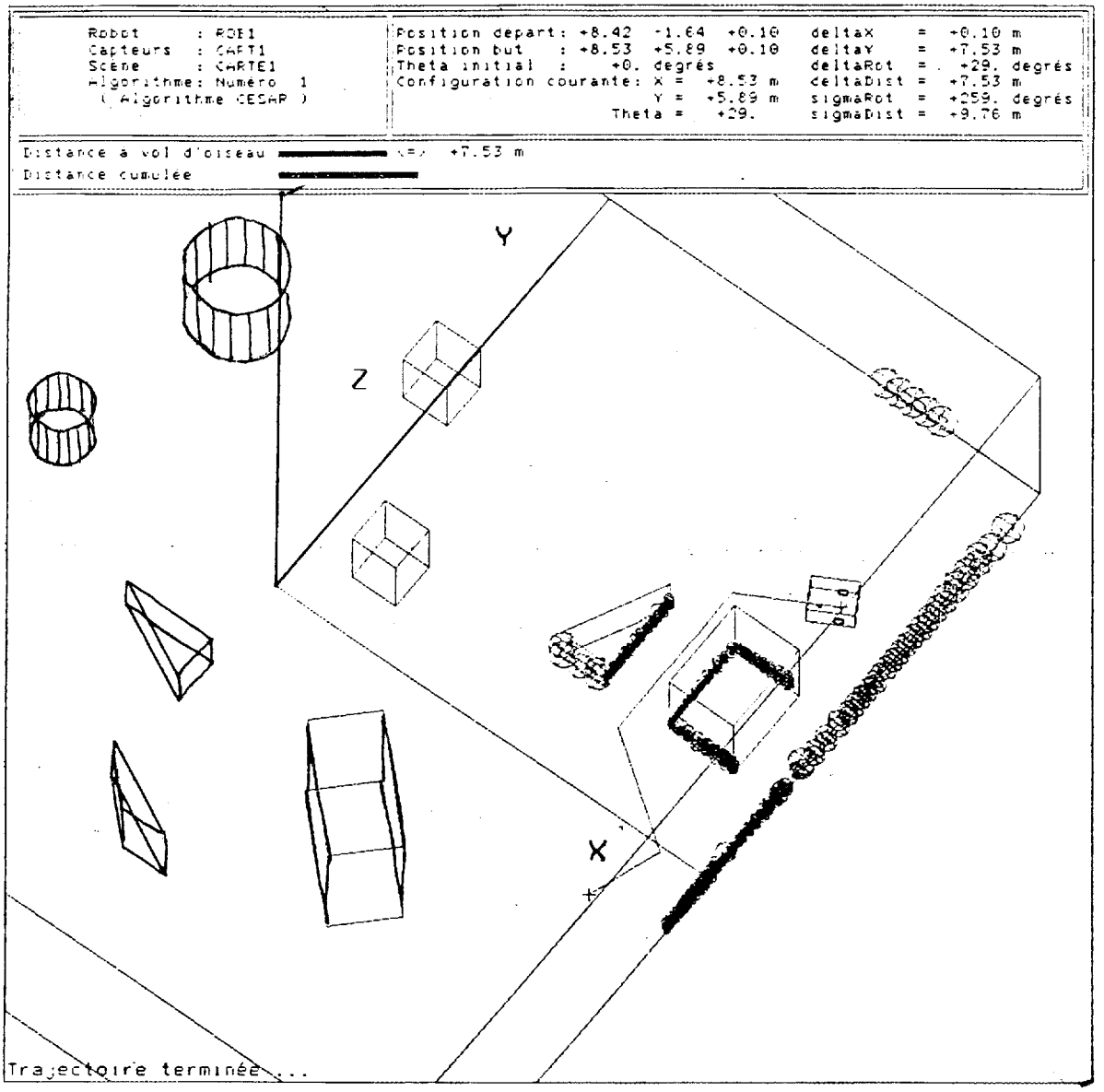
- only keep those spheres whose z-coordinate is comprised between bottom and top of the robot

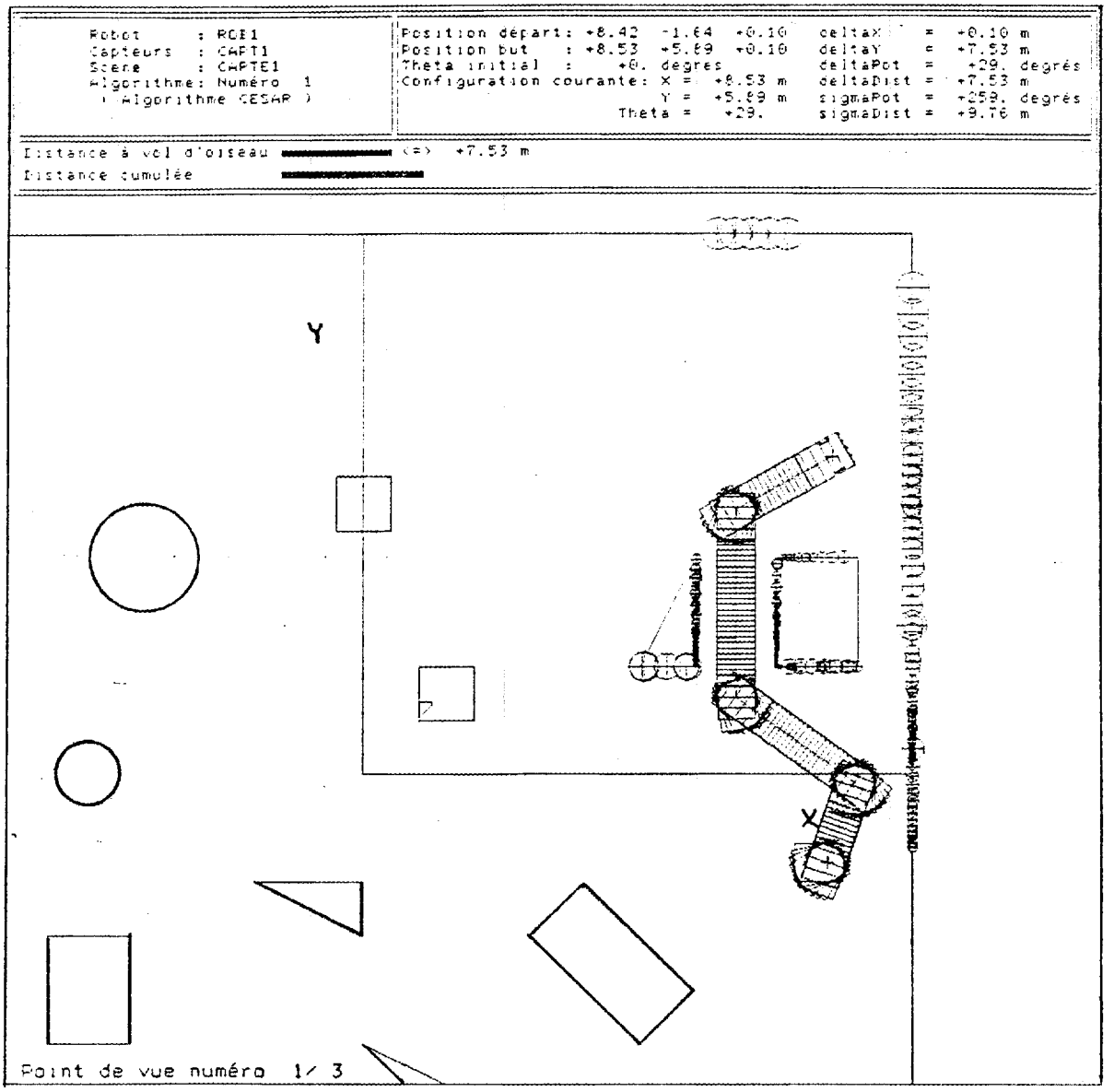
- projection onto a plane before zone sorting
- paved plane

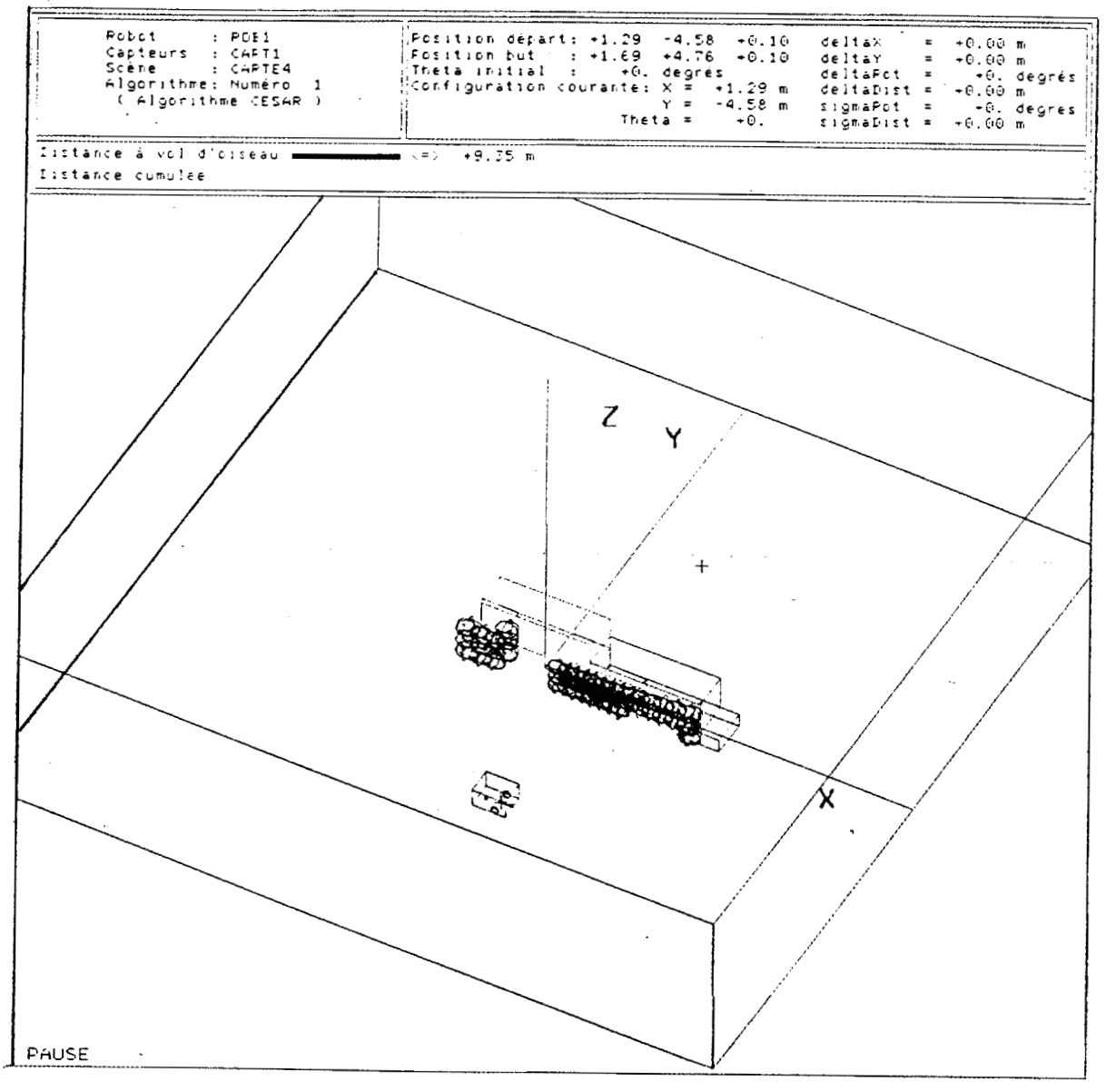

X = -4.70
Y = +3.75
Y
X
Choisissez une position du ROBOT

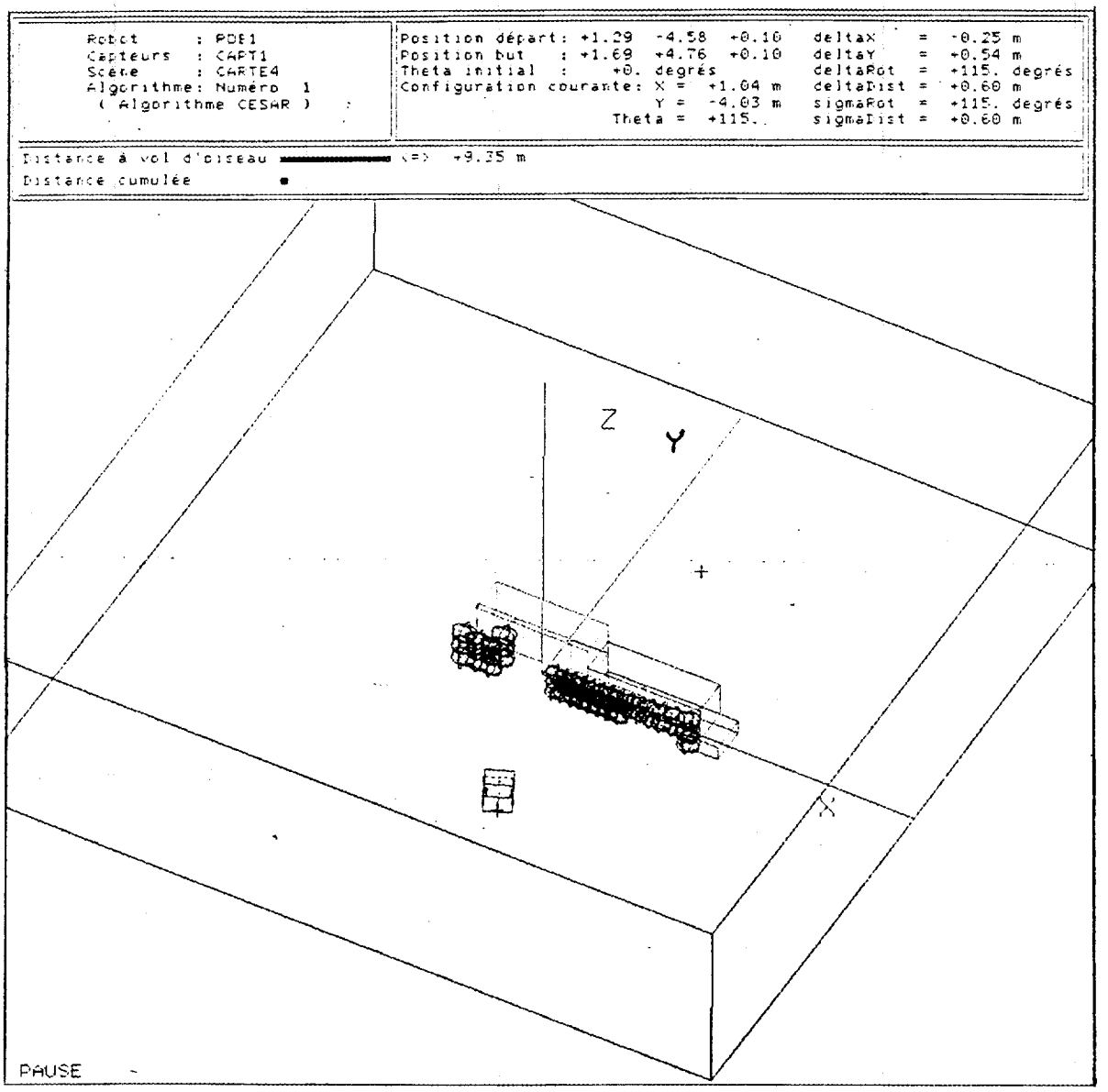

X = -8.12
Y = -3.90
Y
X
Choisissez une position du BUT

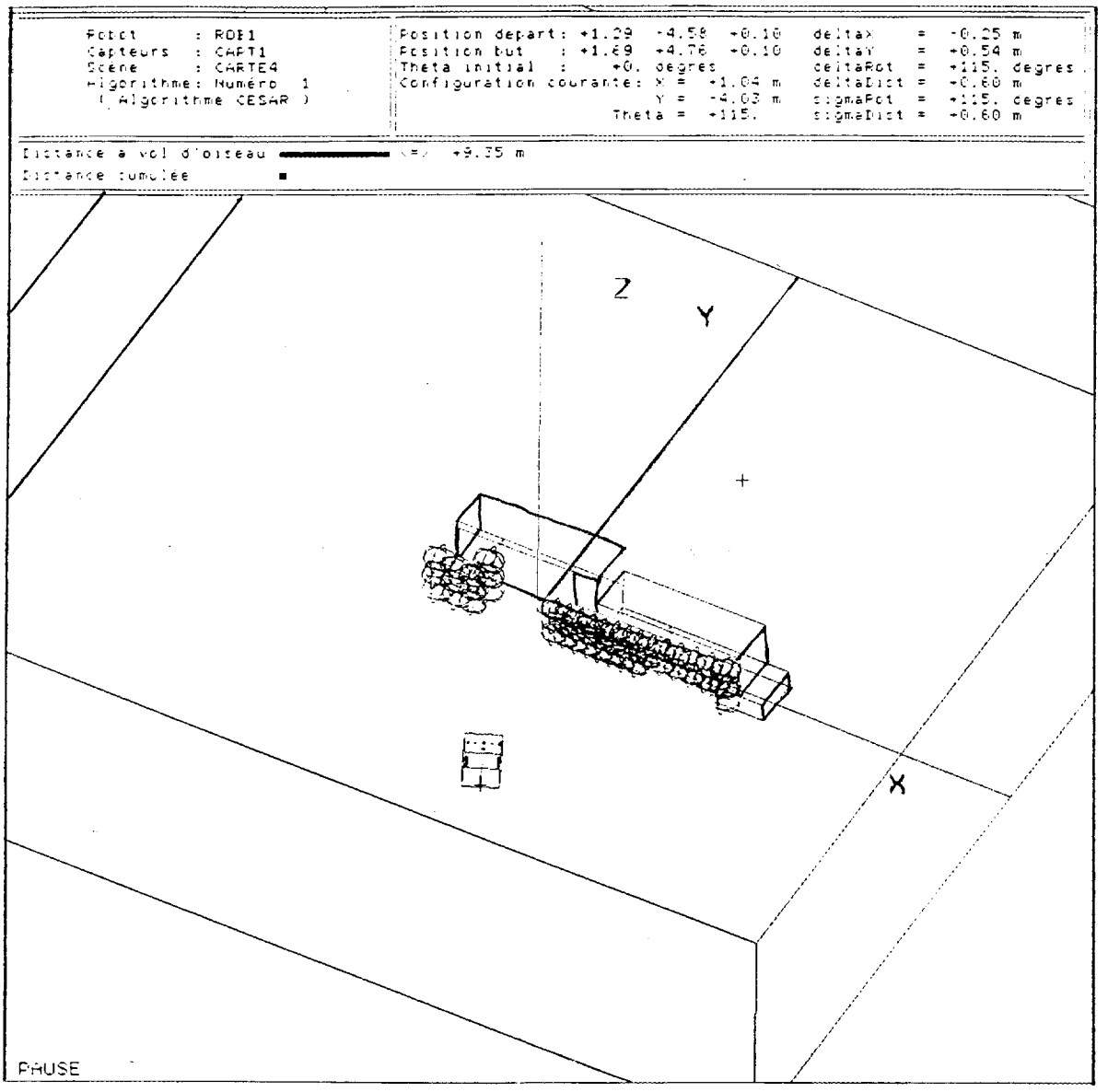

Robot : ROB2
Capteurs : CAPT2
Scene : CAPTE1
Algorithme: Numéro 1
(Algorithme CESAR)
Position départ: +5.03 -1.11 +0.10
Position but : +7.78 +6.37 +0.10
Theta initial : +0. degrés
Configuration courante: X = +7.78 m
Y = +6.37 m
Theta = +77.
deltaX = +2.75 m
deltaY = +7.48 m
deltaPot = +77. degrés
deltaDist = +7.97 m
sigmaPot = +158. degrés
sigmaDist = +8.19 m
Distance à vol d'oiseau <=> +7.97 m
Distance cumulée
Z
Y
X
Trajectoire terminée ...

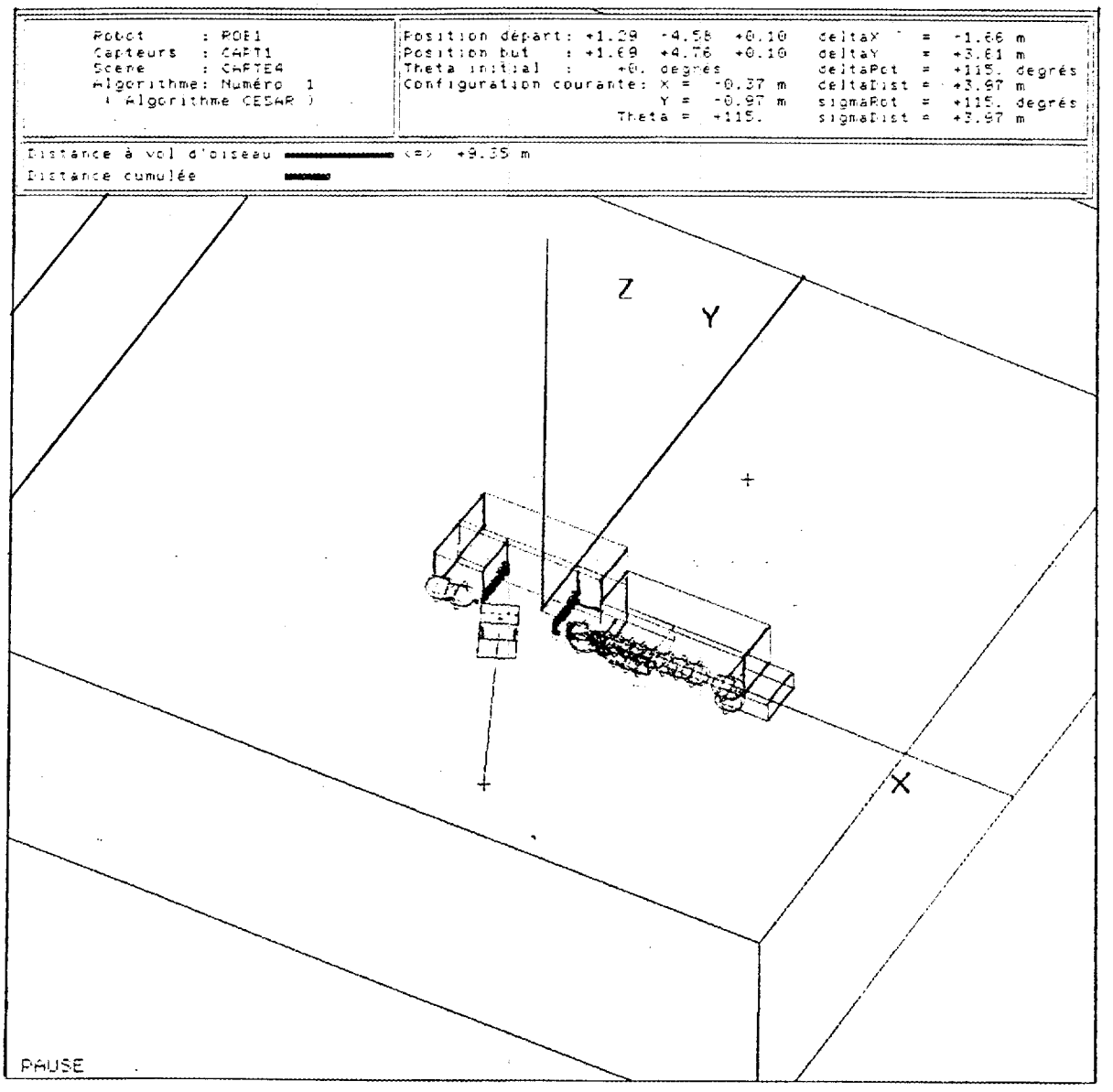

Robot : ROB1
Capteurs : CAPT1
Scene : CARTE1
Algorithme: Numéro 1
(Algorithme CESAR)
Position depart: +8.42 -1.64 +0.10
Position but : +8.53 +5.89 +0.10
Theta initial : +0. degrés
Configuration courante: X = +8.53 m
Y = +5.89 m
Theta = +29.
deltaX = +0.10 m
deltaY = +7.53 m
deltaRot = +29. degrés
deltaDist = +7.53 m
sigmaRot = +259. degrés
sigmaDist = +9.76 m
Distance à vol d'oiseau <=> +7.53 m
Distance cumulée
Y
Z
X
Trajectoire terminée ...

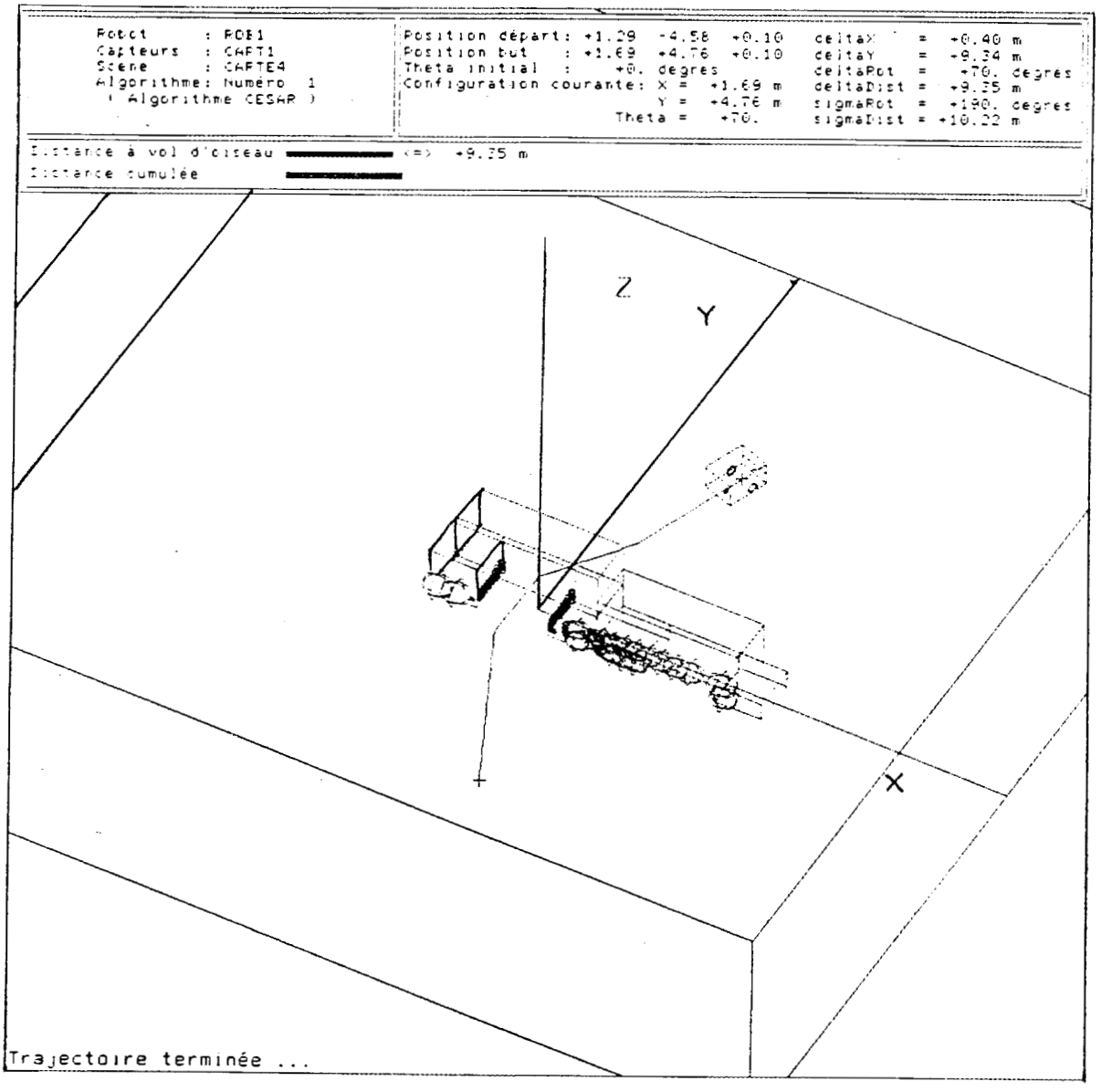

Robot : ROB1
Capteurs : CAPT1
Scene : CAPTE1
Algorithme: Numéro 1
(Algorithme CESAR)
Position départ: +8.42 -1.64 +0.10
Position but : +8.53 +5.69 +0.10
Theta initial : +0. degres
Configuration courante: X = +8.53 m
Y = +5.69 m
Theta = +29.
deltaX = +0.10 m
deltaY = +7.53 m
deltaPot = +29. degrés
deltaDist = +7.53 m
sigmaPot = +259. degrés
sigmaDist = +9.76 m
Distance à vol d'oiseau <=> +7.53 m
Distance cumulée
Y
X
Point de vue numéro 1/ 3

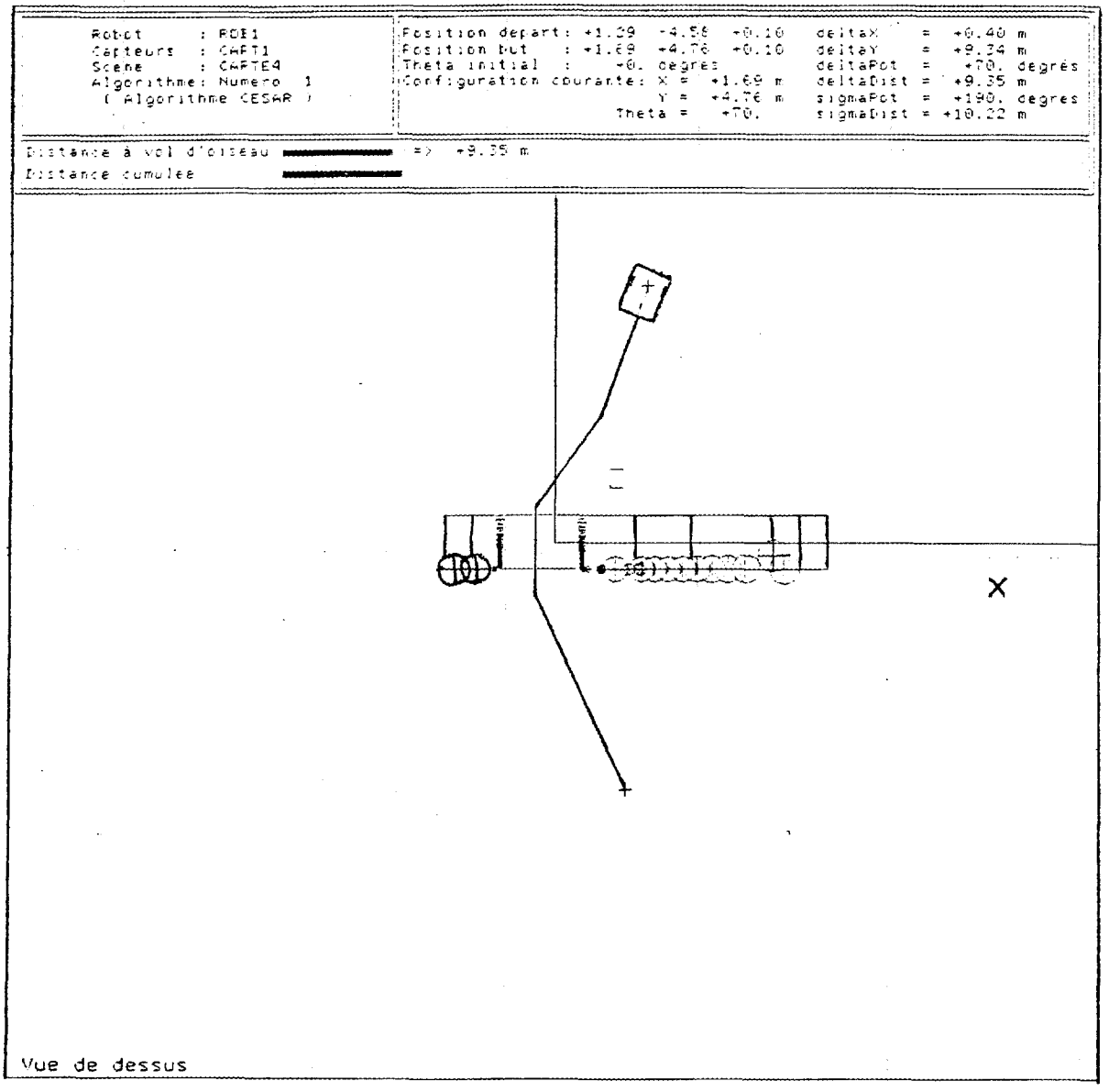

Robot : POE1
Capteurs : CAPT1
Scene : CAPTE4
Algorithme: Numéro 1
(Algorithme CESAR)
Position départ: +1.29 -4.58 +0.10
Position but : +1.69 +4.76 +0.10
Theta initial : +0. degrés
Configuration courante: X = +1.29 m
Y = -4.58 m
Theta = +0.
deltaX = +0.00 m
deltaY = +0.00 m
deltaPot = +0. degrés
deltaDist = +0.00 m
sigmaPot = +0. degrés
sigmaDist = +0.00 m
Distance à vol d'oiseau <=> +9.35 m
Distance cumulee
Z
Y
X
PAUSE

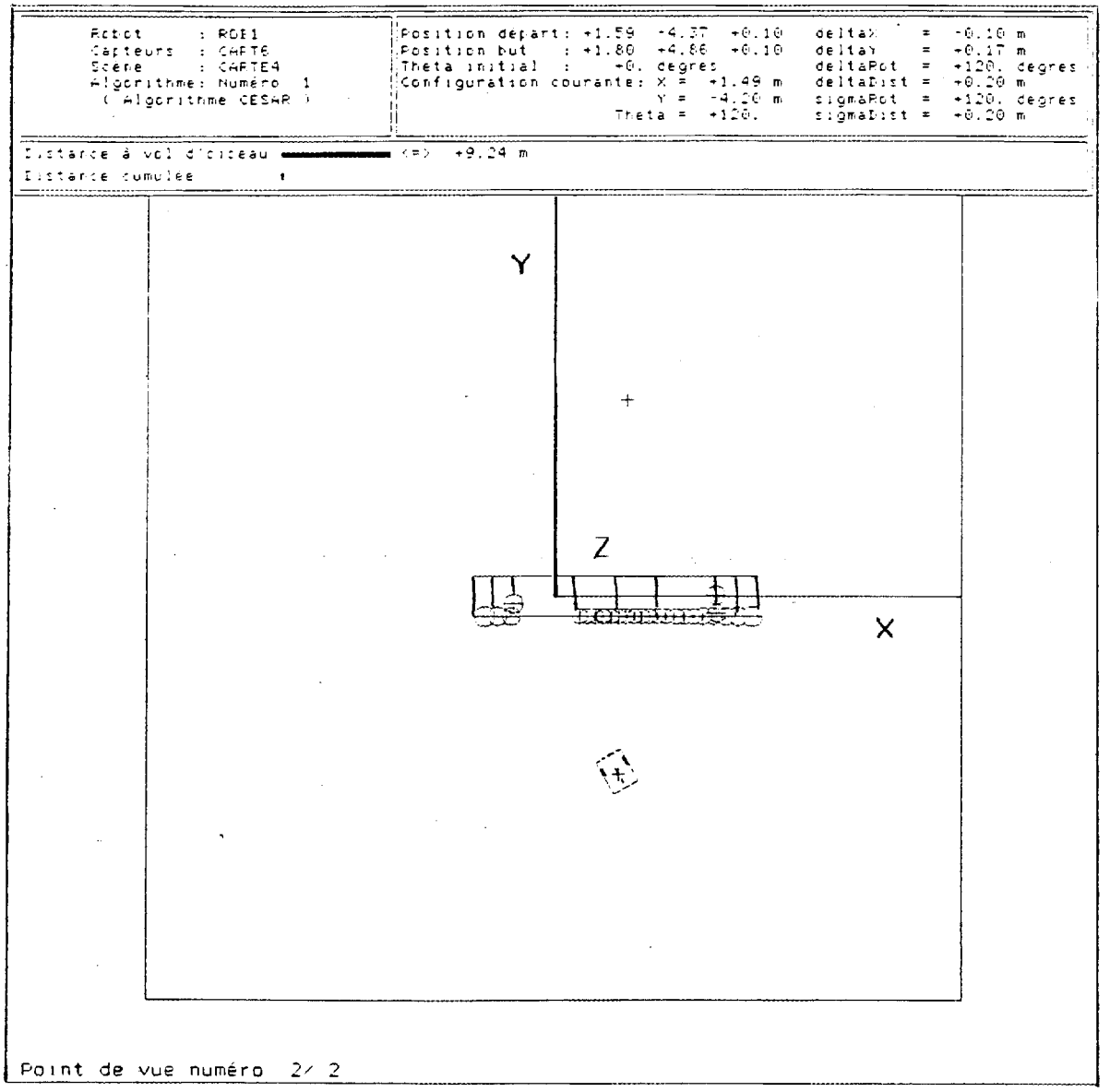

Robot : PDE1
Capteurs : CAPT1
Scène : CARTE4
Algorithme: Numéro 1
(Algorithme CESAR)
Position départ: +1.29 -4.58 +0.10
Position but : +1.69 +4.76 +0.10
Theta initial : +0. degrés
Configuration courante: X = +1.04 m
Y = -4.03 m
Theta = +115.
deltaX = -0.25 m
deltaY = +0.54 m
deltaRot = +115. degrés
deltaDist = +0.60 m
sigmaRot = +115. degrés
sigmaDist = +0.60 m
Distance à vol d'oiseau <=> +9.35 m
Distance cumulée
Z
Y
X
PAUSE

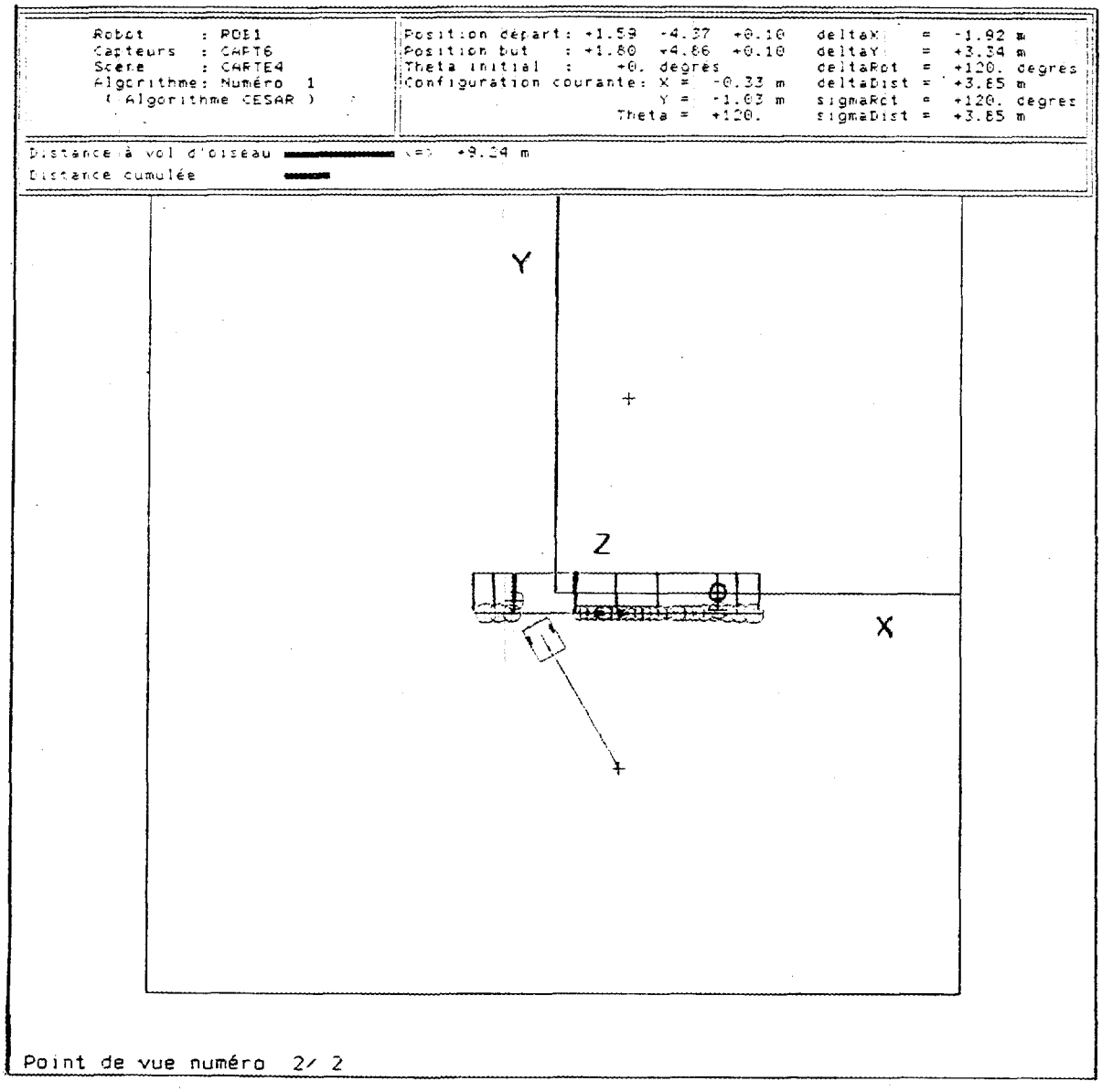

Robot : ROB1
Capteurs : CAPT1
Scene : CARTE4
Algorithme: Numéro 1
(Algorithme CESAR)
Position depart: +1.29 +4.58 +0.10
Position but : +1.69 +4.76 +0.10
Theta initial : +0. degres
Configuration courante: X = +1.04 m
Y = +4.03 m
Theta = +115.
deltaX = +0.25 m
deltaY = +0.54 m
deltaRot = +115. degres
deltaDist = +0.60 m
sigmaRot = +115. degres
sigmaDist = +0.60 m
Distance à vol d'oiseau <=> +9.35 m
Distance cumulée
Z
Y
X
PAUSE

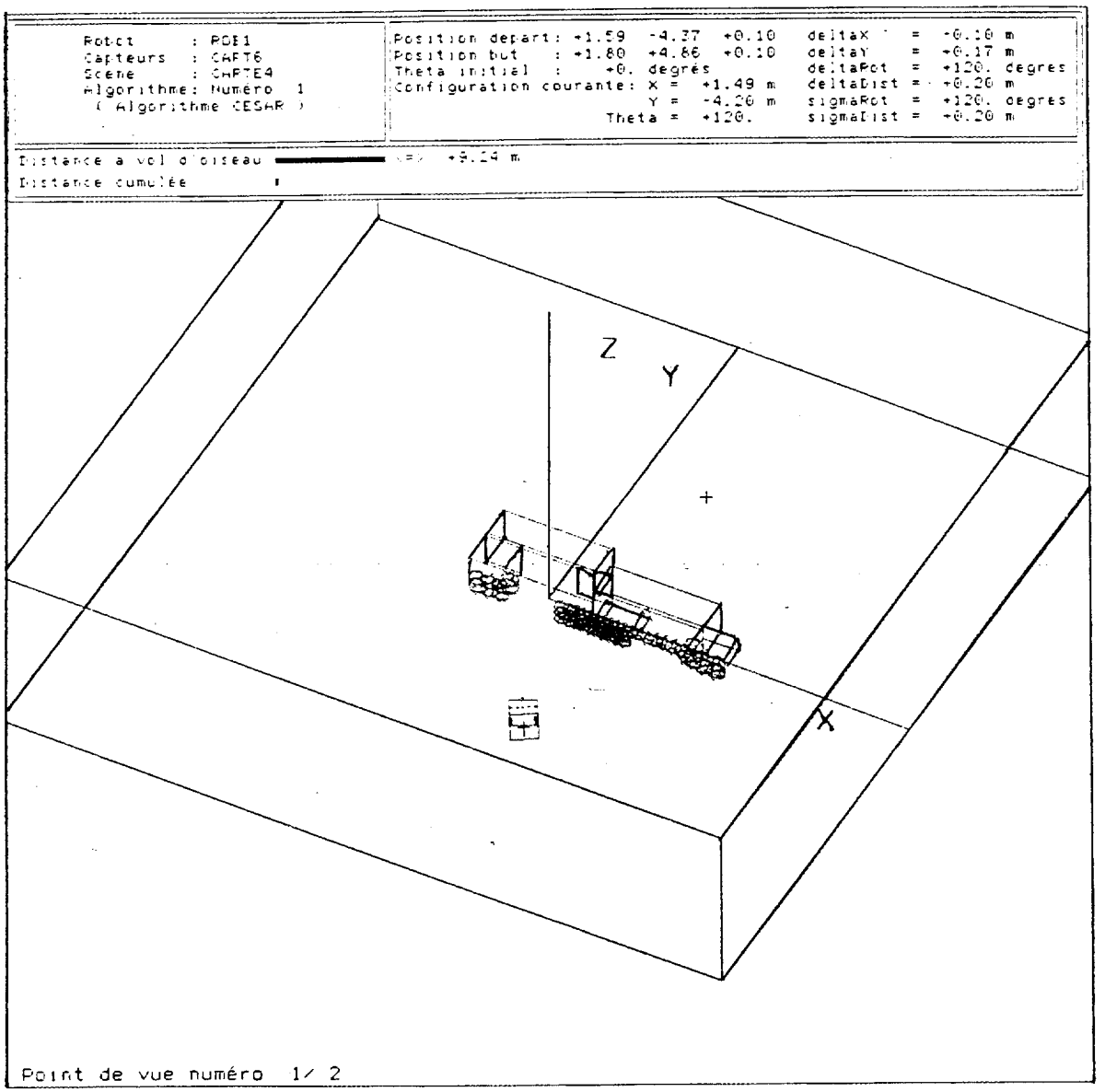

Robot : ROB1
Capteurs : CAPT1
Scene : CAPTE4
Algorithme: Numéro 1
(Algorithme CESAR)
Position départ: +1.29 -4.58 +0.10
Position but : +1.69 +4.76 +0.10
Theta initial : +0. degrés
Configuration courante: X = -0.37 m
Y = -0.97 m
Theta = +115.
deltaX = -1.66 m
deltaY = +3.61 m
deltaRot = +115. degrés
deltaDist = +3.97 m
sigmaRot = +115. degrés
sigmaDist = +3.97 m
Distance à vol d'oiseau <=> +9.35 m
Distance cumulée
Z
Y
X
PAUSE

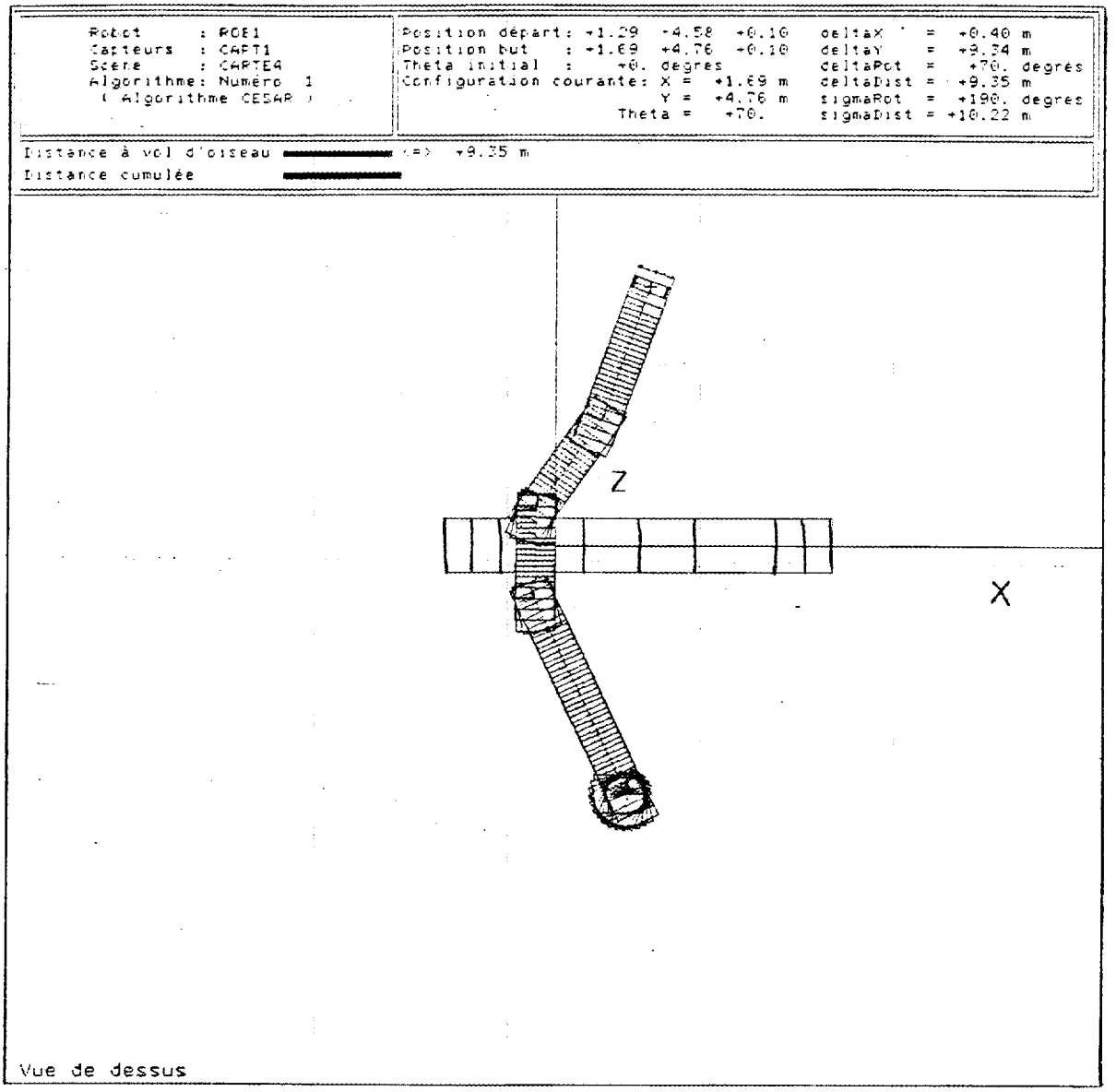

Robot : ROB1
Capteurs : CAPT1
Scene : CAPTE4
Algorithme: Numero 1
(Algorithme CESAR)
Position départ: +1.29 -4.58 +0.10
Position but : +1.69 +4.76 +0.10
Theta initial : +0. degres
Configuration courante: X = +1.69 m
Y = +4.76 m
Theta = +70.
deltaX = +0.40 m
deltaY = +9.34 m
deltaRot = +70. degres
deltaDist = +9.35 m
sigmaRot = +190. degres
sigmaDist = +10.22 m
Distance à vol d'oiseau <=> +9.35 m
Distance cumulée
Z
Y
X
Trajectoire terminée ...


Robot : ROB1
Capteurs : CAPT1
Scene : CAPTE4
Algorithme: Numero 1
(Algorithme CESAR)
Position depart: +1.29 +4.56 +0.10
Position but : +1.69 +4.76 +0.10
Theta initial : +0. degres
Configuration courante: X = +1.69 m
Y = +4.76 m
Theta = +70.
deltaX = +0.40 m
deltaY = +0.34 m
deltaPot = +70. degres
deltaDist = +0.35 m
sigmaPot = +190. degres
sigmaDist = +10.22 m
Distance à vol d'oiseau => +0.35 m
Distance cumulee
X
Vue de dessus


Robot : ROB1
Capteurs : CAPT6
Scene : CAPTE4
Algorithme: Numéro 1
(Algorithme CESAR)
Position départ: +1.59 -4.37 +0.10
Position but : +1.80 +4.86 +0.10
Theta initial : +0. degres
Configuration courante: X = +1.49 m
Y = -4.20 m
Theta = +120.
deltaX = -0.10 m
deltaY = +0.17 m
deltaRot = +120. degres
deltaDist = +0.20 m
sigmaRot = +120. degres
sigmaDist = +0.20 m
Distance à vol d'oiseau <=> +9.24 m
Distance cumulee
Y
Z
X
Point de vue numéro 2/ 2


Robot : ROB1
Capteurs : CAPT6
Scene : CARTE4
Algorithme: Numéro 1
(Algorithme CESAR)
Position départ: +1.59 -4.37 +0.10
Position but : +1.80 +4.86 +0.10
Theta initial : +0. degrés
Configuration courante: X = -0.33 m
Y = -1.03 m
Theta = +120.
deltaX = -1.92 m
deltaY = +3.34 m
deltaRot = +120. degres
deltaDist = +3.85 m
sigmaRot = +120. degres
sigmaDist = +3.85 m
Distance à vol d'oiseau <=> +9.24 m
Distance cumulée
Y
Z
X
Point de vue numéro 2/ 2


Robot : POE1
Capteurs : CAPT6
Scene : CAPTE4
Algorithme: Numéro 1
(Algorithme CESAR)
Position depart: +1.59 -4.37 +0.10
Position but : +1.80 +4.86 +0.10
Theta initial : +0. degrés
Configuration courante: X = +1.49 m
Y = -4.20 m
Theta = +120.
deltaX = -0.10 m
deltaY = +0.17 m
deltaRot = +120. degres
deltaDist = +0.20 m
sigmaRot = +120. degres
sigmaDist = +0.20 m
Distance a vol d oiseau
Distance cumulée
Z
Y
X
Point de vue numéro 1/ 2


Robot : ROB1
Capteurs : CAPT1
Scene : CARTE4
Algorithme: Numero 1
(Algorithme CESAR)
Position départ: +1.29 +4.58 +0.10
Position but : +1.69 +4.76 +0.10
Theta initial : +0. degres
Configuration courante: X = +1.69 m
Y = +4.76 m
Theta = +70.
deltaX = +0.40 m
deltaY = +9.34 m
deltaRot = +70. degres
deltaDist = +9.35 m
sigmaRot = +190. degres
sigmaDist = +10.22 m
Distance à vol d'oiseau <=> +9.35 m
Distance cumulée
Z
X
Vue de dessus

DEMT / SYST / MRS

4. Conclusion

navigation remains limited

- one pb solved = two arches
- one pb raised = robot shape
  - ↳ loose obstacle avoidance ?
  - ↳ more global geometrical approach.

- go further in world modeling using C.G.....

DEMT / SYST / MRS

4. Conclusion

- toward a testbed for alg. study and analysis
- importance of programming interface (alg. should be as reusable and transportable as possible)
- simulation must be qualitative (graphics) <u>and</u> quantitative
- interesting to test heuristics before any integration in a real context

Navigation Algorithms For Hermies

HERMIES is a research tool.

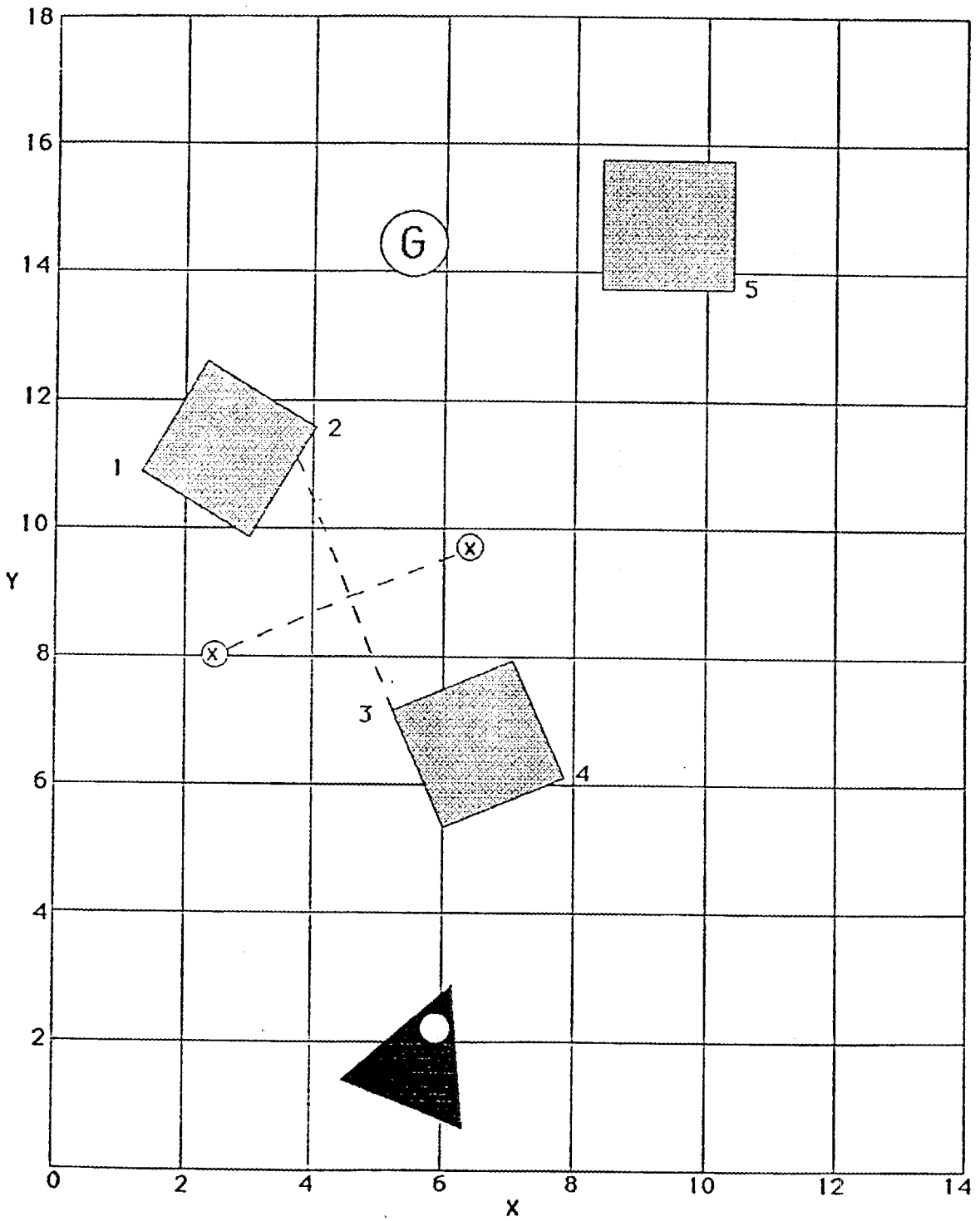
- Navigation algorithms are frequently modified to correct problems detected experimentally.

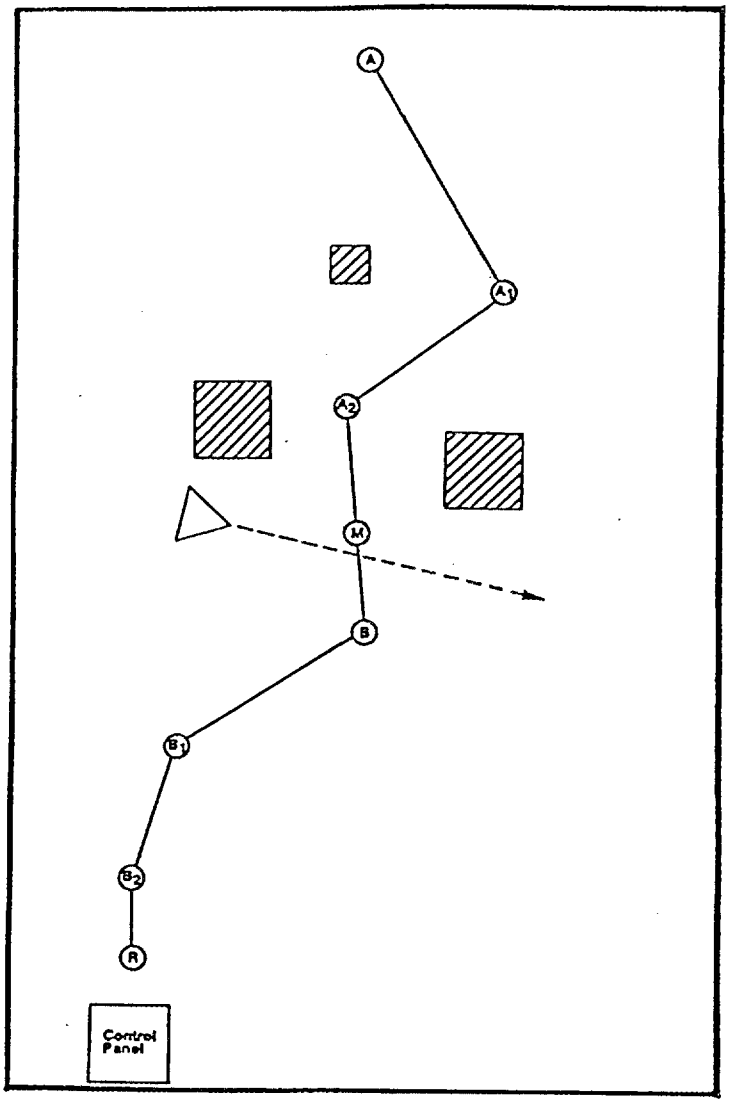
Algorithms consist of:

- Production Rules in CLIPS Expert System Shell.
- Procedures coded in C.
- Primitives actuating effectors, receiving sensor data.

Present Approach:

- Assumes unknown, dynamic environment.
- Is given initial location and angle, and goal location.
- Proceeds to goal by dead-reckoning.
- Uses sonars for obstacle avoidance.
- Does a 360° scan by 3° increments.
- Analyzes sonar data to find corridors.


18
16
14
12
10
8
6
4
2
0
2
4
6
8
10
12
14
Y
X
G
1
2
3
4
5


A
A1
A2
M
B
B1
B2
R
Control Panel

Future Developments:

- Sensor fusion.
- Different modules for different problems.
- Artificial potential fields technique of Khatib.

leti

Mobile Robot Navigation based on Geometrical Approach

FAVRE P., FANTON J.C.
CEA / D.LETI / SETIA
Grenoble - FRANCE

Concerns :

- Local path planning among obstacles in a 2D environment (no vertical consideration)

- Command generation to follow the previous path

In a known or not known environment.

With constraints :

- Kinematic (acceleration, maximum speed)

- Dynamic (centrifugal force limitation)

- Geometrical (robot shape, turning radius limitation)

© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

Three approaches are presented, depending on knowledge levels of the environment, the precision and the execution speed required.

A. Generation and following of a trajectory with given path points.

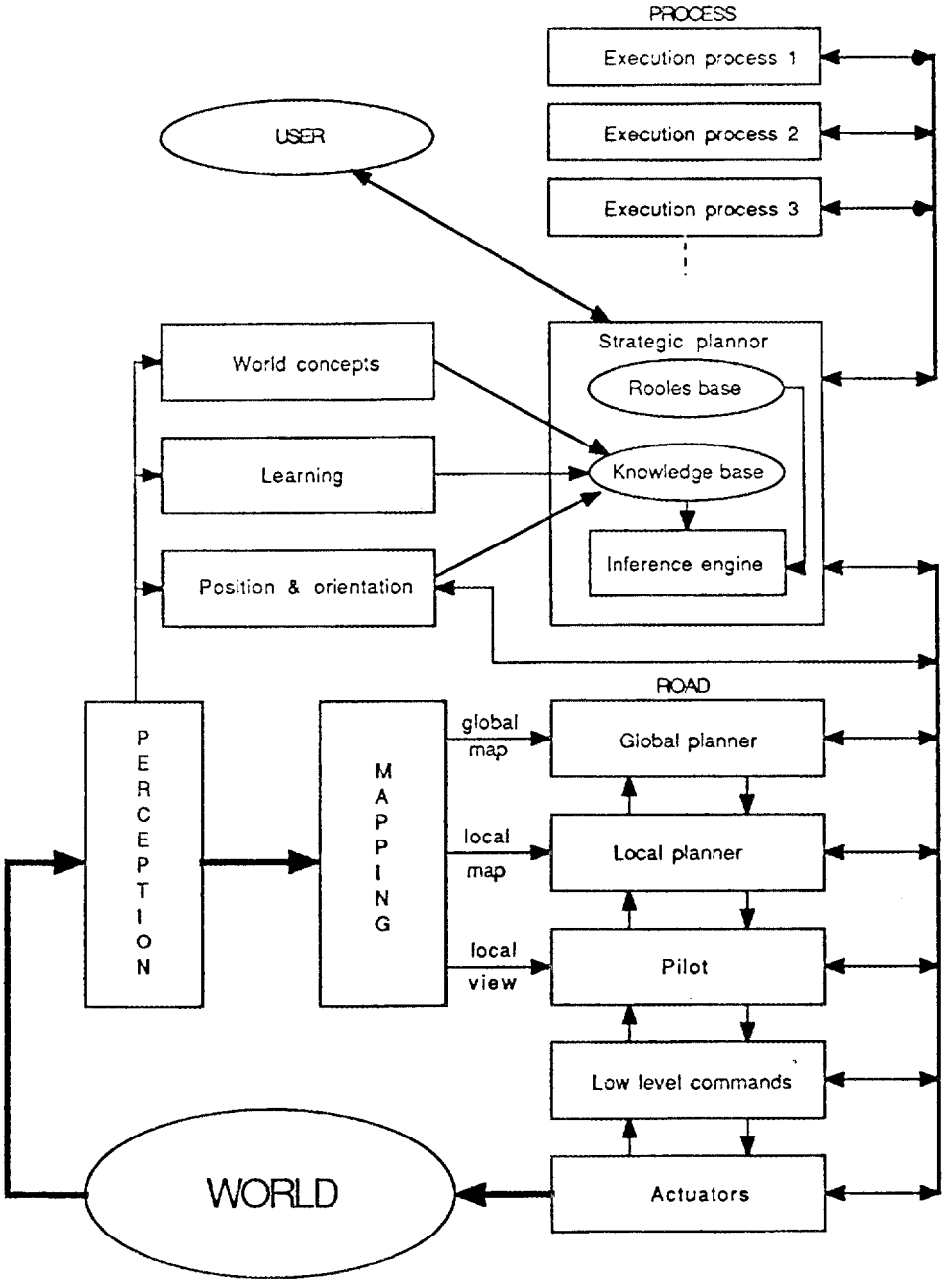
Suitable if the environment is well known or for teleoperation applications

B. Position planning and real time obstacles avoidance

Suitable if the robot has to reach a goal as quickly as possible, in a safe way, and in a not well known environment.

C. Posture planning (Posture = position + heading)

For applications requiring a large precision as docking and manoeuvering in a constraint space. Main constraints of a mobile robot are introduced in the posture planner.



© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

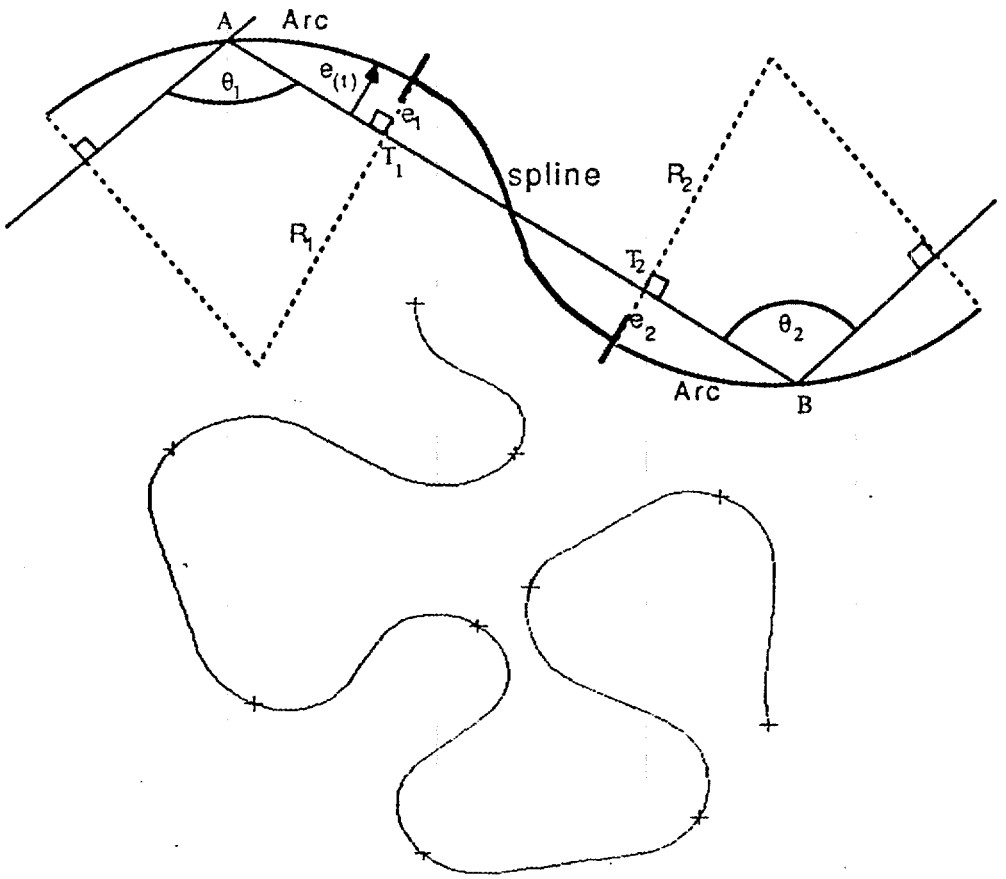
leti

Trajectories generation

The idea is to generate a trajectory which includes given path points.

The chosen fonctions allow the control of the curvature radius and of the path location.

The trajectory is built as a succession of arcs of circle and of spline fonctions (third order polynoms).



leti

Speed planning

The trajectory is discretized => set of points

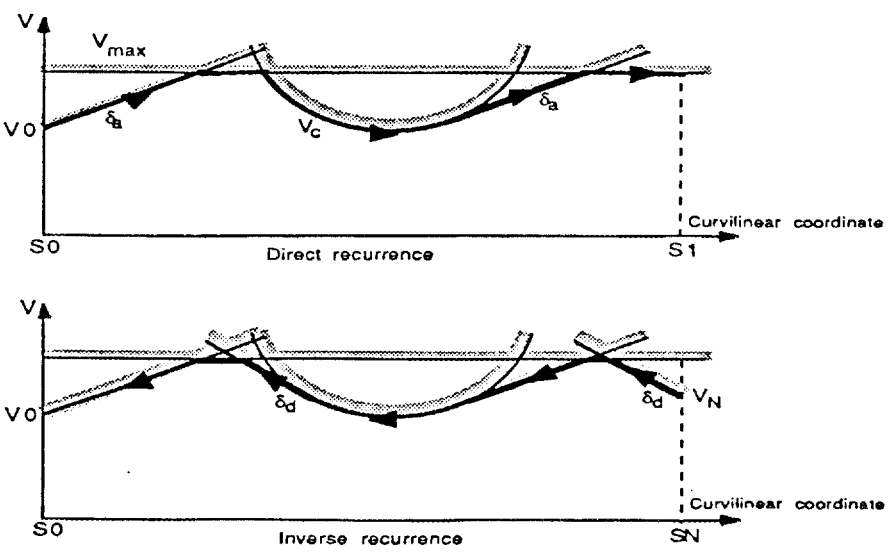
A speed is associated with each point with respect to the kinematic constraints.

Let (x_n) be the set of points of the trajectory.

Two recursive algorithms perform the speed planning :

- Direct recurrence
  Computes :
  - Initial speed
  - Acceleration limitation
  - Centrifugal force limitation
  - Maximum speed

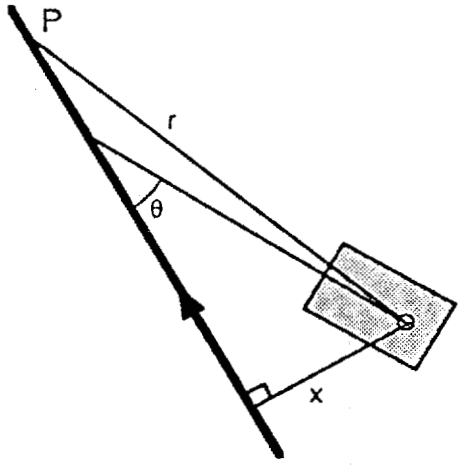
- Inverse recurrence
  Computes :
  - Final speed
  - Deceleration limitation



leti

Following

"Direct pursuit" controller of a point P from a fixed distance from the robot.



θ : Current robot heading.

x : Shortest distance between the robot's location and the path.

α : Difference in orientation between the current heading and the heading toward P.

Steering fonction : $$\Omega = 4.V.\frac{x - r.\sin(\theta)}{r^2.\cos(\theta)}$$

© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

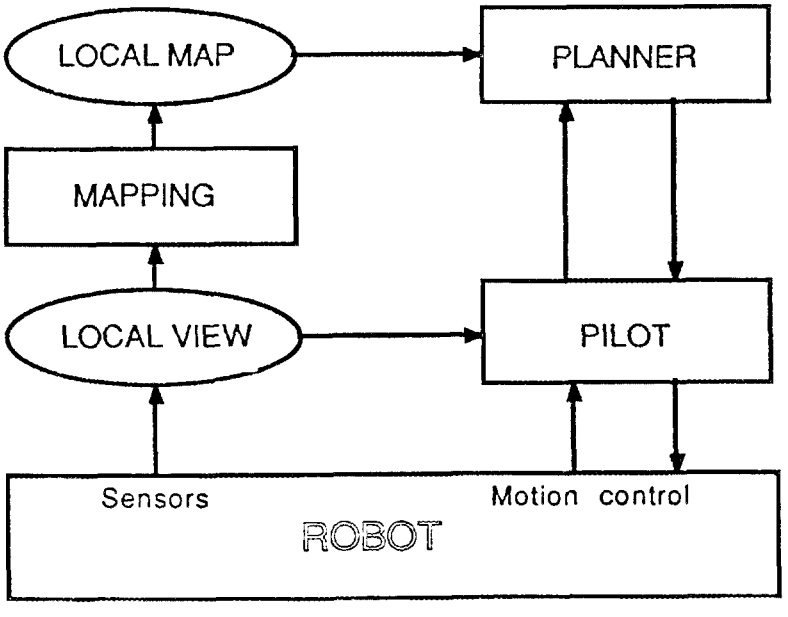
leti

B. POSITION PLANNING AND OBSTACLE AVOIDANCE

The basic idea is the cooperation of two main modules :

- A position planner :
  Using a local map, it finds the path to reach a goal.

- A pilot :
  Using the local view (almost raw sensor's data), it avoids obstacles.
  Using the information from the planner, it drives the robot to the goal.



Navigator architecture

leti

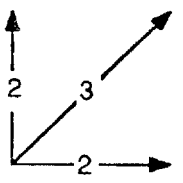
Position planner

Modeling : A grid model is used .

- Easy updating.
- Ability of implementation on parallel architecture.

Propagation algorithm derived from Lee's one :

A cost is propagated from the goal all over the free space.



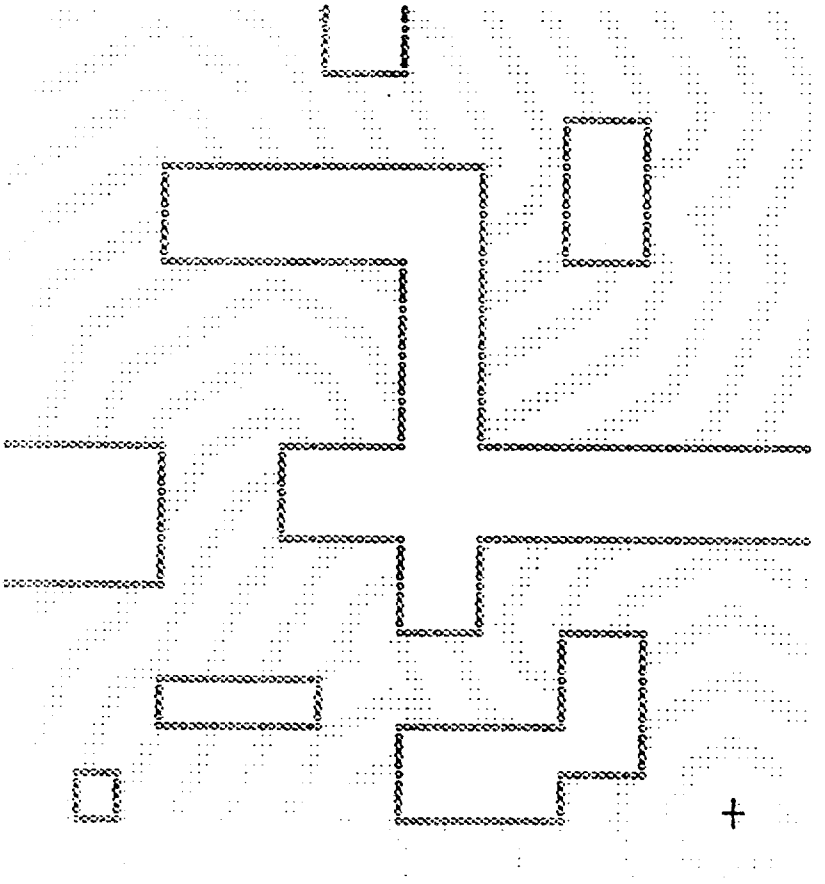
				6					
		6	5	4	5	6			
		5	3	2	3	5			
	6	4	2	0	2	4	6		
		5	3	2	3	5			
		6	5	4	5	6			
				6					

Eight propagation directions

Simulations on a PC 386 : 110 ms for a grid of 100x100 cells.

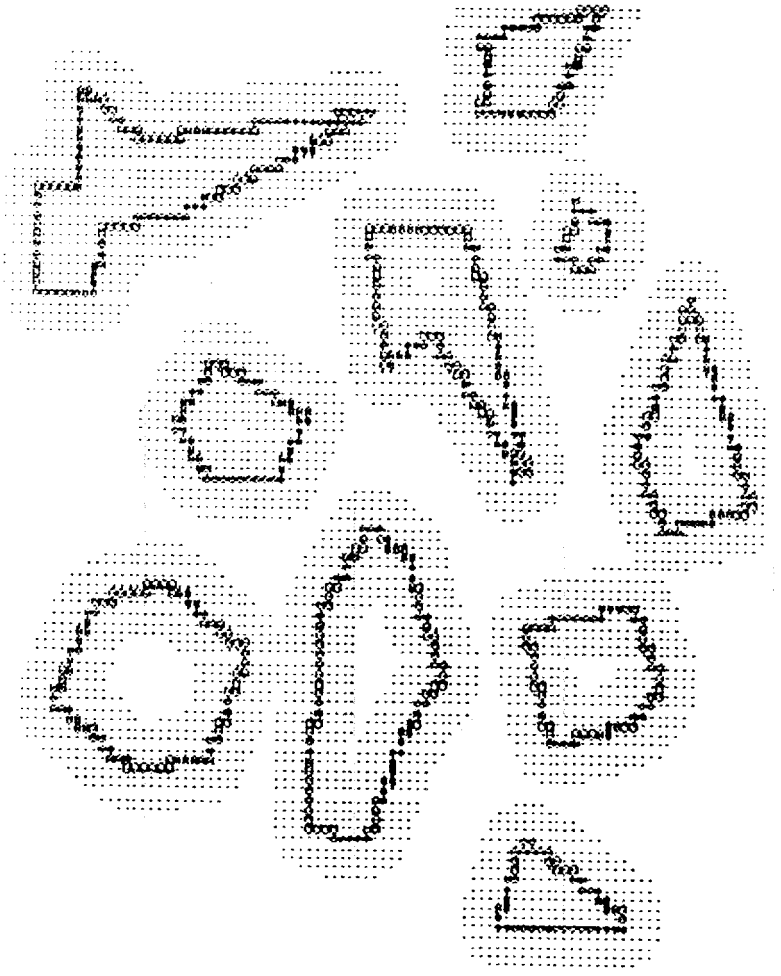
To take into account the robot shape, obstacles are grown...

leti



© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

leti

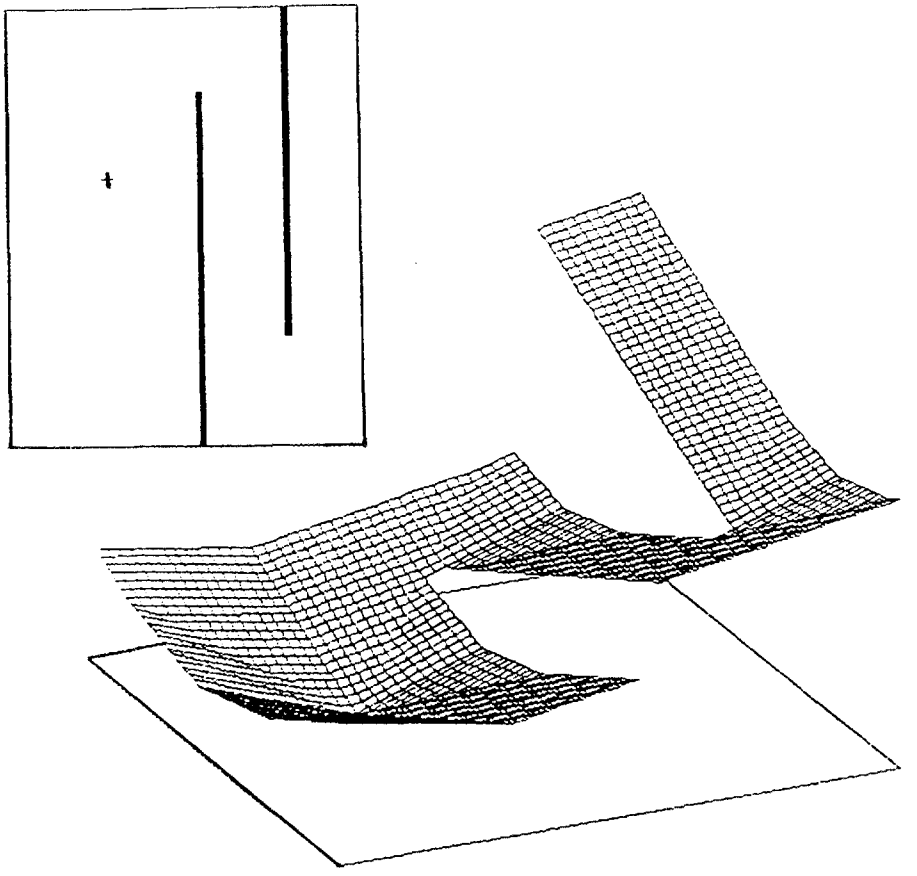


© cea / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

leti

Link between the planner and the pilot

Property beeing used : After performing a propagation, from every cell of the grid, it's possible to compute a free path to the goal.



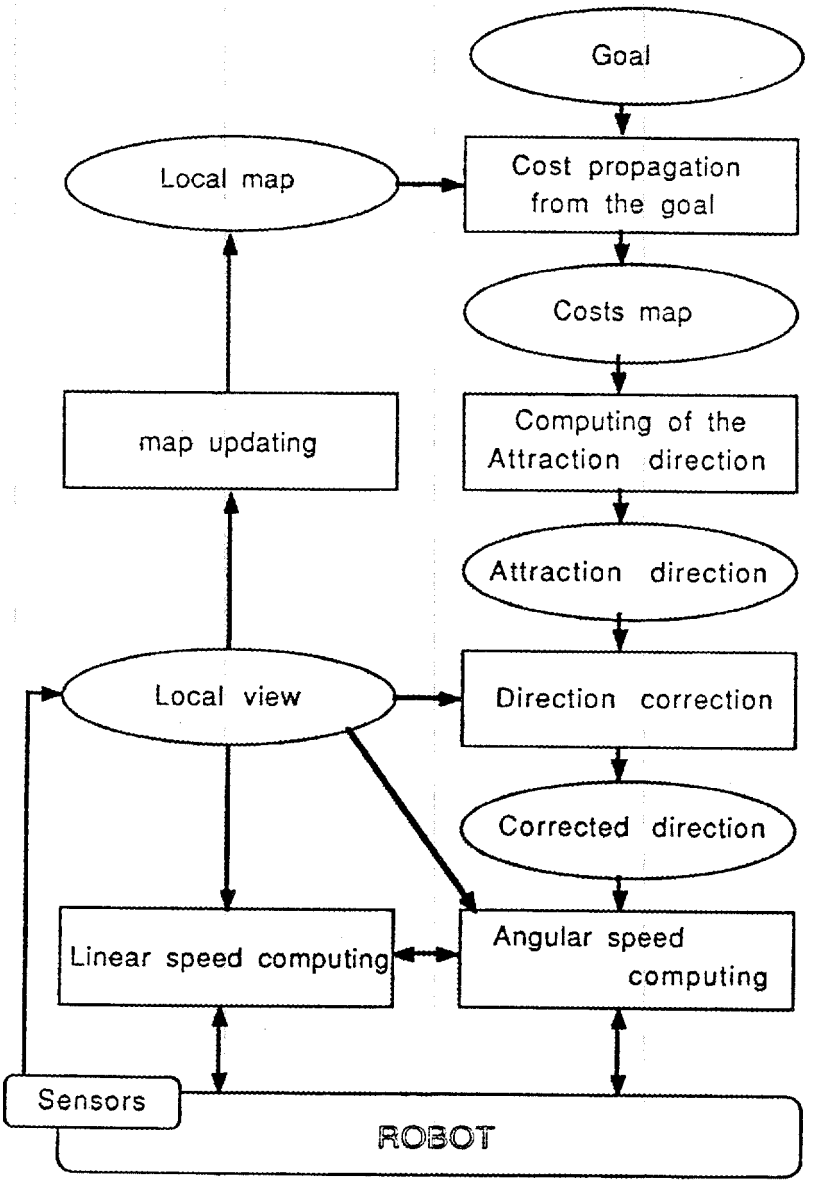
=> Each time the pilot needs it, an "attraction direction" is computed from the current robot's position.

That direction correspond to the straight line which has the "deepest slope" in the map of costs.

© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

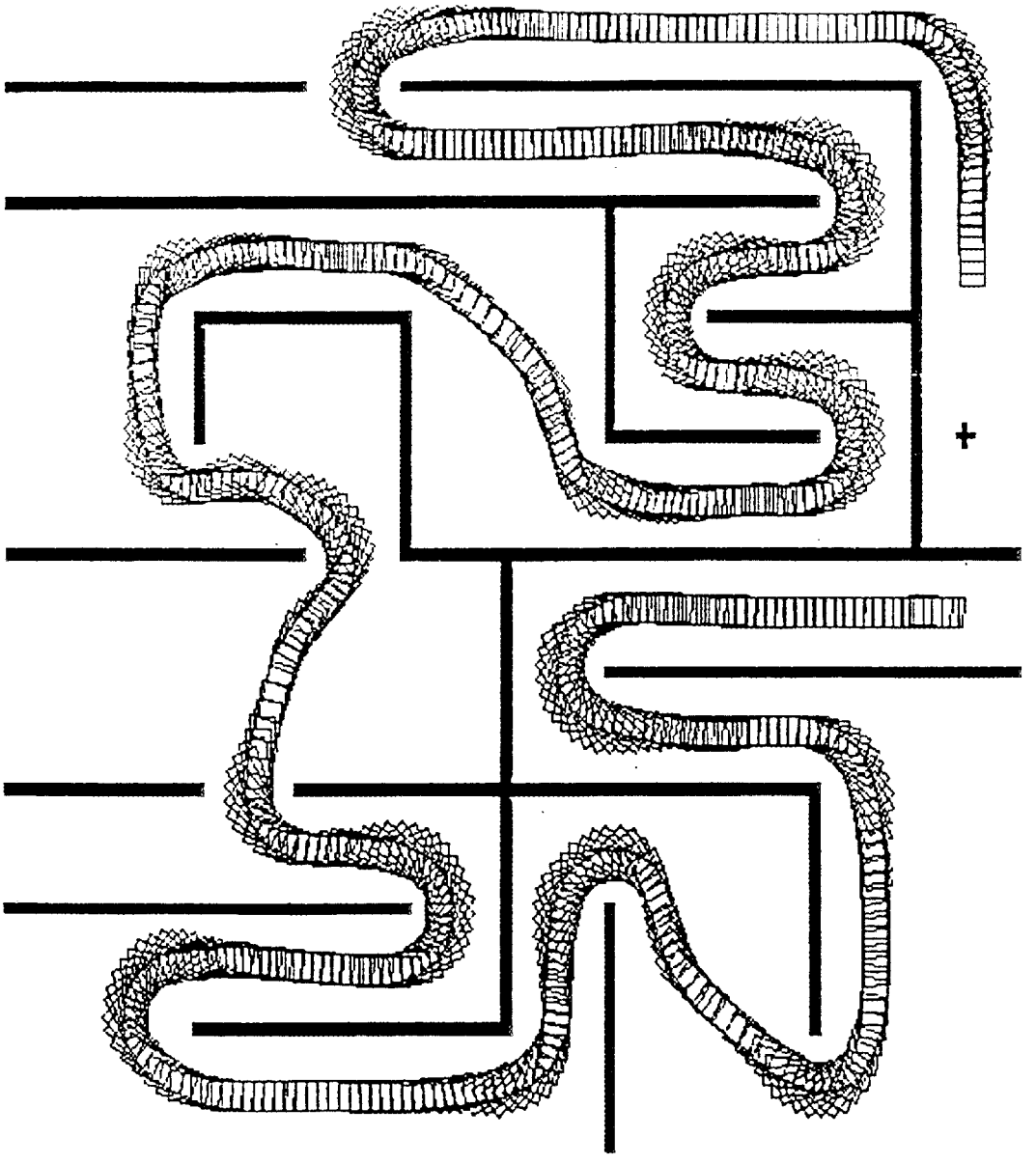
leti

Local navigation architecture



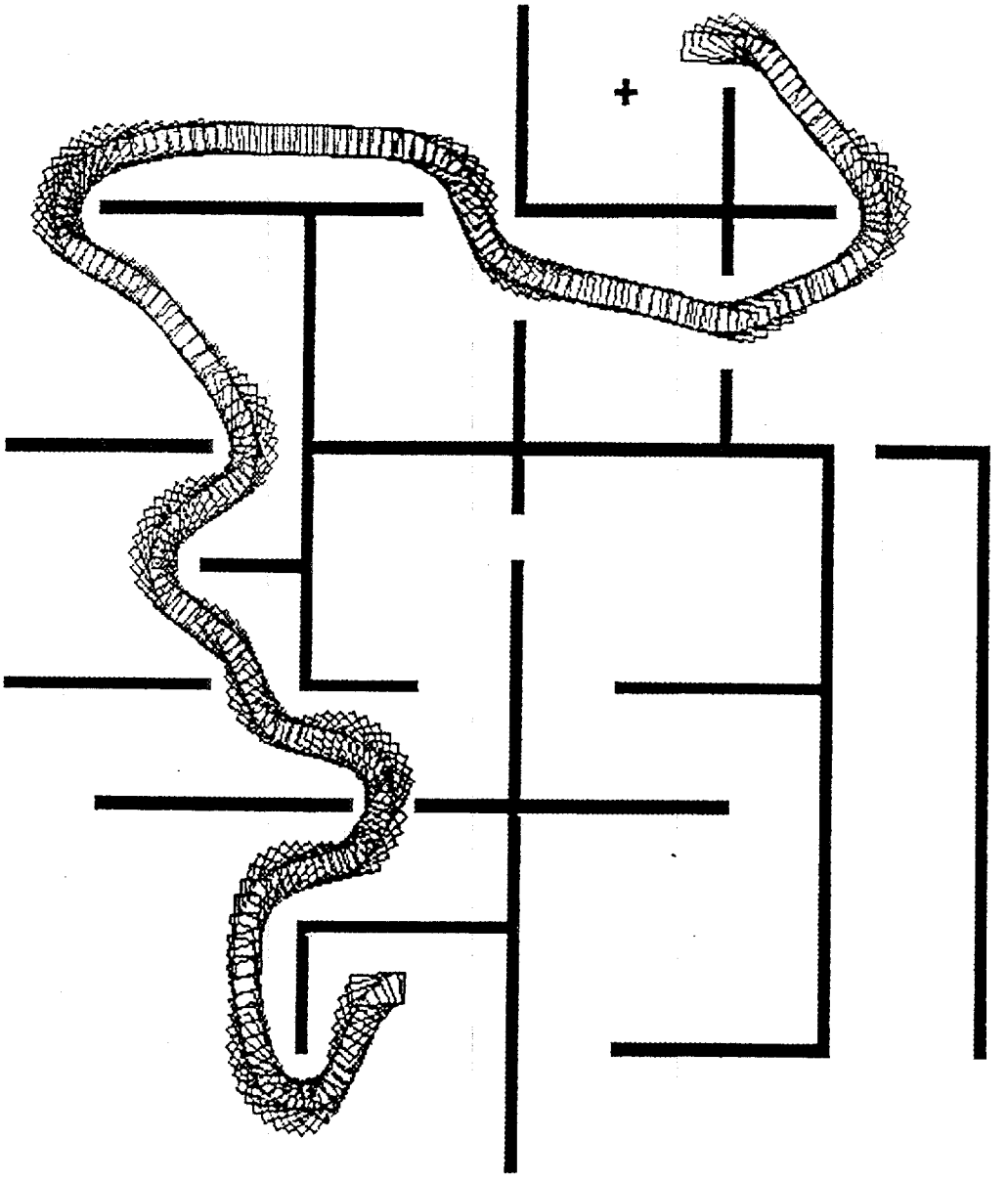
© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

leti



© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

LETI



© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

leti

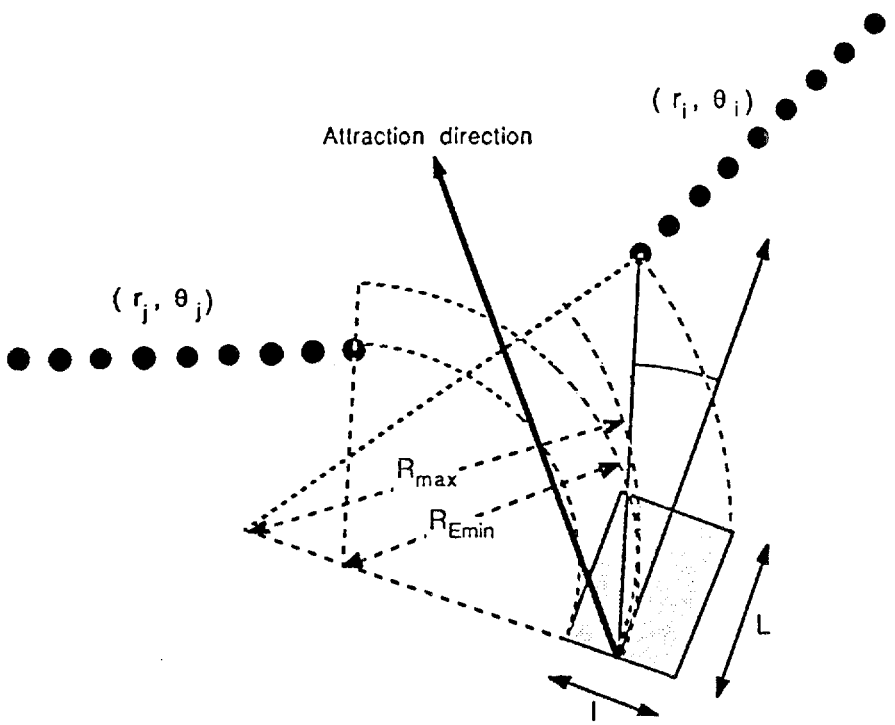
The pilot

The attraction direction from the planner provides the knowledge of the way to reach the goal.

Updating the local map and computing a new cost map need too much time for a fast robot to avoid obstacles.

=> The pilot will use the **local view** to perform the commands for the actuators to avoid obstacles with respect of kinematic, dynamic and geometrical constraints.

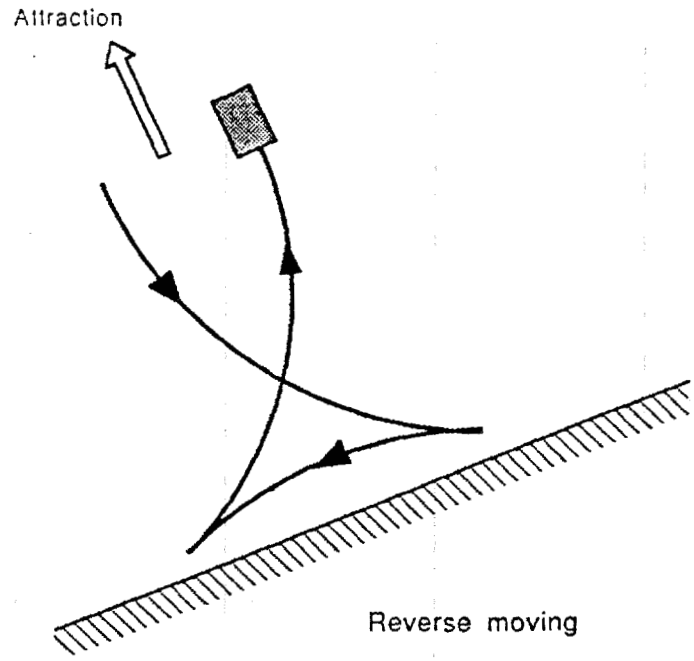
A geometrical method is used (turning radius consideration).



© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATIO'

leti

Reverse motion



The method is heuristic. It's easy to introduce a reverse motion.

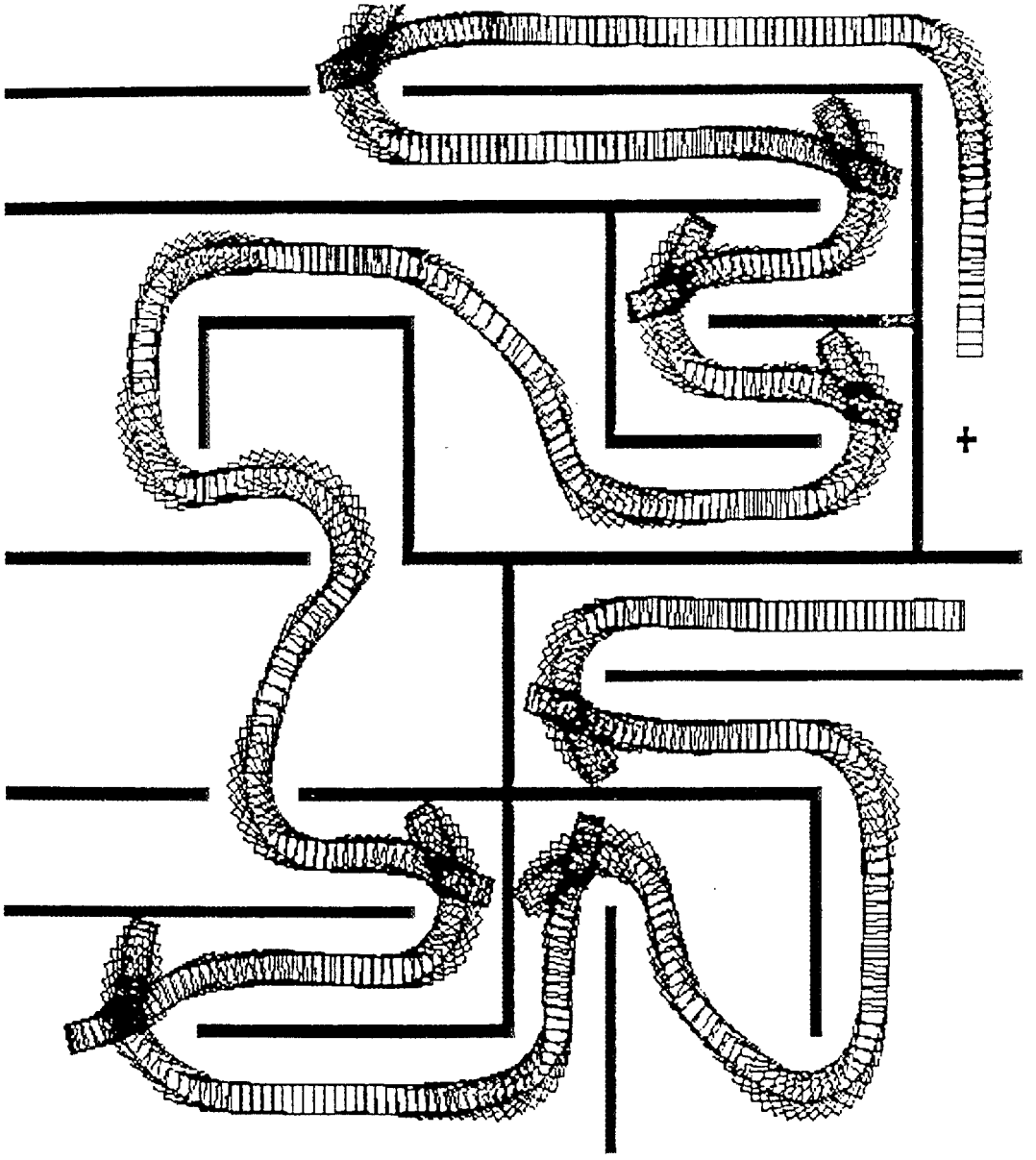
If the robot can't turn because of its turning radius limitation :

- A speed limitation near obstacles make it stop.

- The backward motion is activated.

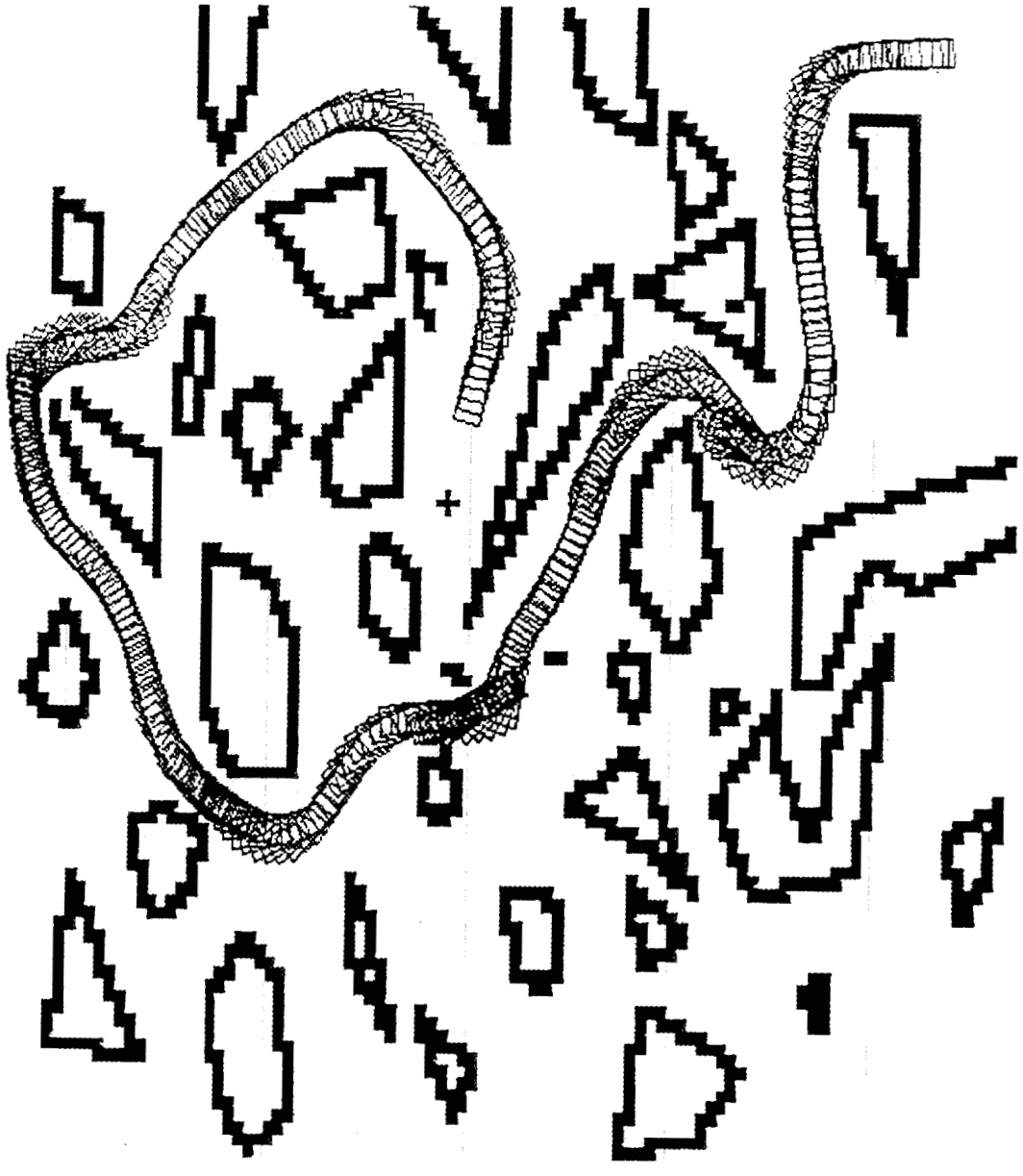
- The forward motion will be activated if the robot's heading is close to the attraction direction or if the robot has to stop again.

leti

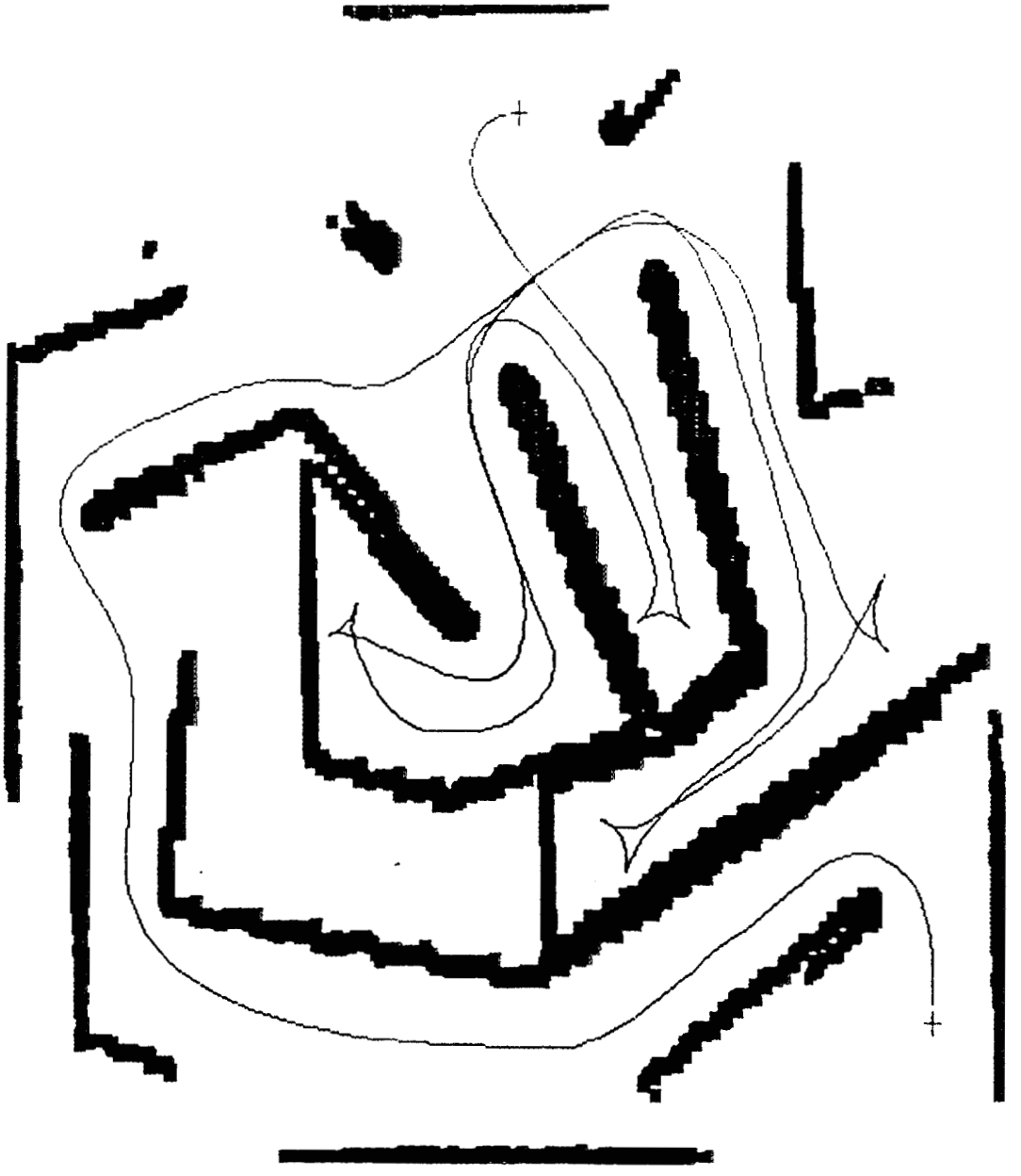


© cea / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

leti



leti



© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

leti

C. POSTURE PLANNING

Posture = Position + Orientation (x, y, θ)

The purpose of that planner is to perform an optimal collision free path to a goal in a grid model, with important geometrical constraints.

Modeling

(x, y) are discretized => grid use.

θ is discretized => one grid for each θ_i

Principle

- A propagation is performed from the goal posture through the free space.

- Each grid contains an image of the environment.

- Within a given plane, the propagation occurs only in the direction θ associated with the plane.

- Each direction change corresponds to a change of the active grid.

© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

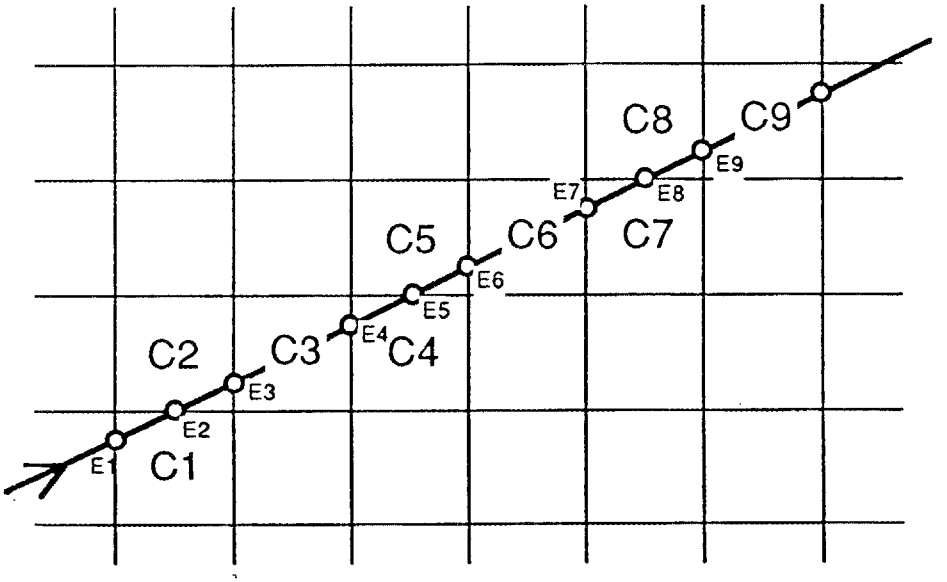
leti

Propagation in a plane

The algorithm is recurrent : each cell has in-points (or out-points) all around its sides.

From an in-point of a cell is computed the out-point.

This out-point is the in-point of the next cell.



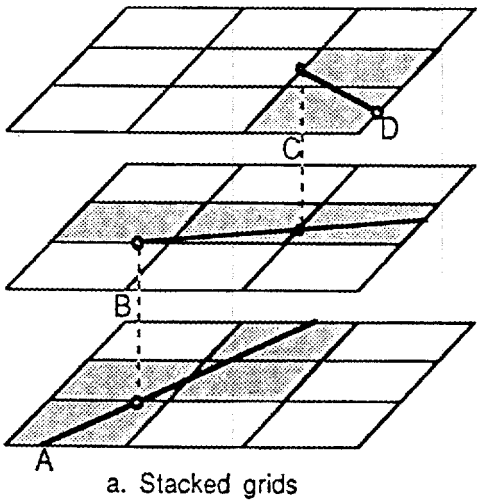
Each cell is activated only once (at the lowest cost).

© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

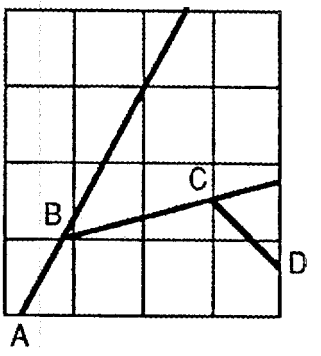
leti

Plane change

- A direction change corresponds to a plane change.
- The propagation change is performed at a in-point.

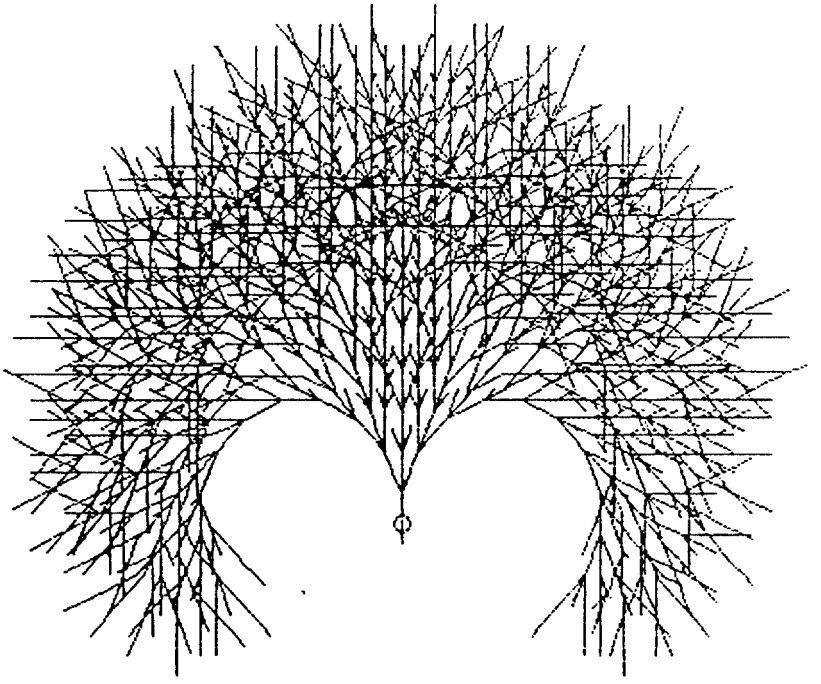


a. Stacked grids



b. Final result after projection in a plan

leti



© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

leti

Applications

Turning radius constraints

The propagation cost in a given plane has to cross a threshold before performing a direction (or plane) change.

Backward motion

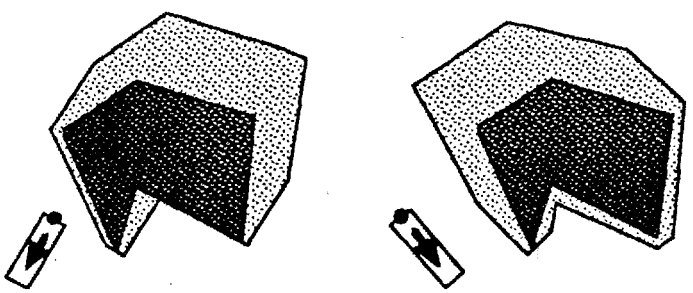
A new propagation rule is added.

A forward activation can perform backward activation and inversly.

New costs are introduced in such a way that it is more costly to move backward and it is costly to change of direction.

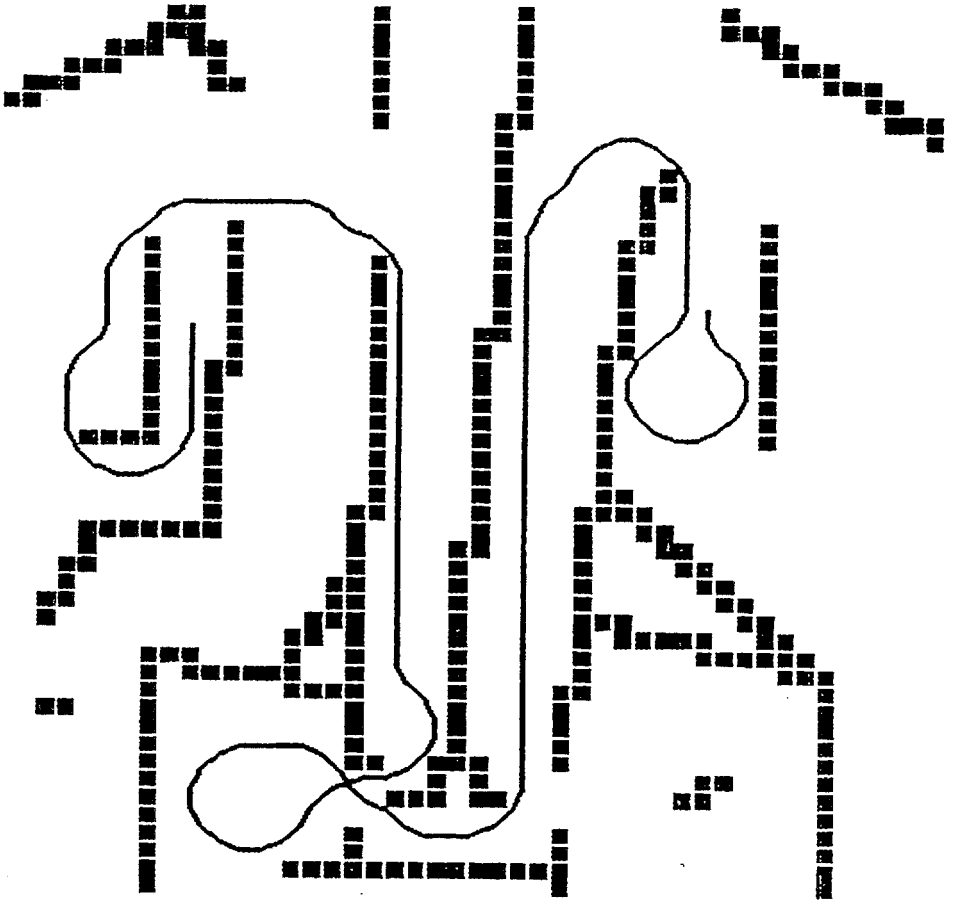
Robot shape

A configuration space (i.e. Lozano Perez) is determined . For each plane relative to θ, obstacles are grown according to the robot's heading θ. This allows us to solve the famous "piano mover problem".



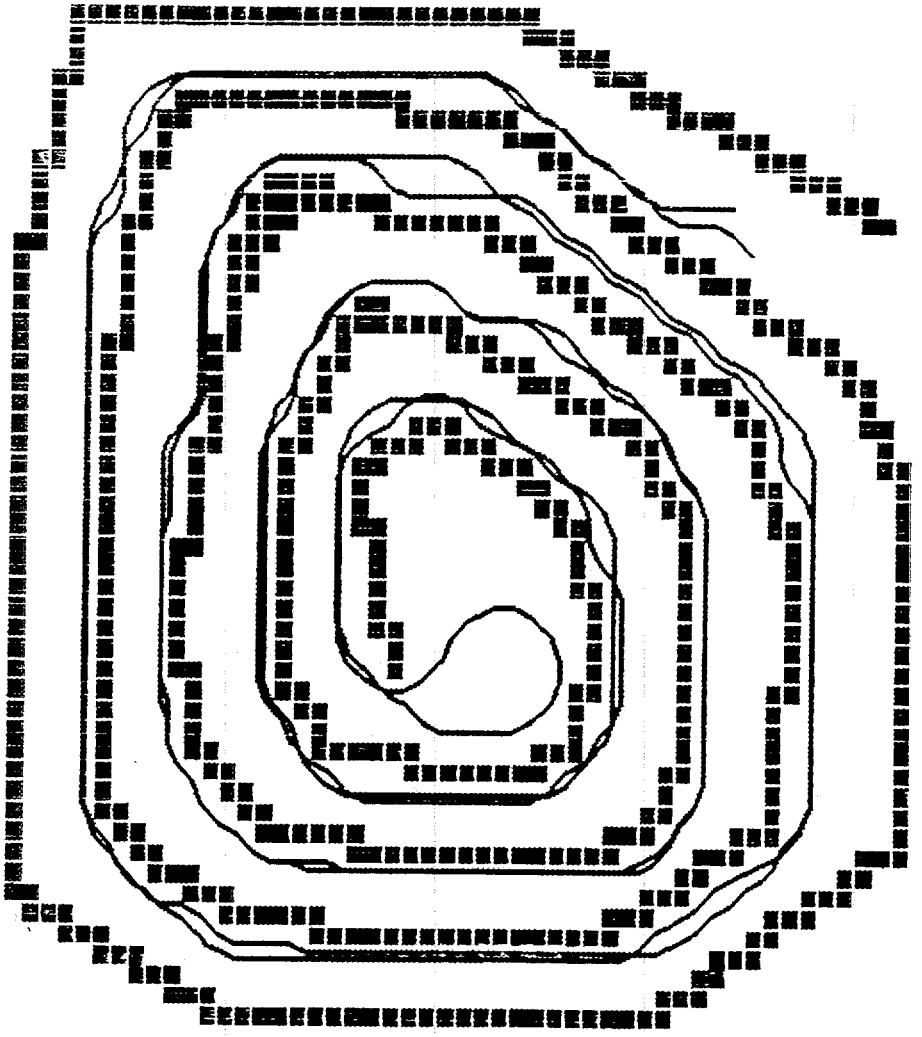
© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

leti



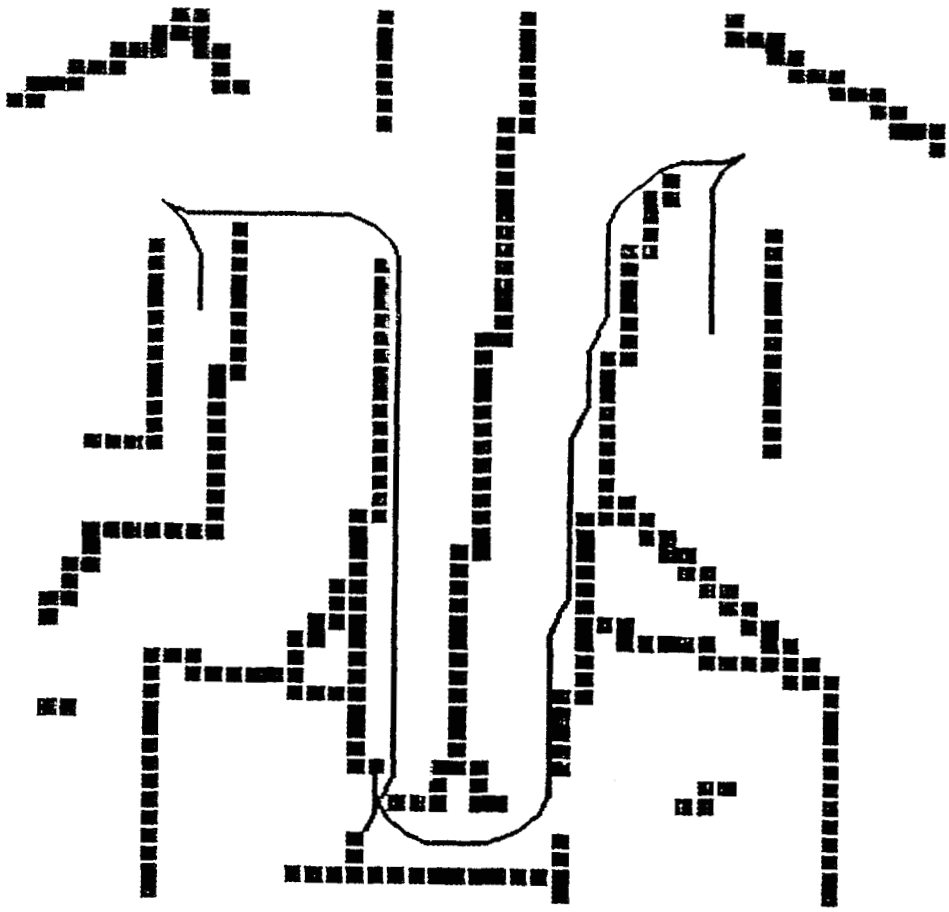
© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

leti



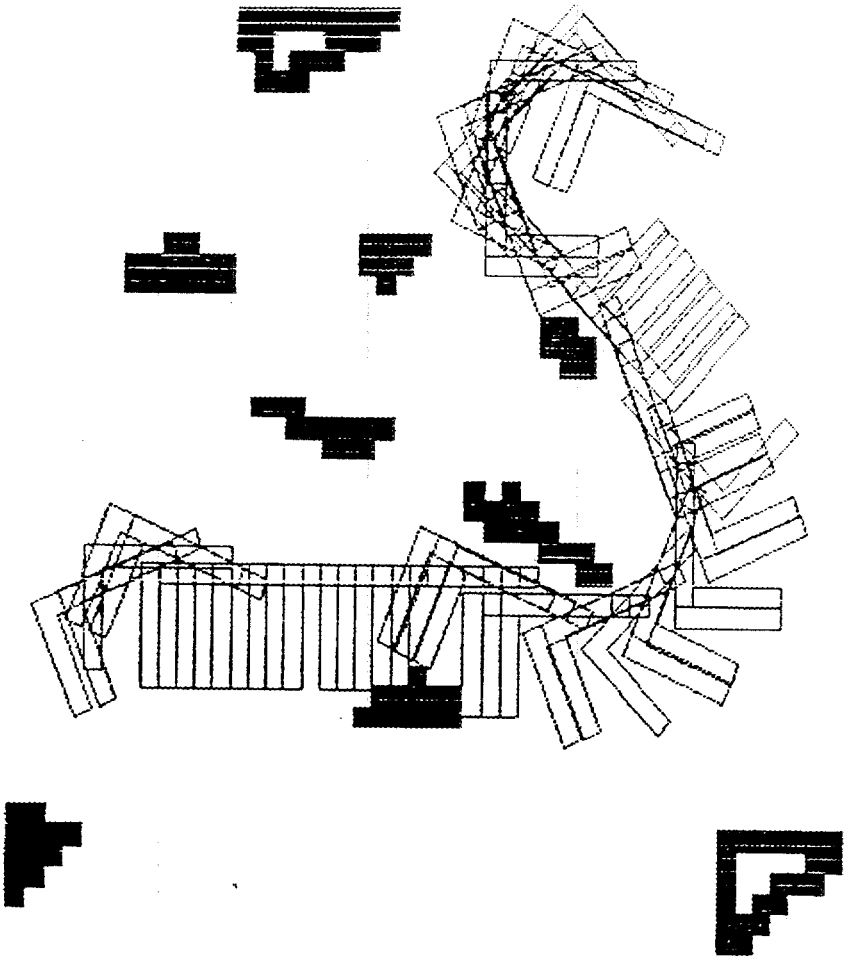
© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

leti

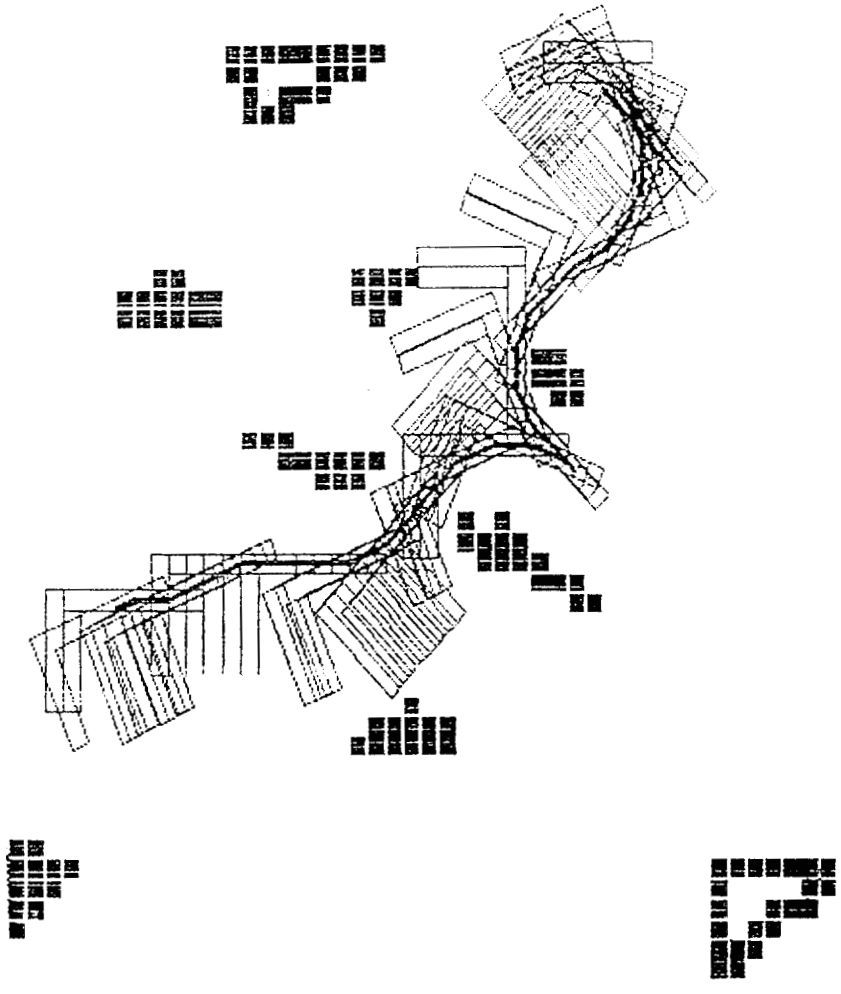


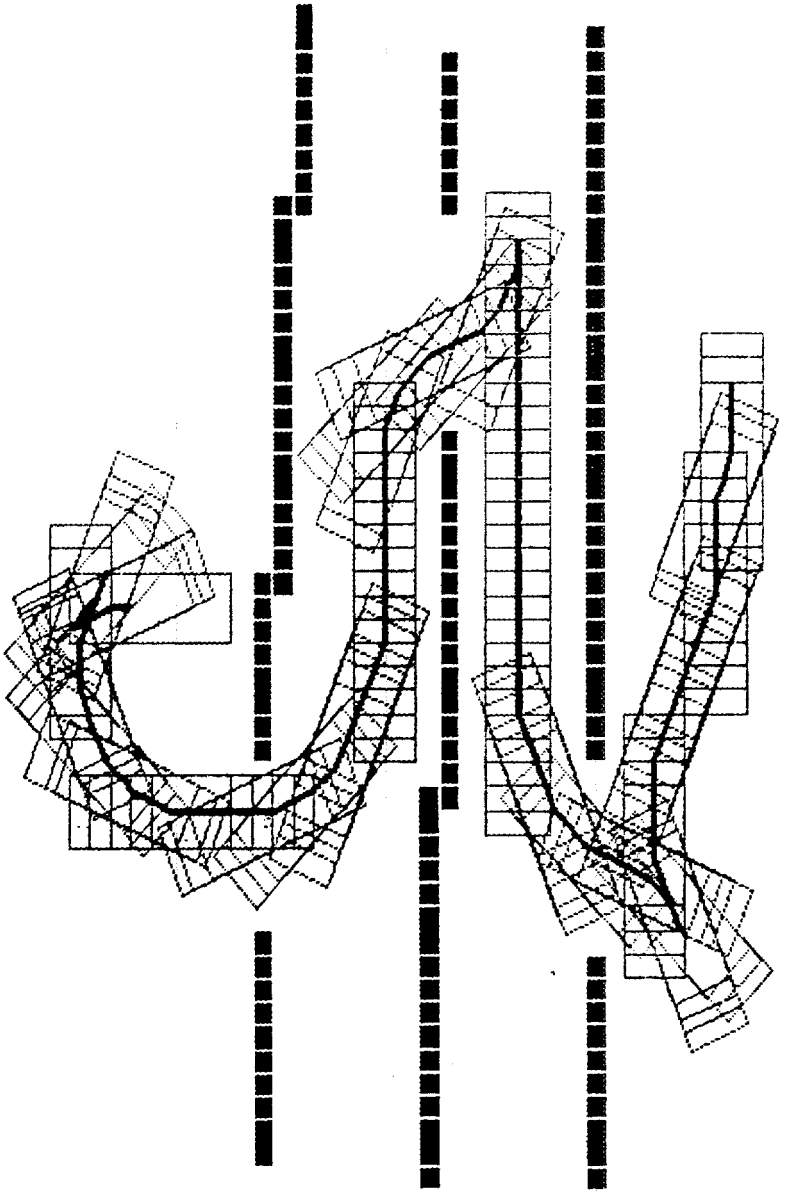
© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION

leti



© CEA / IRDI / DIVISION D'ELECTRONIQUE, DE TECHNOLOGIE ET D'INSTRUMENTATION





CONCLUSION

These three approaches have been simulated.

=> They are relevant for a large number of real time applications.

They are being implemented on our robot VERAU.

Session V: Recent Developments and New Test Beds

CESAR/CEA Workshop
1 june 1989

Autonomous Learning at a Control Panel

by

PHIL PELT

Machine Reasoning and Automated Methods Group

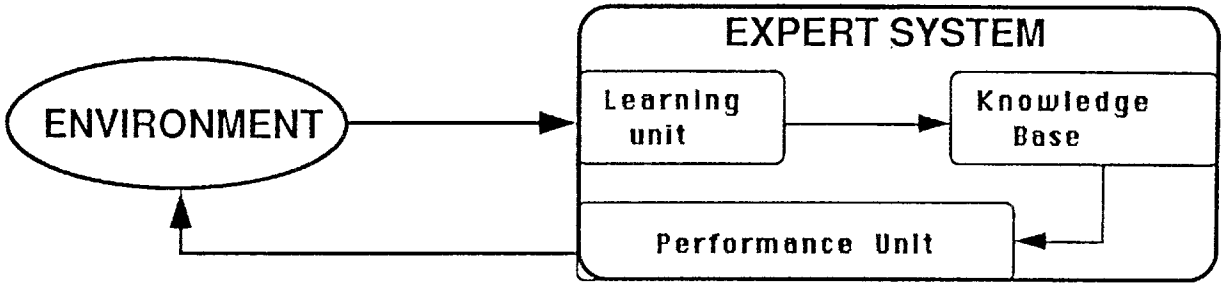
CESAR

Application to AUTONOMOUS Robot

Learning in an **Autonomous Robot:**

Robot's capabilities:

Act on environment and Observe consequences using camera vision and manipulator arms



Requirements of TASK

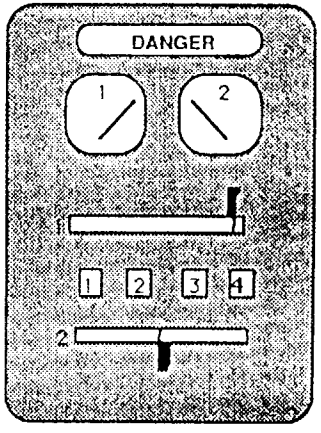
Problem: How to get robot to shut down or adjust process with minimum errors:

1. Learn SEQUENCE of responses for particular Initial Configuration.
2. Be able to CLASSIFY by initial configuration, i.e., find the correct categories.
3. PROPOSE response sequences which were (or might be) correct for initial configurations encountered.

APPLICATION: Details of a Control PANEL TASK

CESAR Panel -- application to specific situation:

Simulated Process Control Panel
- 6 Control Devices
  - 2 levers & 4 push-buttons
- 3 Visual Feedback Devices*
  - Danger Light & 2 meters

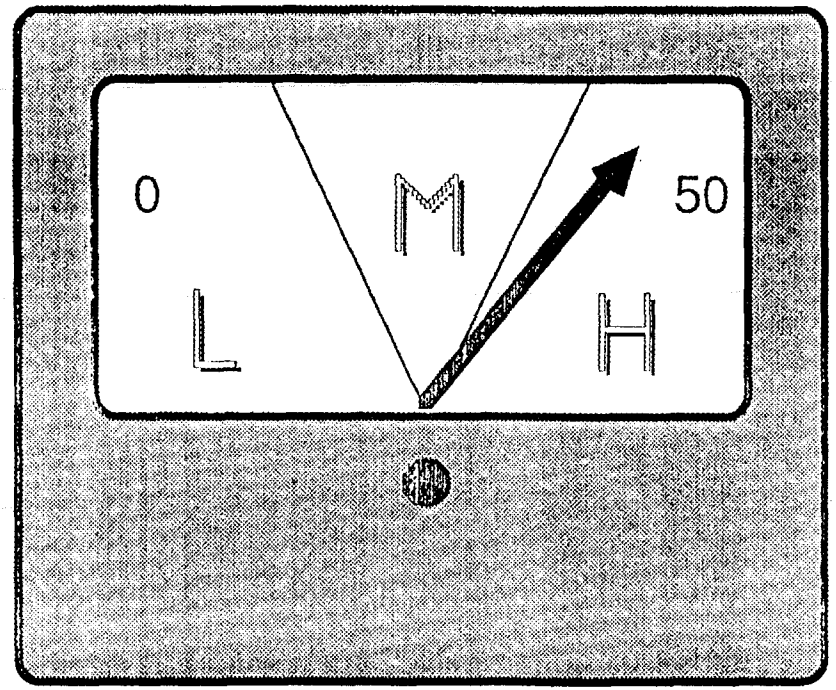


Panel is controlled by a PC, in which a program specifies the relationship between the control & readout devices

The control relationship defines problem categories, which the robot must learn.

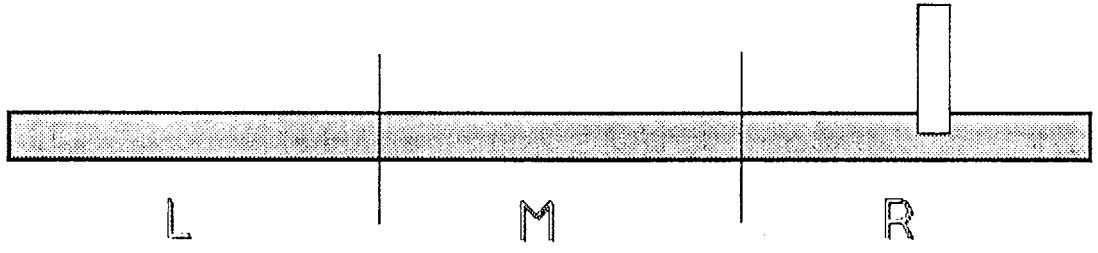
*Lever positions and Button Lights also provide visual feedback

Learning Project



3-value METER logic

Learning Project



3-value LEVER logic

2 Meters + 2 Levers = 4 devices, 3 states each

3^4 yields 81 panel states

TASK

Create a system to

1. *Search* for and remember correct **problem solving** sequences
   - Search by *Experimentation with Environment*
   - Immediate feedback for
     - each response & entire sequence
2. *Discover* problem **categories**
   - Generate *hypotheses* about correct category descriptors for **new** problems
   - Confirm/Deny hypotheses based on further experience with other problems in category
3. Infer solutions to new problems
   - Match attributes of new problems with those in already learned categories
   - Solve new problems with greater efficiency

Approaches to Task

Possible Solutions:

1. Program in all information (or train on all initial states).
2. Train on SOME problems and write inferencing routine(s) which **CLASSIFY** and **GENERALIZE**.

Approaches:

1. Expert System, written in **CLIPS**
2. Neural Network(s) written in 'C'

Three phases to system:

Stimulus-Response learning based on **IMMEDIATE FEEDBACK** for each response tried

Inferencing based on **VERSION SPACE** model by Mitchell, 1979, 1982

Hypothesis-generation

Input Symbols = Attributes & Values

Environment consists of a set of *Attributes* which can assume different ***values*** over time:

$$
\begin{array}{l}
A_1 \longleftarrow v_1 \quad v_2 \quad \cdots \quad v_i \\
A_2 \longleftarrow v_1 \quad v_2 \quad \cdots \quad v_j \\
\vdots \\
A_n \longleftarrow v_1 \quad v_2 \quad \cdots \quad v_k
\end{array}
$$

Problem Definition

A PROBLEM is identified by the initial or presenting conditions:

$$\left\{ \begin{array}{l} A_l \; v_i \\ A_m \; v_j \\ \cdot \\ \cdot \\ \cdot \\ A_n \; v_k \end{array} \right\} \longrightarrow S_x$$

Problem Solution

Initial Conditions

$$S_x = \left\{ R_1 \quad R_2 \cdots R_g \right\} \qquad g = (1, \infty)$$

Machine Reasoning and Automated Methods Group

CESAR

jsu589/20

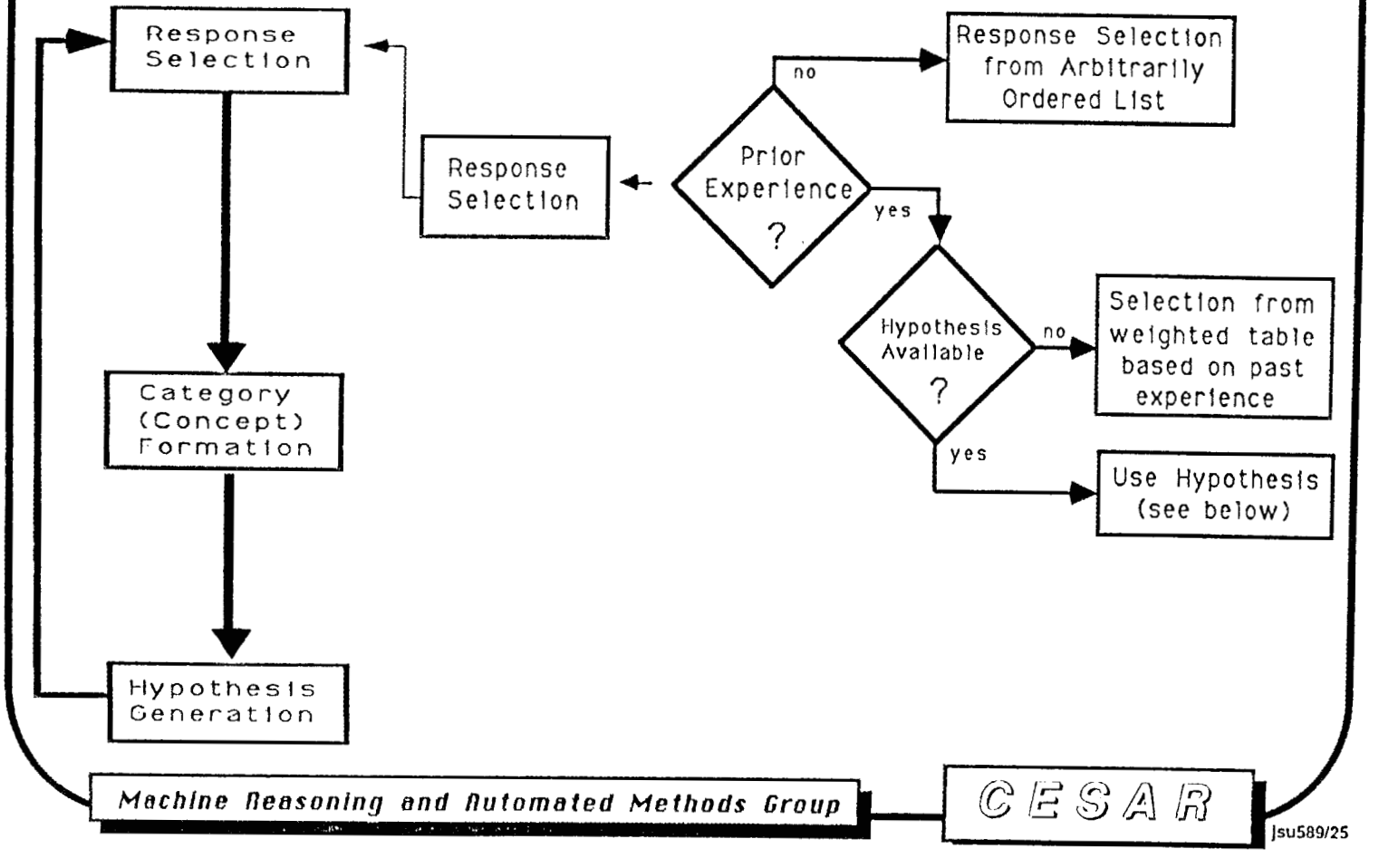
Categories of Problems

A problem *category* corresponds to the <u>response</u> <u>sequence</u> which solves all problems in that category:

GOAL: to find relationships between attribute values for all problems which belong to a particular category.

PRESENTING CONDITIONS				SOLUTIONS
$A_1 v_1$	$A_2 v_2$	. . .	$A_m v_i$	S_n
$A_1 v_1$	$A_2 v_2$	. . .	$A_m v_j$	
$A_1 v_1$	$A_2 v_2$	. . .	$A_m v_k$	
$A_1 v_3$	$A_2 v_1$	. . .	$A_m v_x$	S_t
$A_1 v_3$	$A_2 v_1$	. . .	$A_m v_y$	
$A_1 v_3$	$A_2 v_1$	. . .	$A_m v_z$	

Logic of CLIPS Rule Base: Response Selection Procedure

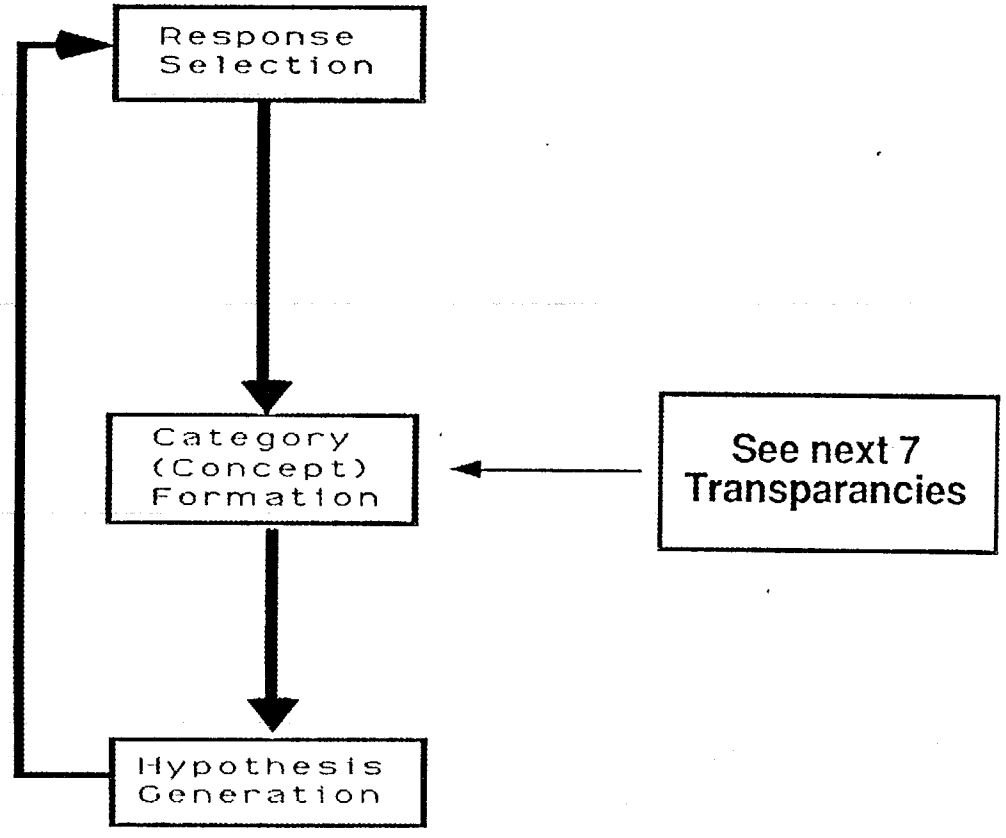


Machine Reasoning and Automated Methods Group

CESAR

Jsu589/25

Logic of CLIPS Rule Base: Concept Formation

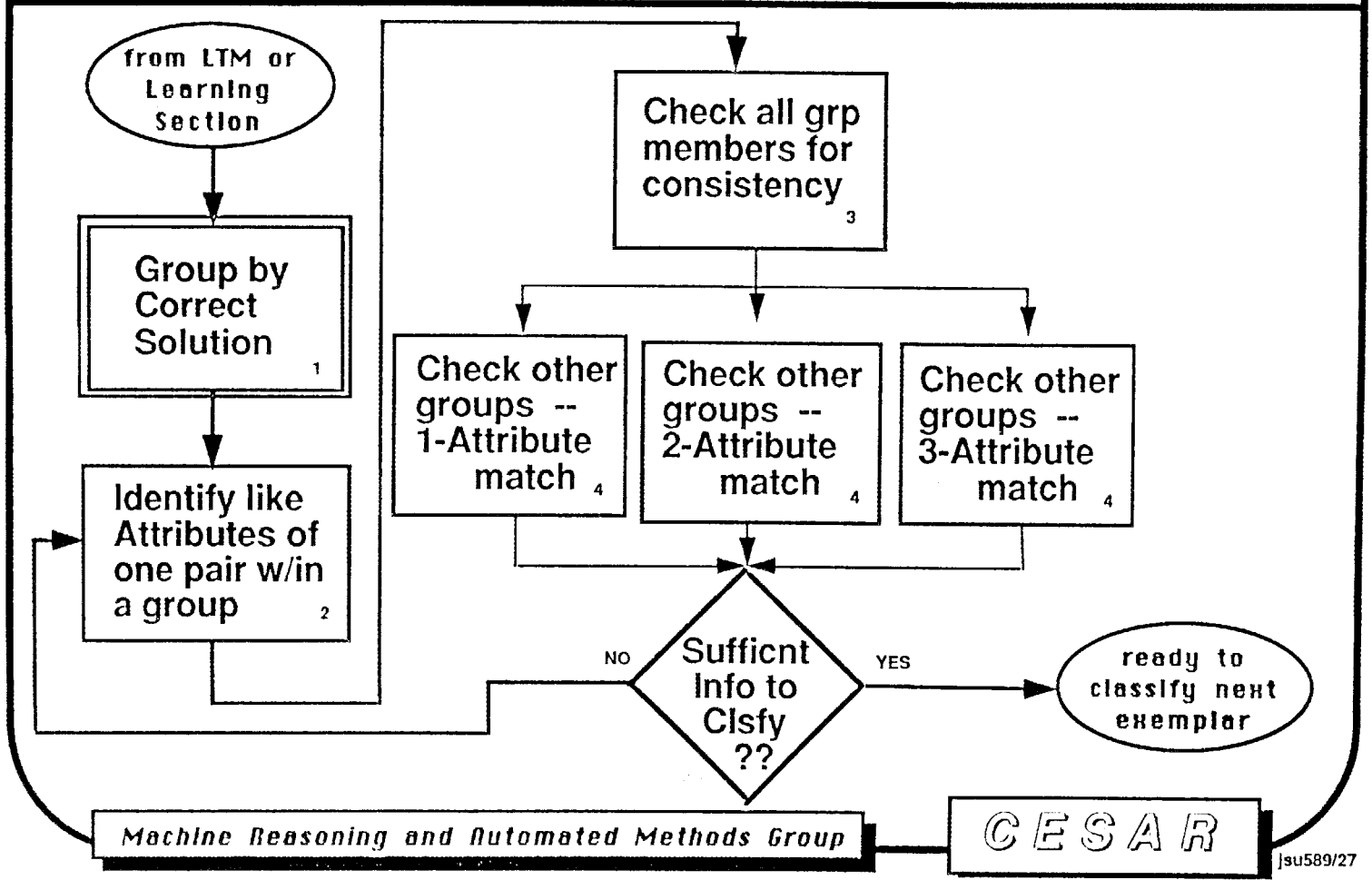


Machine Reasoning and Automated Methods Group

CESAR

jsu589/26

Logic of Categorization Algorithm



Machine Reasoning and Automated Methods Group

CESAR

jsu589/27

1 - Group by Problem Solution

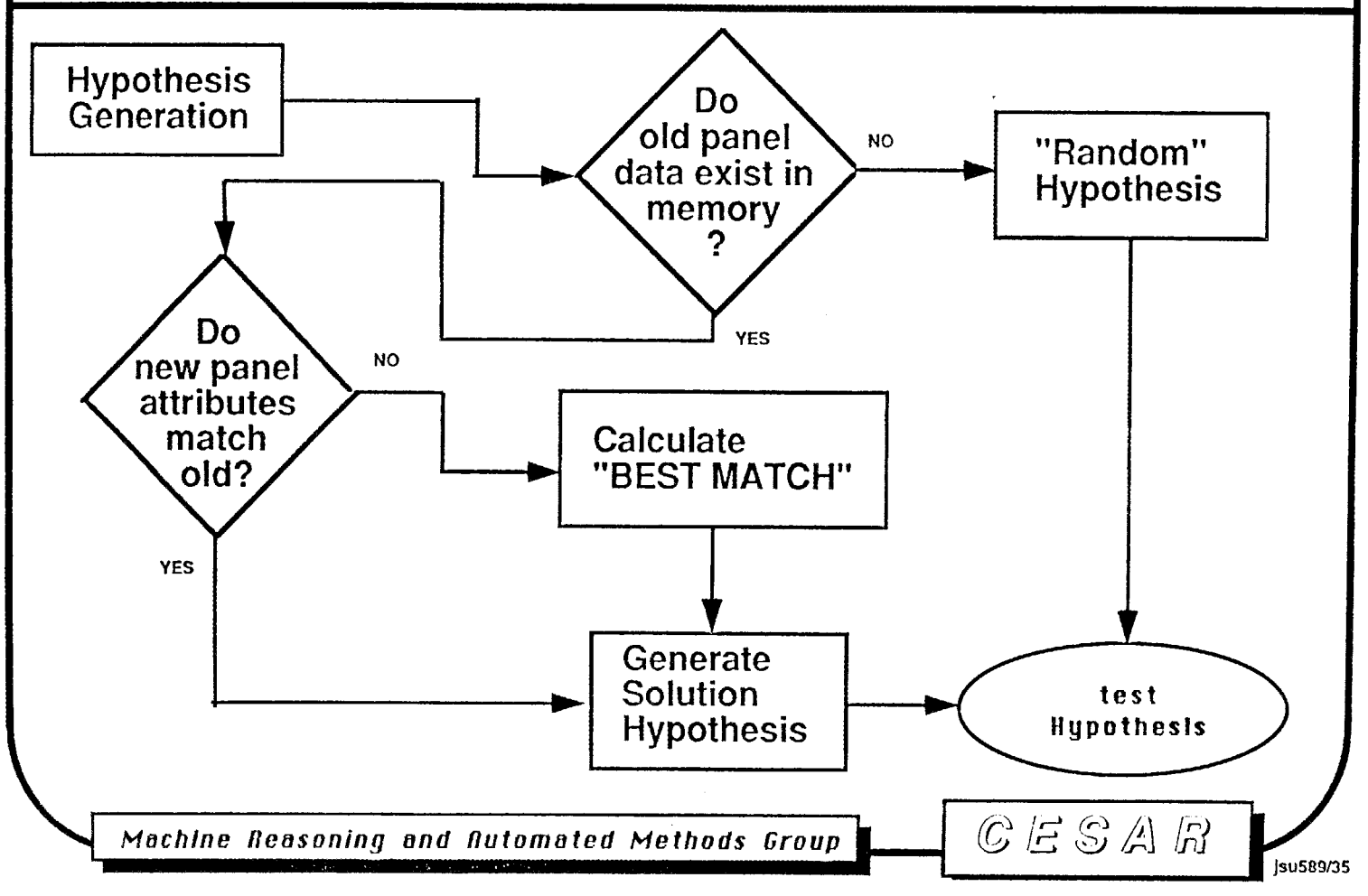
A problem *category* corresponds to the <u>response</u> <u>sequence</u> which solves all problems in that category.

All problems with a common solution are put into the same group:

	PRESENTING CONDITIONS				SOLUTIONS
1	$A_1 v_1$	$A_2 v_2$	. . .	$A_m v_i$	S_n
	$A_1 v_1$	$A_2 v_2$	. . .	$A_m v_j$	
	$A_1 v_1$	$A_2 v_2$	. . .	$A_m v_k$	
2	$A_1 v_3$	$A_2 v_1$	. . .	$A_m v_x$	S_t
	$A_1 v_3$	$A_2 v_1$	. . .	$A_m v_y$	
	$A_1 v_3$	$A_2 v_1$	. . .	$A_m v_z$	

Generation of Hypothesis about Correct Solution

111488/22

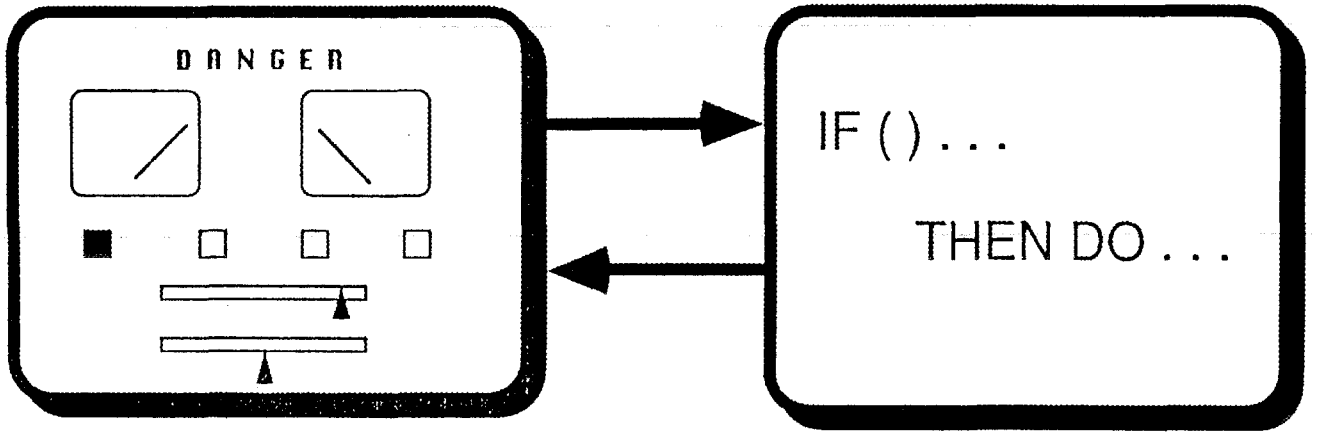


Machine Reasoning and Automated Methods Group

CESAR

jsu589/35

Panel-Robot Simulation Trainer



PANEL Emulator
Programmed in 'C'

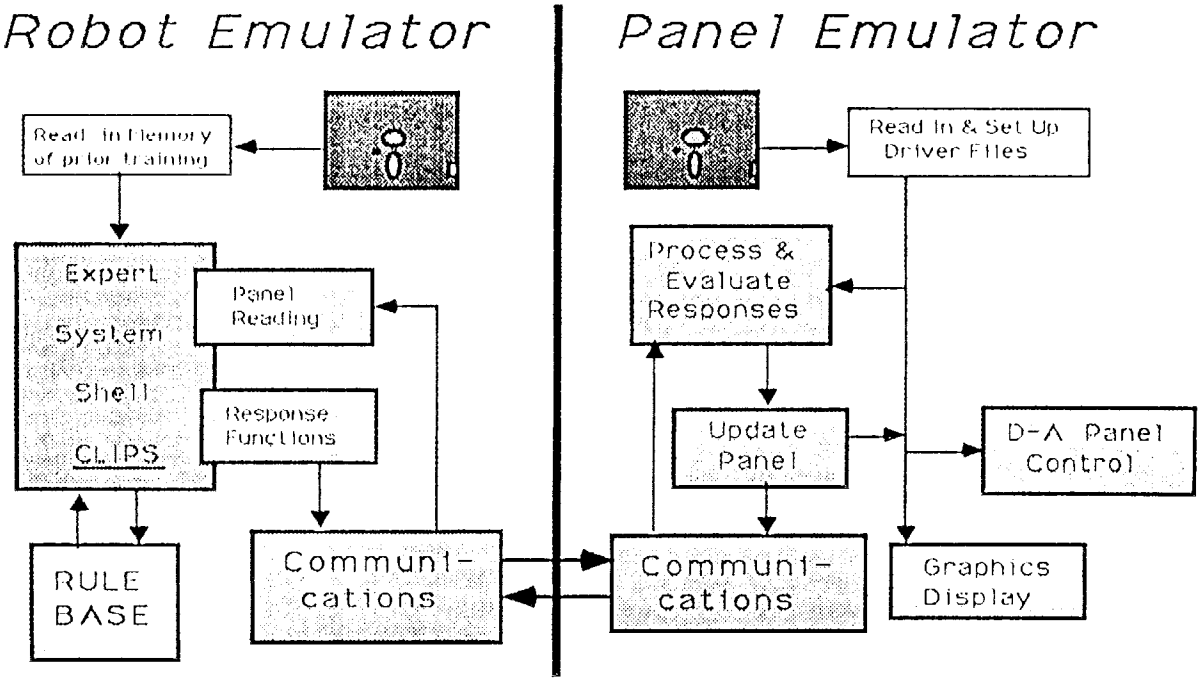
ROBOT Emulator
Rule base written in
CLIPS Expert System

PC-type Machines
Using Serial Communications Ports

Machine Reasoning and Automated Methods Group — CESAR

jsu589/44

Block Diagram - 2-PC Simulation System



Block Diagram of 2-PC Simulation System
for training a Learning Expert System

Machine Reasoning and Automated Methods Group

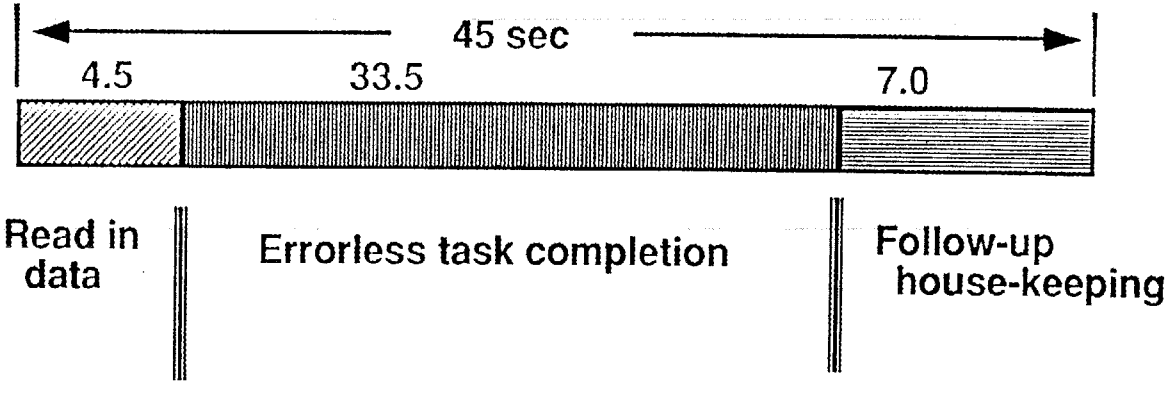
CESAR

Jsu589/45

Performance of Simulator

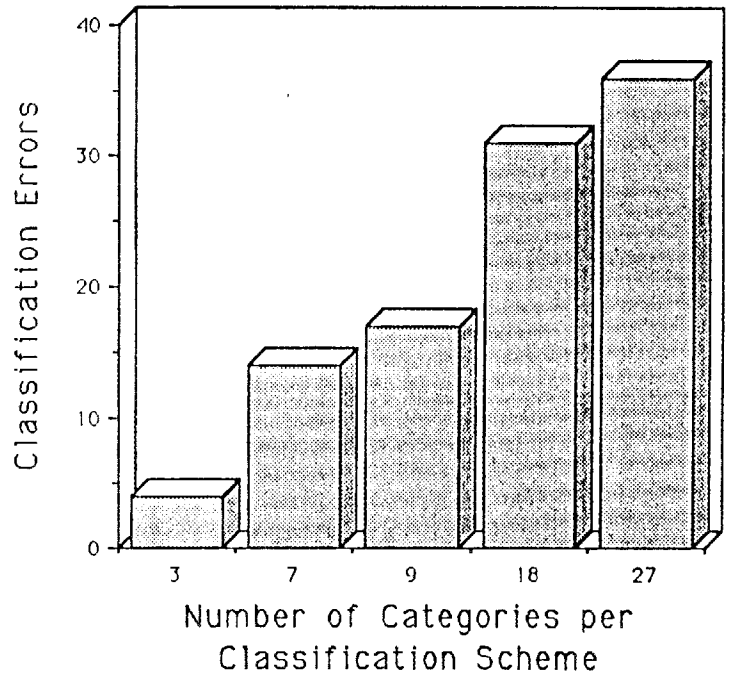
Timing Data (per problem, w/out errors):

$\bar{X}$ = 44.2 sec.
s = 2.2 sec.



Same task for robot at panel: 13-16 minutes

Classification Errors as a Function of Number of Categories

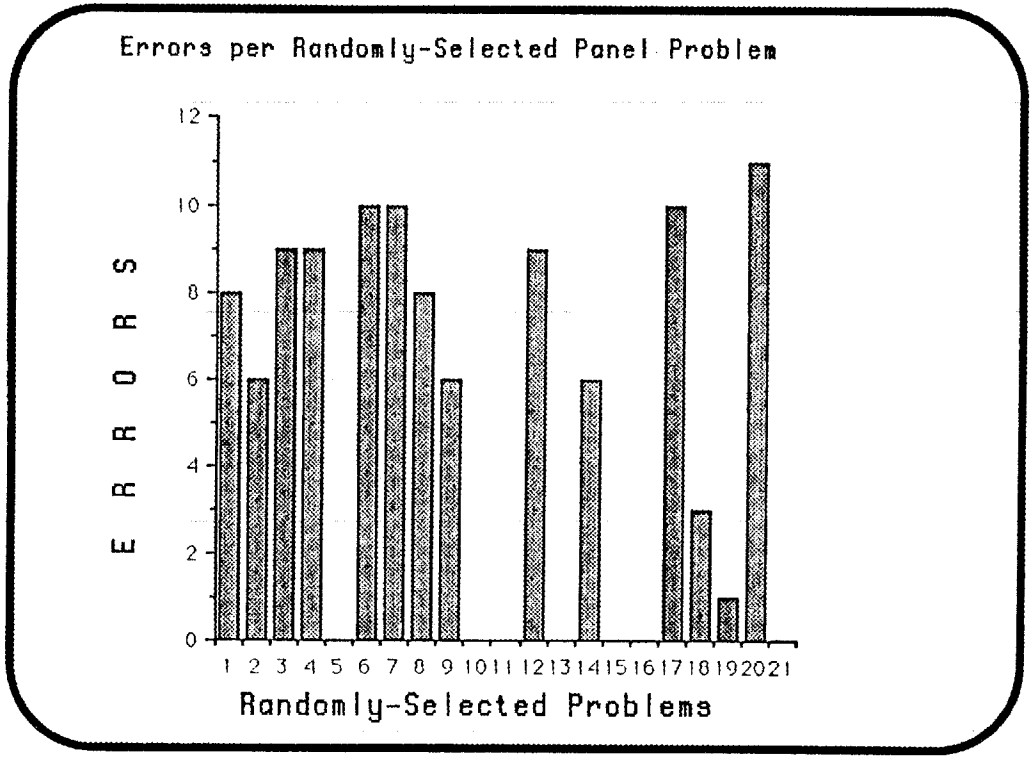


Machine Reasoning and Automated Methods Group

CESAR

Jsu589/48

Learning Project
24 May 1988

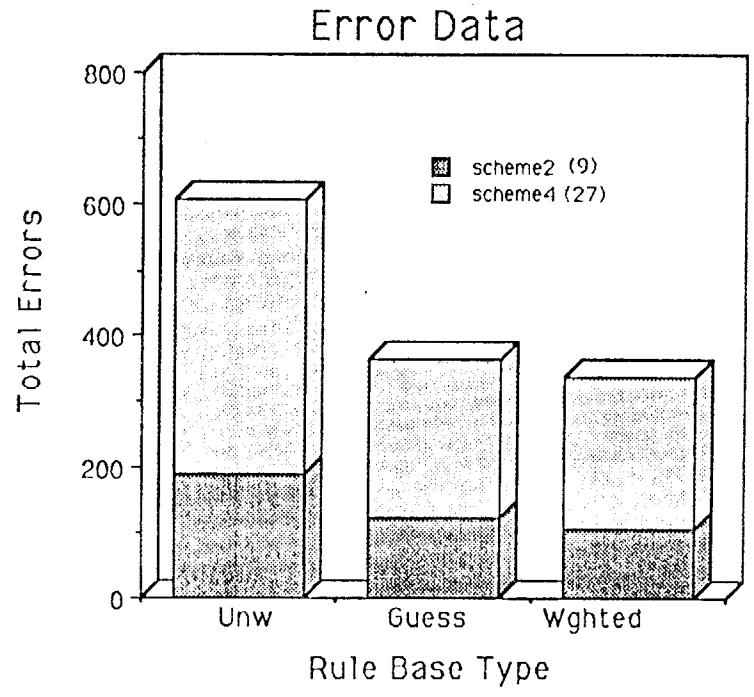


Data from MEMCAT.CLP

Machine Reasoning and Automated Methods Group

CESAR

Errors for 3 Rule Bases on 2 Panel Calssification Schemes



Machine Reasoning and Automated Methods Group

CESAR

jsu589/50

Conclusions

System Learns & Categorizes in (REAL) TIME successively more efficiently across development of 3 systems, and with a variety of classification schemes.

Immediate inferencing (using "Best Guess") leads to better performance than later, off-line LTM processing.

Selected training not necessary - random panels yielded performance as good as when pre-selected exemplars were used.

("Perfected") Expert Systems are more efficient than "bugged" Neural Networks.

Robot of the Future Here Now

". . . The difference between systems of today and systems of tomorrow will not be their strength or their human qualities -- it will be their intelligence. The real goal is not just robotic arms that can do something, but robotic systems that can recognize a situation, make a decision, and *then* do something."

— Andras Pellionisz
biophysicist
New York University
Medical Center

Quoted in "The Robot Reality", Mark Kemp, Discover, 9(11), November, 1988, p. 70.

Products from Project to date

REFERENCES

Spelt, P. F., deSaussure, G., Lyness, E., Pin, F. G., and Weisbin, C. R. Learning by an Autonomous Robot at a Process Control Panel. *IEEE Expert,* November, 1989.

_____, Lyness, E., and deSaussure, G. A Two Computer Simulation System to Develop and Train an Expert System. accepted by *Simulation* , fall 1989.

_____, deSaussure, *et al*, article on LTM processing aspects of project, in preparation.

_____, Silliman, *et al*, article on Rule Base & Neural Net performance, in preparation.

CESAR-87/55

LEARNING BY AN AUTONOMOUS ROBOT AT A PROCESS CONTROL PANEL

by

P. F. Spelt[+], G. deSaussure, E. Lyness[**],
F. G. Pin, and C.R. Weisbin

Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6364
(615) 574-7472

"The submitted manuscript has been authored by a contractor of the U.S. Government under contract No. DE-AC05-84OR21400. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of the contribution, or allow others to do so, for U.S. Government purposes."

Accepted for publication in IEEE Expert November, 1989.

---

*Research sponsored by the Engineering Research Program of the Office of Basic Energy Sciences, U.S. Department of Energy, under contract No. DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

+On leave from Psychology Department, Wabash College, Crawfordsville, Indiana 47933.

**Student at Iowa State University, Ames, Iowa.

LEARNING BY AN AUTONOMOUS ROBOT AT A PROCESS CONTROL PANEL

by

P. F. Spelt, G. deSaussure, E. Lyness, F. G. Pin and C. R. Weisbin

Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6364

The Center for Engineering Systems Advanced Research (CESAR) was founded at Oak Ridge National Laboratory (ORNL) by the Department of Energy's Office of Energy Research/Division of Engineering and Geoscience (DOE-OER/DEG) to conduct basic research in the area of intelligent machines. Within this framework, CESAR has undertaken several research activities in the field of machine learning. In this paper, we describe our approach to a class of machine learning which involves autonomous concept formation using feedback from trial-and-error learning. Our formulation is being experimentally validated using an autonomous robot, learning tasks of control panel monitoring and manipulation in effective process control. The CLIPS Expert System which resides in a hypercube computer aboard the robot, and the knowledge base used by the robot in the learning process are described in detail. Benchmark testing of the learning process on a robot/control panel emulator system consisting of two interacting computers is presented, along with results of sample problems illustrating machine learning and robot performance improvement. Conclusions are drawn concerning the applicability of the system to a more general class of learning problems, and implications for future work on machine learning for autonomous robots are discussed.

I. Introduction

The Center for Engineering Systems Advanced Research (CESAR), founded at Oak Ridge National Laboratory (ORNL) by the Department of Energy's Office of Energy Research/Division of Engineering and Geoscience (DOE-OER/DEG) to conduct basic research in the area of

intelligent machines, has recently undertaken several research activities in the field of machine learning. The present paper describes our initial work in autonomous learning using HERMIES-IIB, our current robotic experimental testbed. The integrated system in HERMIES-IIB (Hostile Environment Robotic Machine Intelligence Experiment Series IIB) is the latest in CESAR's series of autonomous intelligent machines designed to ultimately perform in environments which humans cannot readily enter. A detailed description of this machine and its navigation capabilities has recently appeared in IEEE Expert.[1] The computing power resides in two components -- a VME subsystem for vision input and for the I/O devices, and an IBM 7532 (an industrialized PC-AT) for the "brain". Four AT expansion slots house boards which provide an onboard 16-node NCUBE hypercube parallel computer. The hypercube machine is used for both vision processing and for running the Expert System described below.

With an emphasis on computational autonomy, research to date has focused on navigation in a dynamic environment, including the capacity of the robot to deal with unexpected moving obstacles using any of several strategies (e.g., replanning the goal path, moving small obstacles out of the way, or waiting until moving obstacles have cleared the robot's path)1. The robot's goal was to position itself in front of a "control panel", enabling it to read meters and manipulate buttons and levers. This paper reports the development of a system which learns the control panel's system dynamics and remembers the most efficient series of responses to "shut down" a control process, for

future encounters with similar (but not necessarily identical) situations. Ultimately, this system will also be able to infer a classification scheme for panel categories, enabling it to hypothesize about correct response sequences for panels not yet encountered, and some preliminary work on the inferencing section of the system is also reported here.

II. Background and Related Research

Although considerable research has recently been published on machine learning, most of it has focused on the so-called "higher cognitive functions" (problem solving, concept formation, rule learning, etc.; see 2, 3, and the journal Machine Learning), with relatively little published in the robotics literature. The great interest in higher capabilities understandably stems from the perception that they are uniquely human attributes which are directly associated with "Intelligent Behavior".

However, such exclusively cognitive tasks require no motor behavior capabilities such as our robot displays -- one needs only a stationary "electronic brain" to do the information processing. McMillan[4] has discussed other learning paradigms which might serve as helpful models for learning in intelligent machines. Some of these are especially useful for work with an autonomous mobile robot, the defining features of which are its ability to move around and manipulate the environment. McMillan's work simulated a low-level learning paradigm (Classical or Pavlovian Conditioning) used to modify an operating system's presentation of a menu of commands. The system

commands in the menu are ordered in a sequence intended to anticipate a particular person's next command, based on his or her past use of command sequences. The re-ordering of a hierarchy of commands based on success and failure (in predicting their use) is referred to as reinforcement in psychology and as credit assignment in machine learning[5]. Laird, Rosenbloom and Newell[6] have reported a system called SOAR, which exhibits automated learning in a wide variety of tasks, including motor performance, and which takes biological models as a source for some of its concepts.

A frequently observed type of learning in biological systems involves the manipulation of objects in the environment (Instrumental or Operant Conditioning, based on motor responses), with such behavior followed by some type of feedback (reinforcement or punishment) concerning the suitability of the responses in that setting (e.g., the use of a wide variety of ON/OFF switches on appliances and machinery, and the cracking open of many shelled sources of food such as nuts and oysters, in the animal kingdom). Because much of so-called higher human learning is based on these simpler forms of conditioning at various points in the human's learning history (see, e.g., 7, 8) it seems useful to explore such learning for autonomous robots, as a basis for future developments in more cognitive kinds of robot activities.

IEEE Expert article -- Spelt, et al, 1988

III. Overview of Machine Learning

The general strategies and orientations in the field of machine learning have recently been summarized by Michalski (2, vol 2, chapter 1). Five strategies for transforming information provided in the learning situation (making inferences) have been identified: rote learning, learning by instruction, learning by deduction, learning by analogy, and learning by induction, listed in increasing order of complexity of inferencing on the part of the learner (2, vol 2, p.14). Learning by induction, in turn, has been further divided into learning by observation and discovery, and learning from examples. In the latter case, the examples can be provided either by a teacher, who knows the concepts, or by the environment, on which the learner performs experiments from which it receives feedback on the correctness of the performance. Michalski also divides the process of induction learning into part-to-whole generalization and instance-to-class generalization. In the latter case, the learner receives independent examples of classes of objects, and is to infer from those examples a general classification scheme which describes those classes. Our system embodies this last type, learning-by-induction using instance-to-class generalization based on examples which the robot has provided for itself by experimenting with the environment.

The work reported here involves development of an Expert System consisting of two major components: one Response-Sequence Learning unit to experiment with the environment in order to learn sequences of responses associated with particular environmental states to solve a

problem, and an Inferencing component to generalize from those self-generated examples to be able to infer categories of problems, thereby enabling the robot to generate hypotheses about possible correct sequences for as yet unseen problem examples.

The Response-Sequence Learning unit consists of a subset of the rule base which discovers, through a trial-and-error process, the appropriate sequence of manipulator-arm actions to solve problems represented by the values of various attributes in the environment. This sequence learning process involves a breadth-first search through the available responses to discover which one is appropriate at a particular point in the problem-solving process. A correct response receives immediate feedback, and the system then adds that component to the sequence being built. Once the entire correct sequence has been determined (as indicated by final feedback from the environment), the system then associates that response sequence with the initial or presenting set of environmental attributes in which the sequence was learned. Thus, this unit learns the basic category examples, consisting of initial environmental states and associated response chains, which form the basis for inferring problem categories by the Inferencing unit of the system.

The second major component of the rule base -- the Category Inferencing unit -- is under development at the time of this writing. In order to test the hypothesis-generating capabilities of the response-learning unit, categories of problems were initially programmed into the Expert System, and most of the data presented below

were obtained using these pre-programmed categories. Michalski (2, vol 2, p 16) distinguishes two techniques for generalization: similarity-based and constraint-based techniques. The former technique, which is used in our system, explores examples and counter-examples of a category (inter-example relationships) to create concept descriptions. It searches for attribute values shared by examples in the same class and ignores those that are different, while at the same time identifying those attribute values which are different among different categories.

IV. The Learning Task

We chose to have the robot learn at the control panel in order to take advantage of both HERMIES' capacity for manipulating objects with its manipulator arms and the successful docking of the robot in front of that panel[1]. Our longer-range and more complicated goal is to have a robot autonomously diagnose and repair a variety of similar, but not identical, plant components (e.g., process control valves, meter level adjustments, etc.).[a] Figure 1 shows the robot manipulating a lever on the CESAR control panel to, e.g., determine whether such a motion contributes to a temperature reduction and the shutting off of a high temperature Danger Light. Because the HERMIES-IIB configuration was

[a] We do not intend to suggest that a robot should be sent into a dangerous situation to experiment in a trial-and-error manner with switches on a control panel! The use of this control panel is merely a convenience for testing a general learning system which can be applied to a variety of non-critical real-world situations.

designed to stress research concerned with on-board computation, the hypercube computer architecture on-board is quite sophisticated and powerful[1]. On the other hand, the manipulator arms on this version of the robot are primitive and relatively weak, capable of only very rudimentary environmental manipulations, such as pushing buttons or moving light-weight levers. Thus, the movements to be learned are correspondingly limited, but the methodology is general and adaptable to much more complicated experimental situations. A parallel research effort at CESAR deals with development of a seven-degree-of-freedom arm with considerable speed and sophistication of movement, to be mobilized aboard the HERMIES-III vehicle in 1989, along with a later version of this expert system.

Insert Figure 1 about here

The CESAR control panel is a metal box .61 m wide by 1 m high, containing two 5.7 cm by 10 cm analog meters, a row of four 1.3 cm square pushbuttons, two horizontal slide levers, and a "Danger" light at the top. The panel functions are controlled by a microcomputer which controls the state of the panel -- which buttons are lighted, the settings of the two meters, and the on/off status of the danger light. The panel control computer also has knowledge of the appropriate sequence of button/lever moves to shut off the Danger light, and provides immediate feedback corresponding to the suitability of each robot action: a button pressed at the correct point in the sequence lights up; a correct lever move has the effect of moving its

associated meter to a new position. The final correct response in the sequence turns off the Danger light, while incorrect responses have no effect on the state of the panel.

The Expert System shell chosen for the robot's learning was CLIPS 4.0, developed at NASA/Johnson Space Center[10]. This shell, written in 'C', permits writing IF . . . THEN . . . production system rules with complex antecedents and sophisticated consequences, including mathematical computations and the capacity to call user-defined functions. Knowledge in an Expert System exists in three components of that system: the facts asserted into Working Memory (STM), the rule base and matching which occurs on the Left Hand Side (LHS, the IF . . . part) of the rules, and the control scheme embedded in the inference engine which handles conflict resolution in case more than one rule is simultaneously activated. Our Expert System makes use of two memory functions to show learning both within a particular session in front of the panel, and from one session to the next. These two functions are equivalent to Short Term (Working) Memory (STM) and Long Term Memory (LTM) in the Information Processing model of human cognition[11]. In Expert System terminology, the Knowledge base is traditionally viewed as consisting of the system rules and the facts on which those rules operate. Working Memory (STM) refers to the facts of the expert system, exactly the sense in which we use the term. One major difference between a learning expert system and a more traditional one is that most of the important facts on which the rule base operates are learned, rather than being pre-programmed. In the present system, LTM

content can be recalled from a (DOS) disk file at the start of session if the human operator so chooses, so that a particular session can be run with either a naive or an experienced robot.

A set of vectors contains STM for each correct response the robot makes, and similar information for past responses if the LTM option has been exercised. Each vector contains a set of facts which describe the initial state of the panel meters and levers, the current state of all devices on the panel, the category into which the initial state of the panel was classified (if it was possible to do so), and the present step in the sequence of responses for the current trial. If LTM is not used, a completely naive robot confronts the initial task at the panel, and the only memory is that which is gained during the session (i.e., STM); if the LTM option is selected, then part or all of the robot's experience preserved from past sessions at the panel (prior STM vectors) is retrieved. The system also creates a Response Hypothesis Pool (the set of all six possible responses, 4 buttons and 2 levers) at the start of each task.

The two learning strategies judged to be those most likely to be used by a human expert in remembering how to manipulate a control panel are encoded in the rule base. We have designated these as the Initial State Strategy and the Current State Strategy. The Initial State Strategy bases the entire correct response sequence on the initial configuration of the meters and levers on the panel (prior to the robot's making any manipulations which might change things) and is designed to use two levels of matching of present conditions to past

experience. An exact match of the initial panel state occurs if the starting positions of levers and meters exactly match what was found at the start of a previous panel. A category match occurs if the initial characteristics of the panel permit the system to classify it as a type similar to what was seen before, even though it is not an exact match. When a match occurs, the system reduces the size of the Hypothesis Pool by selecting the response sequence which was successful for that exact panel or that category in the past. If more than one successful response sequence for a particular panel/step combination was successful, the robot will try each in succession until one succeeds. Should there be a match with a previous panel but no previously successful response works (if, e.g., the panel were a different brand or had been rewired), then the system moves to the next most broad category -- from exact match to category match, or from category match to random response selection (see results section below). Once a successful response is found for the current step, the robot reclassifies back to the original kind of match before proceeding to the next step.

The Current State strategy bases a particular correct response on the state of the panel at the point at which that particular response previously was indicated to be correct. Thus, this strategy requires matches at various levels of agreement between current conditions (for each step) of the present and past panels. These degrees of correspondence are pre-arranged hierarchically, with an exact match of all panel characteristics being the highest degree of correspondence;

if none of the panel characteristics match, a random selection from available responses is made. At present, the relative importance of the various panel characteristics (both meters and both levers, only the meters, only the levers, etc.) is predetermined, and is not updated with each experience. For both strategies, the number of possible responses with any kind of match would be considerably smaller than the total number of available responses, ideally only one. Because a large number of errors greatly extends solution time, a particular response is attempted only once at a given step in the process of solving the problem.

V. Performance of the Rule Base

Because the proper logical operation of the rule base is independent of the robot's actual interaction with the panel, we developed a two-computer system which simulates the interaction between the robot and the panel. This system consists of PC/XT-type machines communicating with each other through the serial communications ports. The robot simulator runs the expert system, sending commands concerning responses tried to the panel emulator. The panel emulator[b] sends information concerning the current state of the panel back to the Expert System. Additional features of the panel emulator are its capacity to present a series of problems to the rule base from a batch file, and to record the number of errors made on each problem.

To deal effectively with the many possible panel configurations

[b] The panel emulator was written in 'C' by Mike Pav, a visiting student from Knox College. It was later modified by one of us (E.L.) to handle I/O through the communications ports.

and response sequences, we have selected two general categories of tasks for the robot to initially learn. One task, the "Nulling Problem"[9], comes from the literature on process control. The design of this type of control system is based on a time multiplexing philosophy (the same device being used more than once, sequentially over time) in which a single control device (either of the levers) controls more than one process (the two meters). Because this design ". . . is common, takes time to carry out, time to learn, and is a common source of error" for human operators[9] (p. 331), we have included it as a rigorous test of learning in our Expert System (in our laboratory, the 2 lever moves are preceded and followed by a button press, yielding a sequence of four correct responses). The second task, designated the "In-House" problem, involves no repeated responses, but does require a sequence of four button/lever moves, making the two tasks directly comparable for statistical analysis. An example of an In-House task at the panel follows (refer to Figure 1): **Initial Settings** - all buttons OFF, left meter HIGH, right meter LOW, top lever RIGHT, lower lever MIDDLE. Responses made by robot (correct ones in bold, feedback in parentheses): Button 4, **Button 1** (Lighted) --> Button 2, **Lever 1 left** (left meter to LOW) --> **Button 2** (Lighted) --> Button 3, **Button 4** (Lighted, Danger light OFF). In this example, the robot made 3 errors, and will remember **High - Low - Right - Middle ---> Button 1 - Lever 1 left - Button 2 - Button 4** as the LTM information from that example, to be used for inferring panel categories.

Experimental procedure - The rule base was evaluated using the

simulation system just described by presenting a set of 18 problems to be solved independently by each learning strategy. Each strategy was trained without recalling LTM (completely naive robot). The first four problems were the standard ones (2 different Nulling, and 2 different In-House), with the same four repeated to test STM for the same task. To test for generalization, the third set of four problems changed in the initial configuration of the panel, so that the problems were in the same class[c] as the first four, but were not exact matches (a meter or lever was in a different position, but the same response sequence was required). The fourth set of four problems consisted of identical panel configurations to the standard four, but with a different first response in the sequence, to test whether the robot could behave intelligently after an altered first response in an otherwise known sequence. The final two problems were unique configurations and response sequences, to test whether any benefit comes from prior experience with panel configurations which have little similarity to those the robot presently confronts.

Results. - The data for both strategies are summarized in Figure 2 as a function of type of task. The left panel shows the average errors made by each strategy on the Nulling and In-House problems for the original (naive) tasks, averaged over several presentations of each type of

---

[c] "Class" of panel is determined by the rulebase: The meter and lever positions are coded as a 4-digit number, with the panel category or class being specified as a range around the number representing one of the standard configurations. Eventually the rule base will determine the categories.

problem. The right panel of Figure 2 presents data gathered when panel characteristics exactly matched previous experience, but the first response in the sequence was changed. As can be seen, the Initial Strategy made fewer errors than the Current Strategy[d]. This effect derives from the fact that a particular intermediate state of the panel might appear in more than one solution sequence (i.e., might be associated with more than one Initial Panel State), and thus might have more than one response associated with it. This would produce more errors for the Current State strategy than for the Initial State, which generates a hypothesis about the entire response sequence at the start of work on a problem.

In addition to the data shown in Figure 2, additional data showed that neither strategy made any errors on the exact match (repeated) tasks, indicating that both strategies learned, as reflected in STM for correct responses from trial one to trial two for each task. Also, performance on the Class Match tasks showed no errors for the Initial State strategy, and an average of less than two errors for the Current State strategy (due to the particular way the Current State section selects responses with a partial match). Performance for both strategies on the final two unique problems was no different than on the original set of tasks, as would be expected with no inferencing by the expert system. Finally, the robot's performance when the LTM

[d] A t-test for the difference between two means showed that the mean for the Initial Strategy (2.06 errors) is statistically lower than the mean for the Current Strategy (6.56 errors). The t value was 5.31, with df = 6, and $p < .01$.

option is used is the same as it is after learning during the current session, since the LTM option reads past experience into STM. Taken together, these data show that the system learns from past experience, and that significant benefits result from this learning, even when conditions are not identical to those previously encountered. This ability is a prerequisite for the robot to be able to make use of categories it will later infer from experience with panel examples.

Insert Figure 2 about here

Figure 3 presents data comparing the robot with the average performance of 7 cooperative education students assigned to the laboratory. Two aspects of these data are noteworthy: the robot performs consistently better than the humans for all classes of problem; and the Nulling Problems are much more difficult for humans, as the literature suggests, but not for the robot. Therefore, we appear to have a robust Response-Sequence learning component for the Expert System, and one which can take advantage of categories if they exist.

Insert Figure 3 about here

Figure 4 presents data from the initial efforts at having the Expert System infer categories from experiences generated by operating the control panel under different initial problem conditions. All preprogrammed information about panel categories was removed from the expert system, and a classification scheme was created for the panel emulator which placed all possible initial panel configurations into

one of seven categories, each with a unique solution sequence. A set of 14 pre-selected training problems, two per category, was presented to the robot by the panel emulator, following which the system applied an inferencing scheme based on the ideas presented in section III above. As can be seen in Figure 4, the robot showed no improvement in performance across the 14 problems. However, after LTM processing (inferring the categories), the robot made no further errors on any panel, whether or not it had been seen before. Further development will permit the system to make inferences about category configurations as each piece of new information about the world is obtained, work which will be presented in a future paper.

Insert Figure 4 about here

Because an important characteristic of an Expert System is its ability to explain what it does, HERMIES provides a protocol (record and explanation of actions) of the learning session for later analysis. Table 1 presents some sample output -- the top portion shows that HERMIES has classified the panel into a class already experienced, permitting use of a known sequence of responses. The lower part of the protocol illustrates another important component of an expert system -- the ability to degrade gracefully. If all available responses fail to produce positive feedback, this expert system "consults" with a human cohort via a radio link to a stationary remote terminal[1]. As can be seen, the system listed all failed responses and then asked the consultant whether the robot should quit or try another response. In

this example, the human asked the robot to try button 1 (response # 3) again, which lighted the second time it was tried, a situation which might arise because the end effector missed the button, or did not press it hard enough to cause it to light the first time.

Insert Table 1 about here

VI. Conclusions and Future Developments

The research reported here illustrates the feasibility of developing a learning expert system which can function in an autonomous mobile robot, learning from experience generated by that robot as it manipulates the environment. In the present implementation, the system learned a sequence of responses which would alter or shut down a control process. However, the general methodology is applicable to any situation in which a robot needs to learn a sequence of motor operations, such as an assembly operation. We have demonstrated the use of a simulation system to train the robot prior to its entering a hostile or critical environment. This training, coupled with the system's ability to solve novel tasks by generalizing, eliminates the need to pre-program all possible real world situations. We showed that by presenting a selected set of panel configurations during such a training session, the robot can develop the capacity to handle a wide variety of unanticipated panel configurations, making a minimum number of errors. Finally, we have determined that after training on an initial set of selected panels and inferring panel categories, the expert system will make no further errors on new panel problems taken

from that classification scheme.

As indicated earlier, the benefits of the system described in this paper are not to be realized by sending a naive robot into a dangerous situation to experiment with a control panel. The real benefits come from the fact that various stimulus and response components of the tasks described here occur in a variety of process control situations which are to be found in the general process control environment (e.g., a heat control or a cooling-water valve and its associated temperature indicator), and with which an autonomous robot would be capable of coping in an essentially error-free manner after initial training on a simulator. As one major test of this learning system, we intend to confront our robot with a variety of other process control systems, thereby testing the system's ability to generalize to completely new situations.

As already indicated, additional development of higher cognitive functions in the rule base will be continued in the near future. One method of accomplishing this is to use, e.g., linear regression or other mathematical analysis to determine not only what the categories are, but also which panel components are most useful in defining those categories. We intend to explore ways of having the robot make inferences about when it would be advantageous to try a button rather than a lever, or vice versa[e], and of having the robot select new

---

[e] However, these abilities already exist in the machine learning literature, so adding them to an autonomous robot's repertory would not add substantially to that body of knowledge.

training panels, thereby directing its own training.

The data presented here from the expert system performing under conditions of the two-computer simulation demonstrate the validity of our approach to creating an expert system brain for an autonomous robot which is capable of learning from past experience and generalizing. The final implementation on the actual robot will be accomplished during 1988, along with the extensions described above. Once the learning system is implemented, it will be integrated with an improved version of the navigation system described in Burks, et al (1987)[1], to provide a testbed for possible use in real world applications.

VIII. Acknowledgements

The authors wish to acknowledge the valuable help of Deanna L. Barnett in developing the CLIPS rule base, and of Stephen Killough in panel design and construction. Research for this article was sponsored by the Engineering Research Program of the US Department of Energy's Office of Basic Energy Sciences, under contract No. DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc. This research was conducted in part by an appointment to the Faculty Research Participation Program supported by the US Department of Energy and administered by Oak Ridge Associated Universities under contract No. DE-AC05-76OR00033.

IEEE Expert article -- Spelt, et al, 1988

References

1 Burks, B. L., et al., "Autonomous Navigation, Exploration,and Recognition Using the HERMIES-IIB Robot," IEEE Expert, Winter, 1987. pp. 18-27.

2 Michalski, R. S., J. G. Carbonell, & T. M. Mitchell, Machine Learning, an Artificial Intelligence Approach, vol I, Tioga Publishing Company, Palo Alto, CA, 1983; and vol II, Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1986.

3 Klahr, D., P. Langley, & R. Neches, eds. Production System Models of Learning and Development. The MIT Press, Cambridge, MA, 1987.

4 McMillan, W. W. "Classical Conditioning as an Example of Natural Models in Machine Learning," Methodologies for Intelligent Systems, Z. W. Ras & M. Zemankova, eds., Elsevier Publishing Company, New York, NY, 1987.

5 Minsky, M. "Steps toward artificial intelligence," Computers and Thought, E. A. Feigenbaum & J. Feldman, eds., McGraw-Hill, New York, 1963.

6 Laird, J. E., et al, "Chunking in SOAR: The Anatomy of a General Learning Mechanism," Machine Learning, 1986, vol. 1, no. 1, pp. 11-46.

7 Piaget, J. "Piaget's Theory," Charmichael's Manual of Child Psychology, vol 1, 3e, P. H. Mussen, ed., Wiley Publishing, New York, NY, 1970.

8 Bruner, J. S, "The Course of Cognitive Growth," American Psychologist, vol 19, 1964, pp. 1-15.

9 Buxton, W. "There's More to Interaction than Meets the Eye: Some Issues in Manual Input," User Centered System Design: New Perspectives on Human-Computer Interaction, D. A. Norman & S. W. Draper, eds., Lawrence Earlbaum Associates, Hillsdale, NJ, 1986.

10 Culbert, C. J. CLIPS Reference Manual, NASA Johnson Space Flight Center, distributed by COSMIC Software Information Services, University of Georgia, Computer Services Annex, Athens, GA, 30602.

11 Atkinson, R. C., & R. M. Shiffrin "Human memory: A proposed system and its control processes" K. W. Spence & J. T. Spence, The psychology of learning and motivation: Advances in research and theory, Vol 2, Academic Press, New York NY, 1968.

IEEE Expert article -- Spelt, et al. 1988.

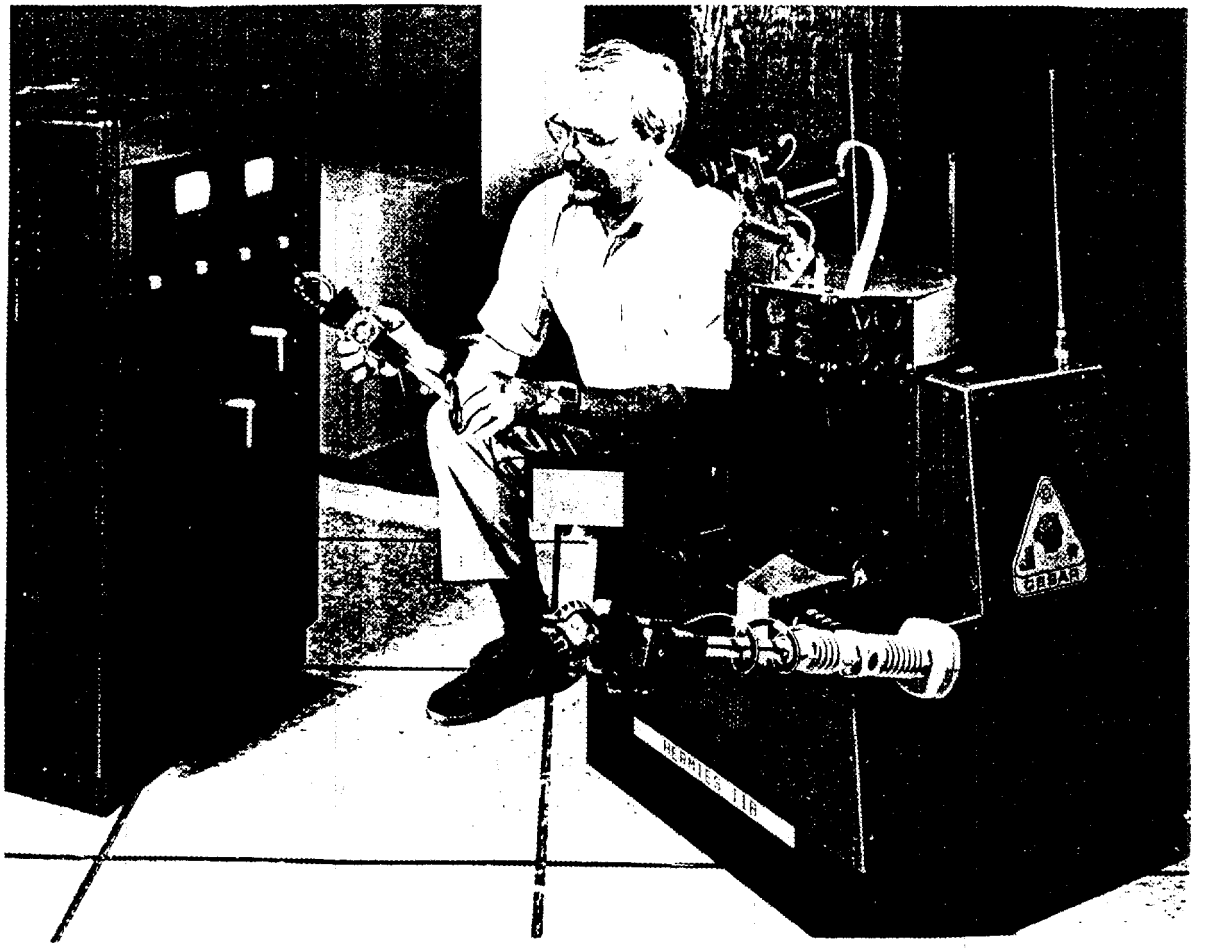


Fig. 1. HERMIES-IIB positioned in front of the CESAR Control Panel, so as to be able to manipulate the devices and read the meters. The panel's Danger Light is not shown in this view.

IEEE Expert article -- Spelt, et al, 1988

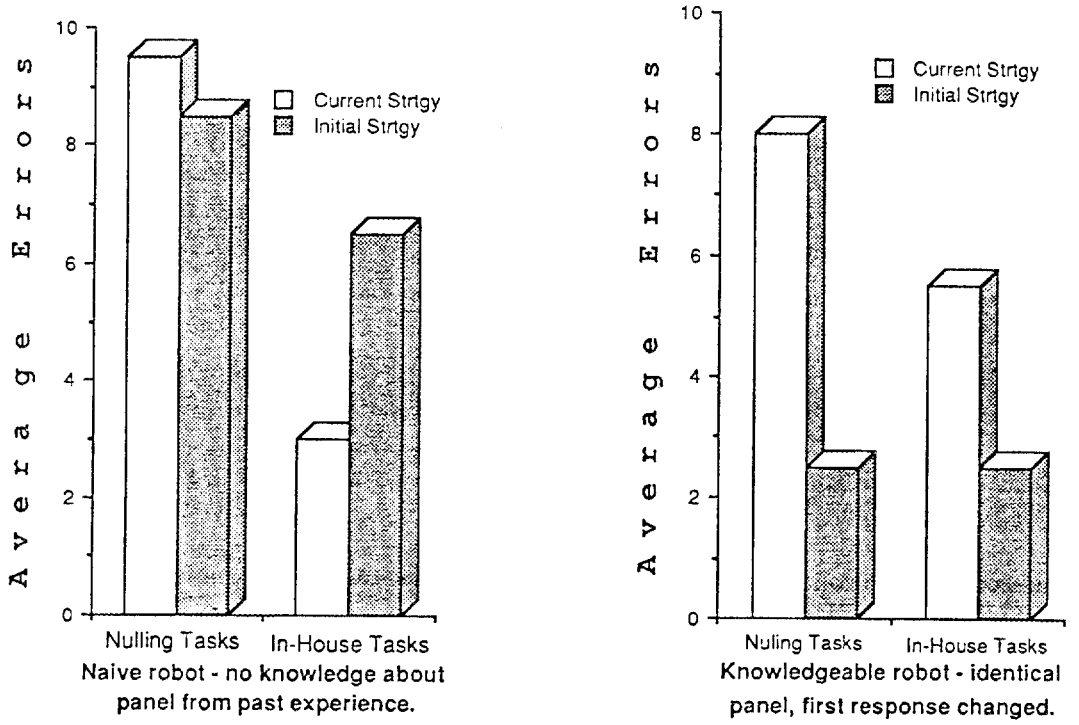
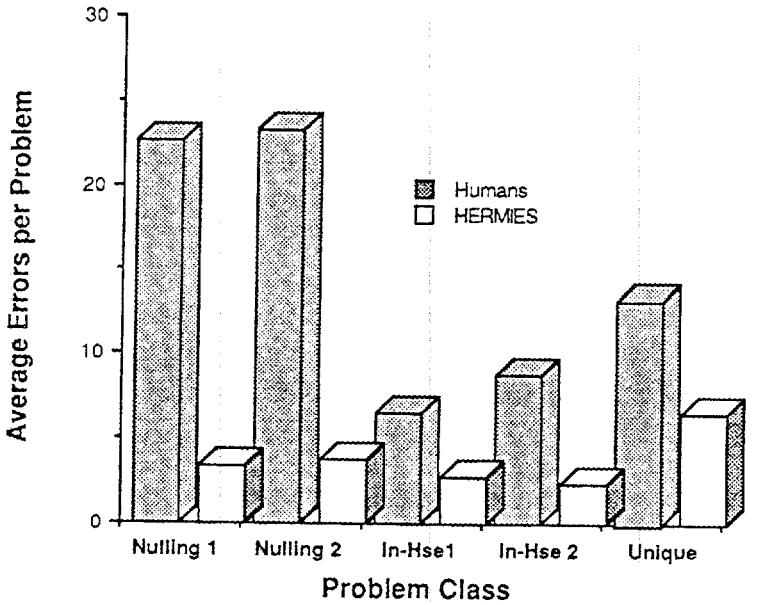


Figure 2. Average errors for Original Panel Configurations (left side) and for those configurations with the first response in the sequence altered (right side), as a function of type of Strategy.

IEEE Expert article -- Spelt, et al, 1988



AVERAGE ERRORS as a function of PROBLEM CLASS

Figure 3. Average errors for Robot and Human subjects as a function of type of problem (Problem Class).

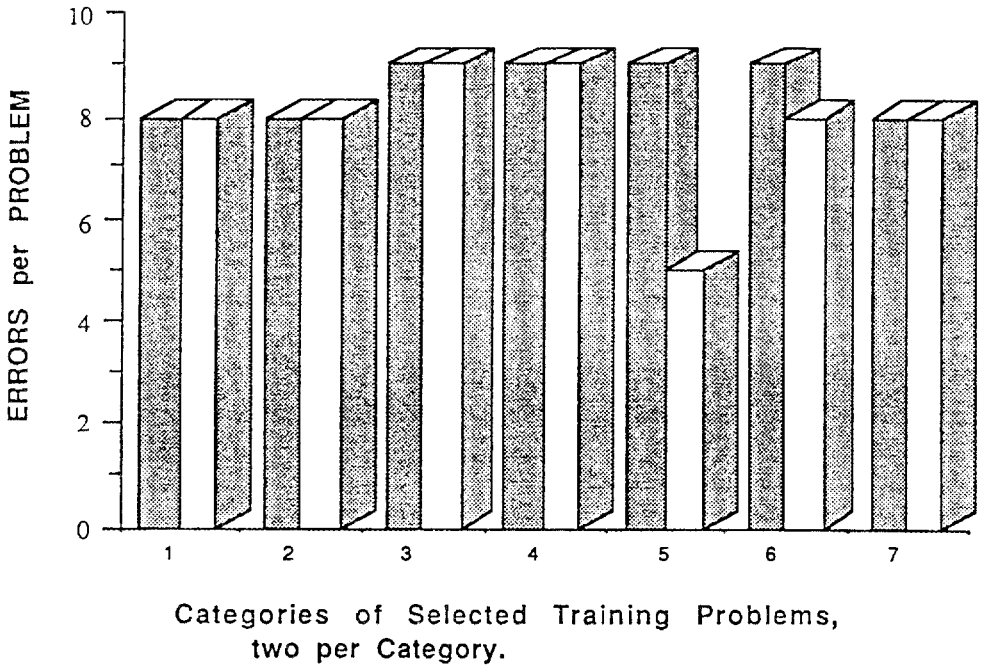


Figure 4. Errors per problem for 14 pre-selected training panel problems, two problems per category.

IEEE Expert article -- Spelt, et al, 1988

Sample Output from
a HERMIES Protocol

Communicated from HERMIES to terminal via radio link:	Meaning/Significance of message:
Setting for needle0 = high Setting for needle1 = middle Setting for lever0 = right Setting for lever1 = left	These are the initial settings read from the panel
Solving In-House #1 •	HERMIES gives problem type based on pre-programmed panel CLASSES (categories)
I have seen this IN_HOUSE-1 category before. So, I will try the sequence(s) which worked in the past.	Robot has categorized the panel type into a familiar CLASS, and will use information from LTM to solve problem

Session went on until, at Step #4, all responses available to the robot failed (due to a "malfunction" of button1). The following output illustrates the Human-Robot Symbiosis component of the Expert System:

All available responses failed on Step 4: button1, Response # = 3 button0, Response # = 5 lever0, Response # = 4 lever1, Response # = 1	Robot reports to Human "consultant" those responses which failed on current step (buttons 2 & 3 were already correctly used earlier)
What should I do now: enter q to quit or t to try another response: t	HERMIES asks for advice, receives message to try another response
Enter Response # of desired response: 3	"Consultant" tells HERMIES to try response #3 -- button1

Button1 was tried again by the robot, and the second time it lighted, indicating that it was the correct response at that step.

Table 1. Sample of dialogue taken from a protocol obtained during a training session of HERMIES-IIB using the Panel Simulator.

NEURAL NETWORKS & GRAPH K-PARTITIONING

Laurent Hérault
Jean-Jacques Niez
CEA -IRDI
Division LETI / DSYS
CENG, Avenue des Martyrs, 85X
38041 Grenoble Cedex, FRANCE

Abstract. With the emergence of neural network architectures, combinatorial optimization problems and NP-complete problems may be tackled with a new attention combining biology, physics and data processing. This paper deals with one of these problems: the graph K-partitioning. After a brief critical review of the conventional methods, we show how a particular vectorial encoding associated with this problem produces original neural network methods. Through different graph families, a comparative analysis of our approaches with one of the best conventional algorithms is developed.

1. Introduction

The graph partitioning, when it is subject to some particular constraints, is a NP-complete problem (5) having a lot of potential applications. One of them concerns the optimal assignment of distributed modules to several processors in order to minimize the cost of running a program. This cost may be money, time or some other measures of resource usage. Another application is the layout of micro-electronic systems: one wants to assign small circuits to packages (chips) of specified sizes in order to minimize one measure of interconnection between them.

1.1 Graph partitioning and computer vision as an example

This problem appears in the field of computer vision where we expect a lot of applications. The first of them concerns the perceptive grouping. In fact, salient features in an image may be described as image entities represented by the vertices of a graph. Topological relationships exist between them, the latter being represented by weighted edges.

The second application, here considered for information only and using non homogeneous graphs, concerns the stereo-correspondence. One wants to match two images of the same scene from different viewing positions in order to extract 3D-informations of the scene. The best methods need graphs to reduce the combinatory and produce valuable results as well (1) (9). Here we use the method developed by Horaud and Skordas (9). Segments are first extracted from both left and right images (see figure 1). Each segment is characterized by its position,

figure 1: Example of segment images extracted from two views of the same scene. The left image has 323 segments and the right one has 314 segments.

orientation and some topological relationships with its nearby segments. So, monocular descriptions of each image are represented as graphs (see figure 2). Each vertex represents a segment and a weighted edge between two vertices is associated to a topological relationship between two segments in the image (left of, right of, coli

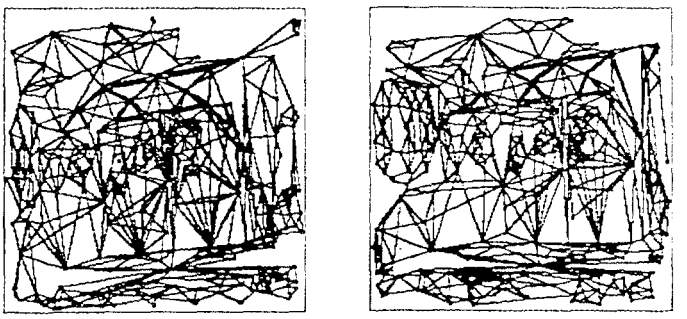


figure 2: Monocular descriptions associated to the figure 1 images. They correspond to non homogeneous graphs in which every vertex represents a segment and an edge between two vertices represents a topological relationship.The left monocular description has 323 vertices and 910 edges. The right one has 314 vertices and 874 edges.

near with, same junction as). Those two graphs are generally non homogeneous and have to be matched. But they are so complex that it is necessary to partition them into subgraphs in order to make a parallel treatment. The cost of the partition is measured by the total sum of all edge weights between vertices of distinct subsets in the partition. So far, the authors have used an arbitrary way of partitioning: they cut images in slightly overlapping windows (see figure 3). In their case, the subset number is a power of 4. One notices that the partitioning does not

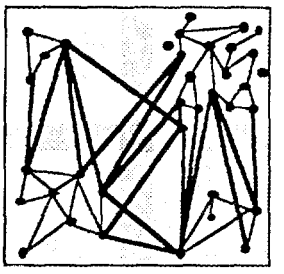


figure 3: Example of an arbitrary partition by slightly overlapping windows.

take into account the non homogeneity of the total graph. Consequently, subsets may be largely unbalanced and the interconnection cost may be very high. In fact, salient structures in the image corresponding to high local topological relationships may be broken (see figure 4). Therefore it is necessary to impose some constraints on the partition. Every subgraph of an image is matched with the entire graph of the other image. So, a first constraint must be imposed: the interconnection cost between the subgraphs must be as small as possible. On the other hand, in order to optimize the running of the parallel matching, we have to impose the following second constraint: the subsets must have specified sizes in order to have a good load balancing between processors.

1.2 Theoretical formulation of the graph K-partitioning problem

Given an undirected graph G=(V,E) of N vertices and M positively weighted edges, one wants to partition this graph into K distinct subsets of specified sizes $N_1,\ldots, N_K$ in order to to minimize the total weight of edges connecting vertices in distinct subsets. Let $A=(a_{ij})$ be its weighted adjacency matrix. One defines the density d of a graph as the ratio between M and the number of edges in a complete graph of N vertices. So, the average degree of a vertex, i.e. the average number of vertices connected to a vertex is N.d. The standard deviation of the degrees appraises the graph homogeneity. Thus we consider two graph families: the family of small standard deviation graphs, named homogeneous graphs, and the family of non homogeneous graphs.The more non homogeneous the graph, the more necessary and non obvious the partitioning. Henceforward, we will use these different graphs to test the methods here proposed. In all cases, one can prove that the interconnection cost of a perfectly balanced partition ($N_1= \ldots= N_K$) is a function of K and d which is bounded by C_{min} and C_{max} given by (see Appendix A):

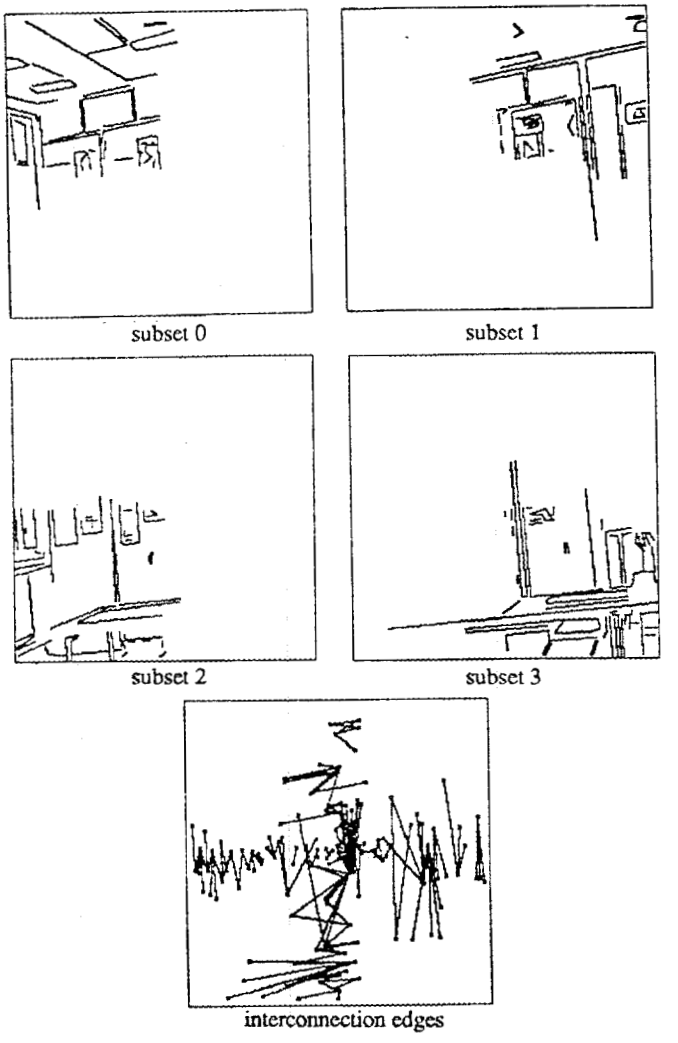


figure 4: Example of an arbitrary partition of the left monocular description. The distribution of vertices is the following: 74 vertices in subset 0, 75 vertices in subset 1, 78 vertices in subset 2 and 96 vertices in subset 3. One notices that the subsets are quite unbalanced and that the interconnection cost is high.

$$C_{min}(K) = 0 \quad \text{if} \quad K \leq \frac{N}{(N-1).d+1} \tag{1.1}$$

$$= \frac{N^2.d}{2}.\left[1 - \frac{1}{K}.\left(\frac{d-1}{N.d}.K + \frac{1}{d}\right)\right] \quad \text{otherwise}$$

and

$$C_{sup}(K) = \frac{N^2.d}{2}.\left(1 - \frac{1}{K}\right), \quad \forall K \leq N. \tag{1.2}$$

The function C_{min} is obtained by supposing that every subgraph of the partition is complete (density = 1). The function C_{sup} is obtained by supposing that the internal density of every subgraph is equal to the graph density. An exemple is shown in figure 5. In reality, if the graph is homogeneous and N much greater than K, we can estimate that the internal density of each subgraph is about K.d (see Appendix A).

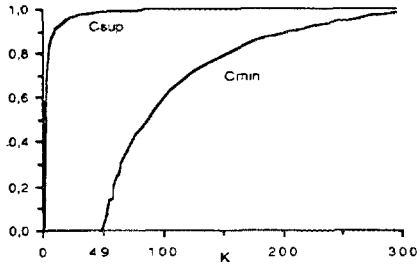


figure 5: Curves bounding the interconnection cost, divided by M, of the left graph partition shown in figure 2.

Among the homogeneous graphs, we name *regular with torus pattern*, the most homogeneous ones. The most famous examples of such graphs are the rectangular grid (see figure 6) and the hexagonal grid.

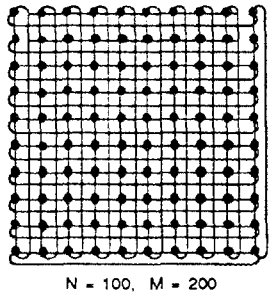


figure 6: Example of an homogeneous graph: rectangular grid with torus pattern.

Among the non homogeneous graphs, one finds the figure 2 monocular descriptions.

Two novelties are presented in this paper. The first is the formulation of an extension of neural methods (simulated annealing, Hopfield neural network, mean field theory, mean field annealing) to the manipulation of vectorial entities used as optimization variables. The second concerns the application to the graph K-partitioning problem. This paper is organized as follows. In section 2, we show that an exhaustive production of all the solutions is impossible. In section 3, we briefly review previous conventional approaches of the problem. In section 4, we develop neural methods using a new vectorial encoding.

2. Exhaustive production of solutions

Let us suppose that one wants to explore exhaustively the space of the possible distributions of N.K objects into K subsets of size N. The total number of feasible partitions is:

$$\frac{1}{K!} \cdot C_{N.K}^{N} \cdot C_{N.K-N}^{N} \cdot \ldots \cdot C_{2.N}^{N} \cdot C_{N}^{N} = \frac{(N.K)!}{K!.(N!)^{K}} . \qquad (2.1)$$

Typically, with N.K=250 and K=10, the number of configurations to study is greater than 10^{234}. Such an exploration would need thousands years of CPU time of the most powerful computers. So there is no question of developing exhaustive methods to solve such a problem: we have to develop some heuristics.

3. Previous conventional approaches

Three classical method families emerge from the sixties.

3.1 Linear programing

In the past, our problem has been considered as a linear programing problem. The first who has described the problem in such terms is Lawler (13). In this framework, a lot of methods have been developed. Among them, Donath (4) proposed to use the eigen vectors of a modified adjacency matrix; IBM researchers (16) used a Cholesky factorization of a modified adjacency matrix to iteratively improve a partition which is the solution of a linear system of N.K variables. Lukes (14) (15) used a dynamic programing procedure to generate a good partition. All these methods are not adapted to problems involving large size graphs: they are inextricable. Moreover, their parallel implementation seems to be very difficult.

3.2 Use of the Ford-Fulkerson maxflow-mincut theorem

Another approach to solve only the bipartitioning problem is to consider the graph as a network of pipes conveying some commodity from a source vertex to a sink vertex. The edge weights represent the capacities of the pipes. Stone (19) and Bokhari (2) used the maxflow-mincut theorem to solve the problem of the optimal assignment of modules on two processors. But this approach doesn't allow one to impose the sizes of the two subsets. Therefore, the problem is not NP-complete. Rao (18) studied this problem when the memory size of each processor is limited. More generally, it seems very difficult to extend successfully those methods to the K-partitioning problem.

3.3 Iterative improvements

Burnstein (3) has made a review of iterative improvement heuristics and considered two heuristic families to solve the bipartitioning problem: methods of constructing a good initial partition and methods of improving an initial partition. Very few of them produce good results because most tend to converge on the first found local minimum. Nevertheless one of them, proposed by Kernighan (10) (11), rapidly produces a very good bipartition. The idea is the following: given an initial graph bipartition which is perfectly balanced, the optimal bipartition may be obtained by interchanging a vertex group of one subset of the bipartition with a vertex group of the other subset. In order to approximate those vertex groups, one executes a sequence of vertex permutations from one subset to the other so that globally the interconnection cost decreases. Thus the algorithm allows a temporary increase of the interconnection cost. By reason of that this method keeps from being trapped at the first local minimum: one is able to leave shallow valleys of the solution landscape. The more homogeneous the graph, the better the final partition because the depth valley disparity is small. The major drawback of this approach is its sensibility to the initial partition quality.

We have extended this approach to the K-partitioning problem with $N_1 = ... = N_K = N/K$ by a dichotomic recursive procedure illustrated in figure 7 (case of the 7-partitioning).

Experimental results are presented for the 5-partitioning of the following graphs:

- homogeneous graph: regular hexagonal network of 324 vertices (see figure 8),
- non homogeneous graph: left monocular description of figure 2 (see figure 9).

Experimentally, this heuristic gives insufficient results when the desired subset number is greater than 4. In fact there is a contradiction in the dichotomic way of partitioning: first, a bipartitioning procedure tries to maximize the internal connection cost in a subset (ie. minimize an interconnection cost), and then a new bipartitioning procedure imposes to minimize a connection cost in this subset. So, the greater K, the worse the result.

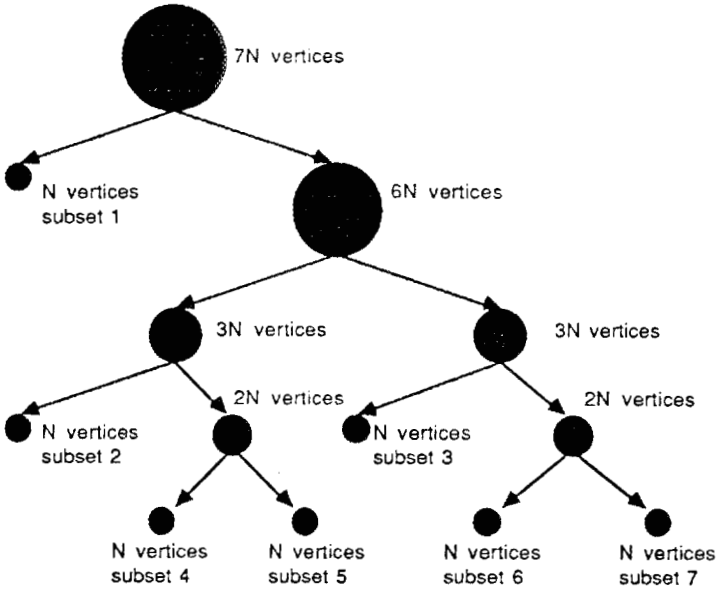


figure 7: Dichotomic procedure for the 7-partitioning using the generalized Kernighan method.

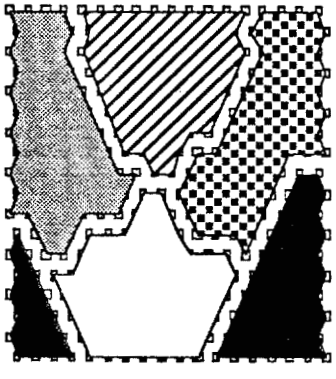


figure 8: Balanced 5-partitioning of a regular hexagonal network of 324 vertices and 901 edges provided by the generalized Kernighan method. The interconnection cost is 107 (edges are not shown).

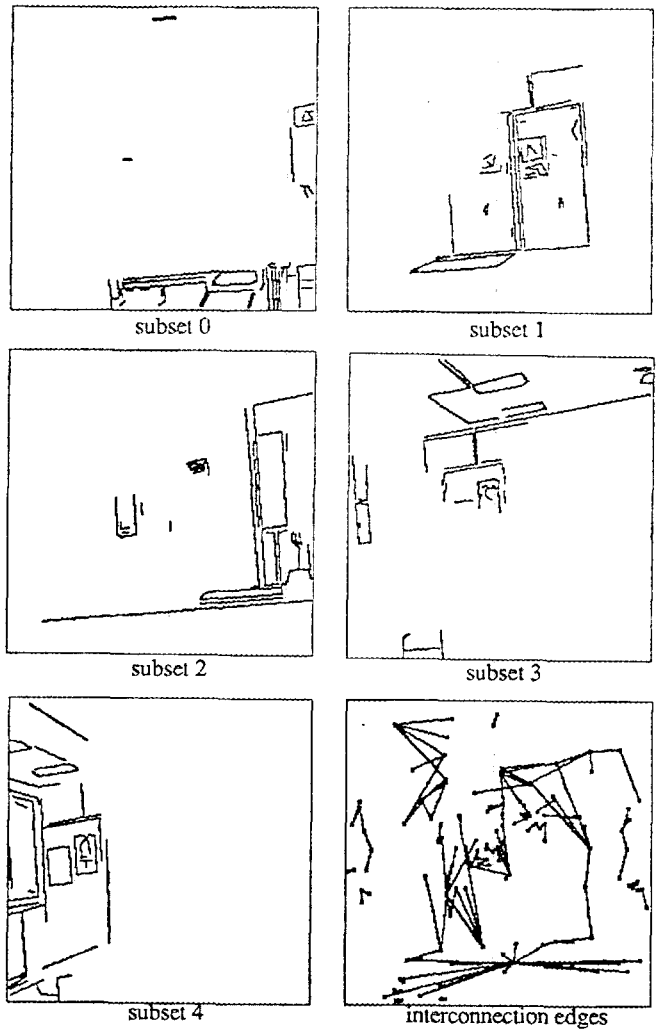


figure 9: 5-partitioning of the left non homogeneous graph of the figure 2 provided by the generalized Kernighan method but the subsets are perfectly balanced. The interconnection cost is high: 125.

4. Neural approaches

The following approaches are the result of a conjunction between biology, physics and data processing. Solving an optimization problem subject to constraints such as the graph K-partitioning is equivalent to minimizing a global quadratic energy which describes the partition state. Here we present some original neural methods to minimize such an energy. Those methods differ from previous approaches (see section 3) by the following characteristics:

- their ability to relax constraints (this is desirable for the partitionning of non homogeneous graphs),
- they can easily be implemented on massively parallel architectures such as neural networks,- the initial vertex state does not noticeably influence the final partition quality,
- they give good results whatever the number of desired subset,
- they can be easily extended to solve a lot of other combinatorial optimization problems.

A network of formal neurons is a set of highly interconnected processing elements which imitate biological neurons. A formal neuron is defined by:

- an internal state (identical to the output),
- connexions with some other neurons or with the environment,
- a non linear transition function which allows to calculate the internal state as a function of the signals received on its synaptic connections.

A synapse between two neurons is represented by a weighted connection between the output of a neuron and one input of the other. Despite the extreme simplicity of this model, collective computations with formals neurons are particularly well suited to solve combinatorial optimization problems (20) (21).

We distinguish two neural networks families: networks with binary neurons and networks with analog neurons. The following algorithms take into account this characteristic feature of the neurons.

4.1 Transcription of the optimization problem in terms of energy

We associate to every vertex a vector which defines its localization in the partition:

$$\vec{V} = [\, V^1 \ldots V^K]^t \tag{4.1}$$

where $V^k = 1$ if the vertex is in the subset k and $V^k = -1$ otherwise.

Let us calculate the partition interconnection cost. First, we notice:

$$a_{ij}.\left(\frac{V_i^k - V_j^k}{2}\right)^2 = a_{ij} \text{ if one and only one of vertices i and j is in the subset k,} \tag{4.2}$$

$$= 0 \quad \text{otherwise.}$$

Therefore, the interconnection cost between the subset k and the other subsets is:

$$\frac{1}{2}.\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}.\left(\frac{V_i^k - V_j^k}{2}\right)^2. \tag{4.3}$$

The total interconnection cost between all the subsets, which we name the interconnection energy, can be written:

$$E_{interconnection} = \frac{1}{2}.\sum_{k=1}^{K}\left\{\frac{1}{2}.\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}.\left(\frac{V_i^k - V_j^k}{2}\right)^2\right\}. \tag{4.4}$$

After some algebra, we get:

$$E_{interconnection} = -\frac{1}{8}.\sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}.V_i^k.V_j^k + \frac{K}{8}.\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}, \tag{4.5}$$

i.e.

$$E_{interconnection} = -\frac{1}{8}.\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}.\vec{V}_i.\vec{V}_j + \frac{K}{8}.\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}. \tag{4.6}$$

Now let us define an energy function which expresses the imbalance of the partition. First we notice that if the partition is perfectly balanced, then in a subset k, N_k V_i^k equal +1 and $N-N_k$ equal -1. Therefore:

$$\forall\, k \in \langle 1,K\rangle,\ \sum_{i=1}^{N} V_i^k = 2.N_k - N. \tag{4.7}$$

An imbalance measure in the subset k is defined by:

$$D_k = \left(2.N_k - N - \sum_{i=1}^{N} V_i^k\right)^2. \tag{4.8}$$

The total partition imbalance, which we name imbalance energy, is measured by:

$$E_{imbalance} = \sum_{k=1}^{K} D_k. \tag{4.9}$$

We notice:

$$\forall\, i \in \langle 1,N \rangle,\ \sum_{k=1}^{K} V_i^k = 2 - K. \tag{4.10}$$

Thus, after some algebra, one finds that the linear term of D_k is a constant for all k and we it leads:

$$E_{imbalance} = \sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N} V_i^k.V_j^k - 4.\sum_{k=1}^{K}\sum_{i=1}^{N} N_k.V_i^k - K.N^2 + 4.\sum_{k=1}^{K} N_k^2\ . \tag{4.11}$$

To find a good solution to this optimization problem, we associate to it an energy function E. The minimization of E must ensure a respect of the constraints and the minimization of the total interconnection cost. We define E as:

$$E = E_{interconnection} + \lambda/8\ .\ E_{imbalance} \tag{4.12}$$

where λ is a parameter which allows to balance the constraints. After simplifications, we get:

$$E = \frac{1}{8}.\left\{\sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N}(\lambda - a_{ij}).V_i^k.V_j^k - 4.\lambda.\sum_{k=1}^{K}\sum_{i=1}^{N} N_k.V_i^k\right\} + \frac{1}{8}.\left\{K.\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij} - \lambda.K.N^2 + 4.\lambda.\sum_{k=1}^{K} N_k^2\right\}. \tag{4.13}$$

It is clear that it is not necessary to keep the constants and the multiplicative factors in this energy. Then, we minimize the following quadratic energy:

$$E \approx \sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N}(\lambda - a_{ij}).V_i^k.V_j^k - 4.\lambda.\sum_{k=1}^{K}\sum_{i=1}^{N} N_k.V_i^k\ . \tag{4.14}$$

In order to statistically give the same importance to the balance constraint and to the interconnection cost minimization constraint, one can estimate the value of the parameter λ (using a similar kind of approach as Kirkpatrick (12)):

$$\lambda \approx \alpha.\frac{1}{N^2}.\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij} \tag{4.15}$$

where α is an adjustable parameter always around 1.

We empirically notice that the partitioning of homogeneous graphs with a parameter α close to 1 provides a partition with an excellent balance. This can be explained by the fact that the energy landscape is quite smooth: the valley depth disparity is small. Therefore, the optimization algorithm easily moves from one energetic valley to another one until the obtention of a well balanced partition. On the contrary, the partitioning of non homogeneous graphs needs a balance parameter greater than 1 because the valley depth disparity grows with the non homogeneity of the graph. Typically, good results are obtained with α close to 2.

In the following, experimental results will be given by supposing that the subsets have the same size ($N_1 = ... = N_K$).

4.2 Partition energy minimization by a network with binary neurons

In this neural network family, the internal state (i.e the output) of each neuron is binary: the non linear transition function associated to a neuron is a Heavyside type function.

4.2.1 Hopfield network with binary neurons

The interconnection network of the Hopfield model is complete (see figure 10). The synaptic connection between the neuron i and the neuron j is weighted by T_{ij} which is positive (excitatory synapse) or negative (inhibitory sy - napse). The neuron output is a function of its inputs:

$$V_i = f\left(\sum_{j=1}^{N} T_{ij}.V_j + I_i\right). \tag{4.16}$$

The transition function f is defined by the following:

$$\begin{aligned} &\text{if } x<0 \text{ then} && f(x) = -1, \\ &\text{otherwise} && f(x) = 1. \end{aligned} \tag{4.17}$$

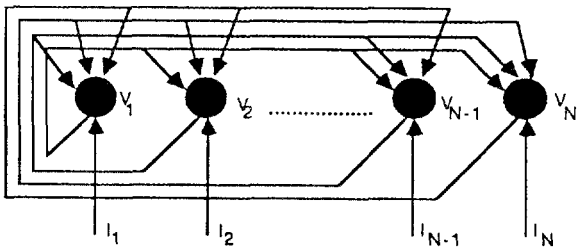


figure 10 : Hopfield neural network. The interconnection graph is complete.

Hopfield has shown (8) that in the case of an asynchronous dynamics, a symmetrical matrix T with 0 diagonal elements drives the system to stable states in which the outputs of all neurons are either +1 or -1. These stable states of the network correspond to the local minima of the quantity, which we call the energy of the system:

$$E = -\frac{1}{2}.\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}.V_i.V_j - \sum_{i=1}^{N} I_i.V_i \tag{4.18}$$

where V_i is the output of the ith neuron and I_i is the externally supplied input to the ith neuron.

Let us associate to our optimization problem an Hopfield network having a matrix organization of N.K neurons in which the output of the $(i,k)^{th}$ neuron expresses the V_i^k value and is either +1 or -1 (see figure 11).

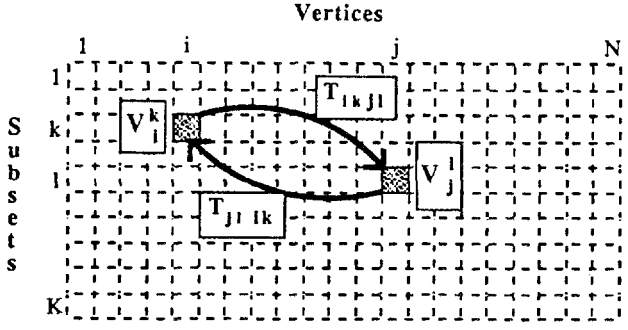


figure 11: Matrix structure of an Hopfield network adapted to the graph K-partitioning problem.

This network is a neural transcription of the vectorial representation previously defined. To each neuron is associated a processing element with several inputs and one output connected to the other neurons (see figure 12). A coefficient of T represents the synaptic weight between two neurons. The output (internal state) of the $(i,k)^{th}$ neuron is:

$$V_i^k = f\left(\sum_{j=1}^{N}\sum_{l=1}^{K} T_{ik,\,jl}.V_j^l + I_{ik}\right). \tag{4.19}$$

We must add to the energy function associated to our optimization problem (equation 4.14) an energy term which takes into account the structural organization of the network (see figure 11). Therefore, we have to minimi

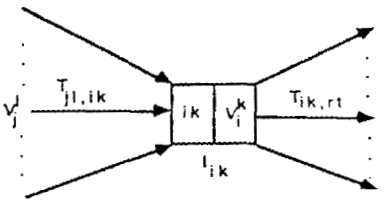


figure 12: Details of a formal neuron of the figure 13

ze the energy:

$$E = \sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N}(\lambda - a_{ij}).V_i^k.V_j^k - 4.\lambda.\sum_{k=1}^{K}\sum_{i=1}^{N}N_k.V_i^k + B.\sum_{i=1}^{N}\left(2 - K - \sum_{k=1}^{K}V_i^k\right)^2. \tag{4.20}$$

The term relative to B is an energy term which is minimum when only one component V_i^k of each neuron vector V_i characterizing a graph vertex is equal to +1. We get:

$$E = \sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N}(\lambda - a_{ij}).\delta_{kl}.V_i^k.V_j^k - 4.\lambda.\sum_{k=1}^{K}\sum_{i=1}^{N}N_k.V_i^k$$
$$+ B.\sum_{i=1}^{N}\sum_{j=1}^{N}\left(2 - K - \sum_{k=1}^{K}V_i^k\right).\left(2 - K - \sum_{l=1}^{K}V_j^l\right).\delta_{jl}\,. \tag{4.21}$$

After some algebra, we find:

$$E = -\frac{1}{2}.\sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N}[-2.(\lambda - a_{ij}).\delta_{kl} - 2.B.\delta_{ij}].V_i^k.V_j^l$$
$$- \sum_{i=1}^{N}\sum_{k=1}^{K}[2.B.(2 - K) + 4.\lambda.N_k].V_i^k + B.N.(2-K)^2\,. \tag{4.22}$$

Once again, it is not necessary to keep the constant term. Our energy can be written as an Hopfield energy:

$$E = -\frac{1}{2}.\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{K}\sum_{l=1}^{K}T_{ik,\,jl}.V_i^k.V_j^l - \sum_{i=1}^{N}\sum_{k=1}^{K}I_{ik}.V_i^k \tag{4.23}$$

where:

$\forall\ (i,j) \in <1,N>^2,\ \forall\ (k,l) \in <1,K>^2,$

$$T_{ik,\,jl} = -2.(\lambda - a_{ij}).\delta_{kl} - 2.B.\delta_{ij}, \tag{4.24}$$

$$I_{ik} = 2.B.(2 - K) + 4.\lambda.N_k\,. \tag{4.25}$$

The excitatory part of the synaptic weight (positive term) tends to put highly interconnected vertices in the same subset but the inhibitory part (negative terms) imposes the balance and the disjunction constraints on the subsets. One notices that the matrix T of synaptic weights is symmetrical:

$$\forall\ (B,\lambda) \in \Re^{+^2},\ T_{ik,\,jl} = T_{jl,\,ik}. \tag{4.26}$$

Only the two parameters λ and B are necessary two calculate the matrix T. Additionally, with the relation:

$$B = -\lambda, \tag{4.27}$$

all the diagonal coefficients of the matrix T are 0 (since all the diagonal coefficients of the adjacency matrix are 0). With this condition, one can theoretically show (8) that the system converges to the nearest local minimum of the configuration hypercube. Moreover, the value of B given by (4.27) seems to be a good one because the parameters then globally balance the constraints of the problem. Then λ is the only parameter to be determined and is easily approximated (equation 4.15).

The running on a conventionnal sequential architecture would be very expensive in terms of CPU time. So we have not experimentally validated this method.

4.2.2 Simulated annealing

A good way to find low energy states of a complex physical system such as a solid is to heat the system up to some high temperature, then cool it slowly. This process, called annealing, forces the system evolution into regions of low energy, while not getting trapped in higher-lying local minima. Geman (6) has shown that with an infinite initial temperature and by using an exponential law for the temperature decreasing, an absolute energy minimum is reached in an exponential number of iterations. The idea of the simulated annealing is to express those concepts in terms of an algorithm. So, we identify the energy function of the system to be optimized with the energy of a physical system.

Let us consider:

- a list of feasible elementary transformations which determine the energy landscape of our problem (the topology),
- an initial configuration of the system,
- a law of the temperature decreasing.

The smaller the temperature, the more rigid the system (there is a small number of elementary transformations which are operated) and the more deterministic the system evolution. Kirkpatrick (12) has developed this approach in the case of the graph bipartitioning problem. We extend this method to the K-partitioning problem by using the previously defined vectors.

Given the global energy to be minimized (equation 4.14), let us caculate the energy variation associated to an elementary transformation. We define an elementary transformation as the move of a vertex i from a subset k to a subset l. The total number of possible elementary transformations is N.(K-1). It leads:

$$\Delta E_i^{k\to l} = E\,(V_i^k = -1,\ V_i^l = 1) - E\,(V_i^k = 1,\ V_i^l = -1). \tag{4.28}$$

After some algebra, we obtain:

$$\Delta E_i^{k\to l} = 4.\sum_{j=1,\ j\neq i}^{N} (\lambda - a_{ij}).(V_j^l - V_j^k) + 8.\lambda.(N_k - N_l)\ . \tag{4.29}$$

New states of the system are generated by applying a set of elementary transformations to the system. Each elementary transformation is accepted or rejected using the following criterium:

if $\Delta E_i^{k\to l} \leq 0$, then accept the move,

otherwise accept the move with the probability

$$P\,(\Delta E_i^{k\to l}) = \exp\left(-\frac{\Delta E_i^{k\to l}}{T}\right), \tag{4.30}$$

where T is the temperature parameter.

Some curves representing P(x,T) are shown in figure 13. One verifies that the acceptance probabilty decreases as the temperature.

The principle of the simulated annealing algorithm is the following: the system is put in a high temperature environment. At this temperature is applied a sufficiently long sequence of random elementary transformations (Markov chain) to reach the equilibrium at this temperature. Then, the ambient temperature is slightly decreased and a new sequence of random moves is applied. So, the system converges slowly to a minimal energy state. This process is iterated until the system is frozen, in other words when there are not enough global significant energy improvements. By analogy to spin glass physics, one takes as a good initial temperature:

$$T_0 = (N.d)^2 \tag{4.31}$$

where d is the graph density. Here, we notice that the Markov chain length of elementary transformations which is necessary to obtain the equilibrium at a fixed temperature closely depends on the graph homogeneity. The more homogeneous the graph, the smaller the necessary length of the Markov chains because the slopes of the relevant energetic valleys are then more abrupt. This will be visualized in section 4.3.2. The simulated annealing algorithm can be found in Appendix B.

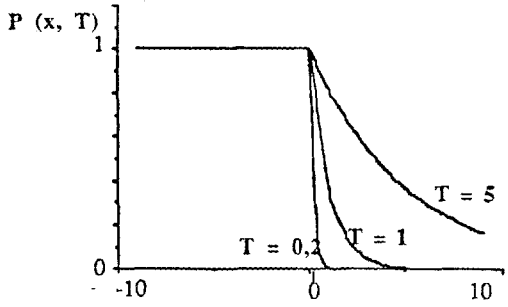


figure 13: Simulated annealing: acceptance probabilities of an elementary transformation as a function of the associated energy variation and of the temperature T.

Experimental results are given for the 5-partitioning of the following graphs:

- homogeneous graph and α=1: regular hexagonal network (see figure 14-a). Figure 14-b shows the evolutions of the energies (interconnection energy, imbalance energy and total energy) as a function of the number of temperature steps. In figure 14-c one shows the corresponding evolution of the temperature and of the number of accepted elementary transformations. One can see that in average those energies decrease with the temperature. Those curves are highly non linear: the system suddenly freezes in a certain range of temperatures.

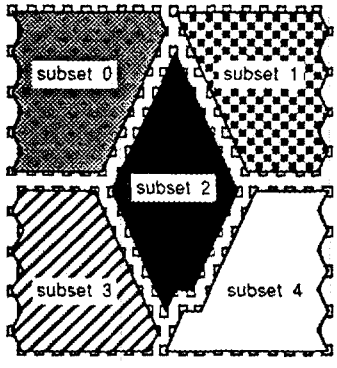


figure 14-a: Balanced 5-partitioning with α=1 of a regular hexagonal network of 324 vertices and 901 edges provided by the simulated annealing algorithm. The interconnection cost is 85 (edges are not vizualized). The imbalance energy is 99,20. The total energy is 85,21. The initial temperature is 40 and the final temperature is 0,378. The number of temperature steps is 109. The distribution of vertices is the following: 69 vertices in subset 0, 65 vertices in subset 1, 62 vertices in subset 2, 64 vertices in subset 3 and subset 4.

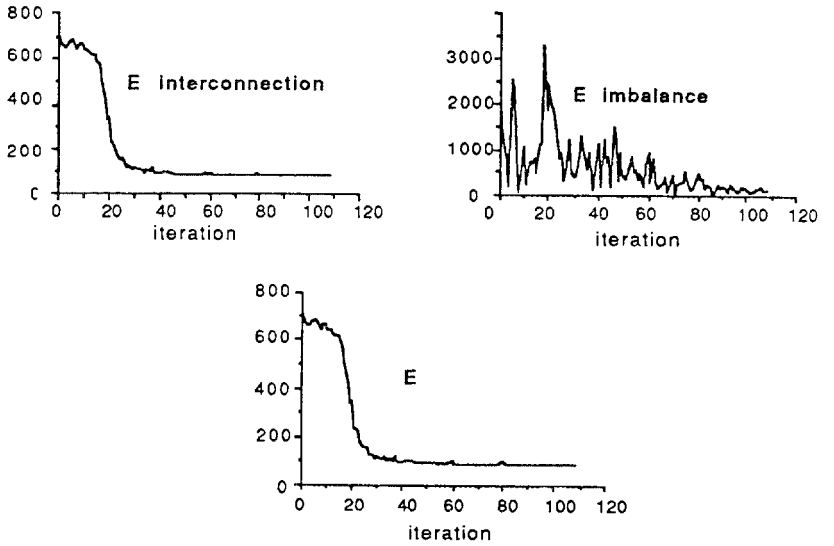


figure 14-b: Evolution of the system energies as a function of the temperature step number.

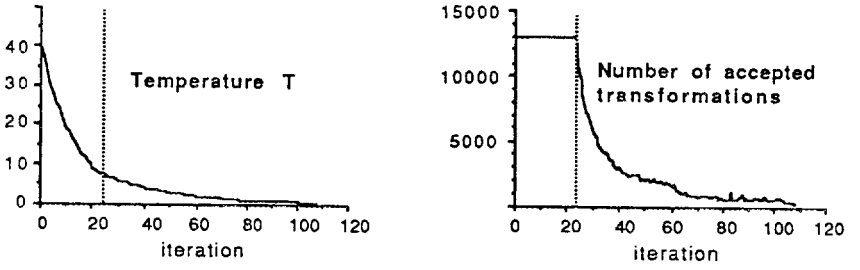


figure 14-c: Evolution of the temperature and of the number of accepted transformations as a function of the temperature step number.

- non homogeneous graph and α=1: left monocular description of figure 2 (see figure 15-a, 17-b and 17-c). One notices that α=1 produces a bad imbalance energy: α must be greater.

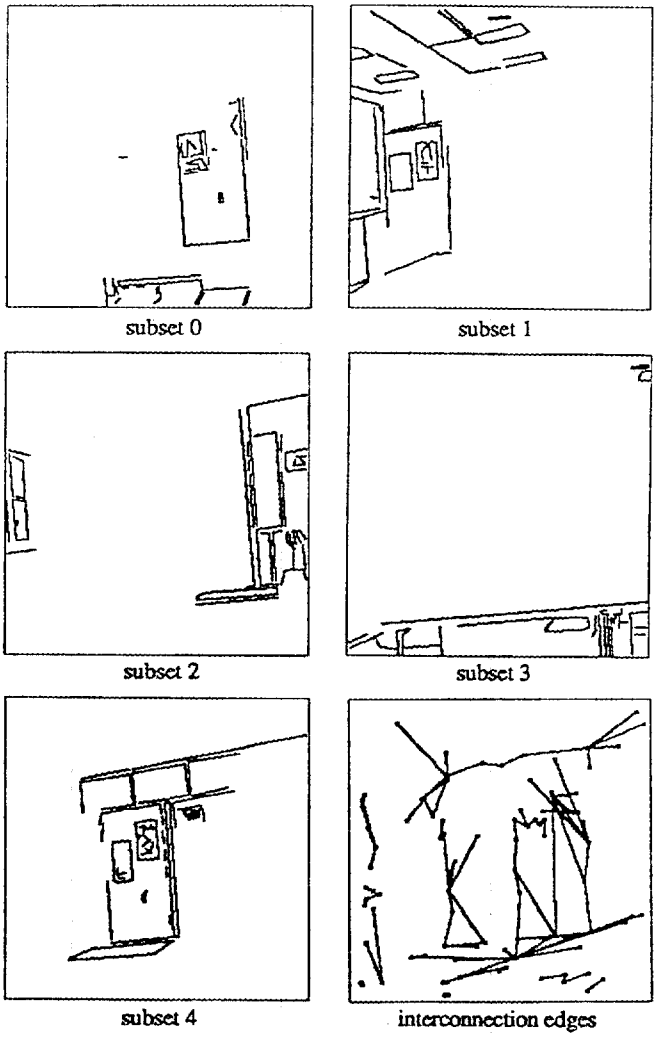


figure 15-a: Balanced 5-partitioning with α=1 of the left non homogeneous graph of the figure 2 provided by the simulated annealing algorithm. The interconnection cost is 85, the imbalance energy 3572,8 the total energy 92,81. The initial temperature is 40 and the final temperature 0,714. The number of temperature steps is 88. The distribution of vertices is the following: 53 vertices in subset 0, 69 vertices in subset 1, 67 vertices in subset 2, 48 vertices in subset 3 and 86 vertices in subset 4.

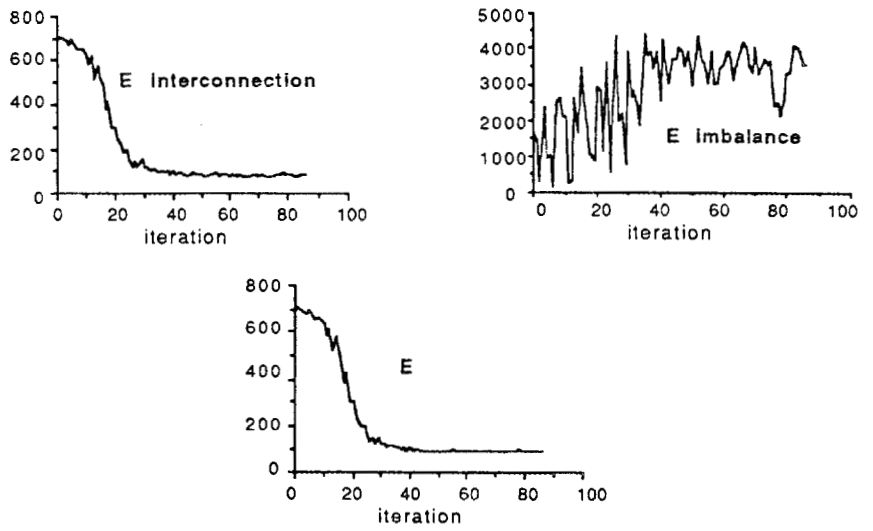


figure 15-b: Evolution of the system energies as a function of the temperature step number.

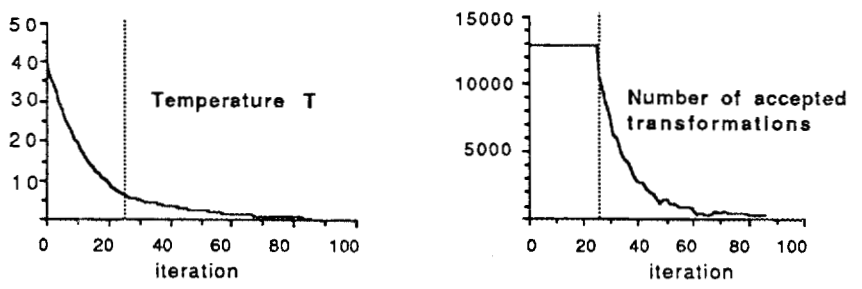


figure 15-c: Evolution of the temperature and of the number of accepted transformations as a function of the temperature step number.

- non homogeneous graph and α=2: left monocular description of figure 2 (see figure 16-a, 18-b and 18-c). The 5-partition and the energies produced with such a value of α are excellent.

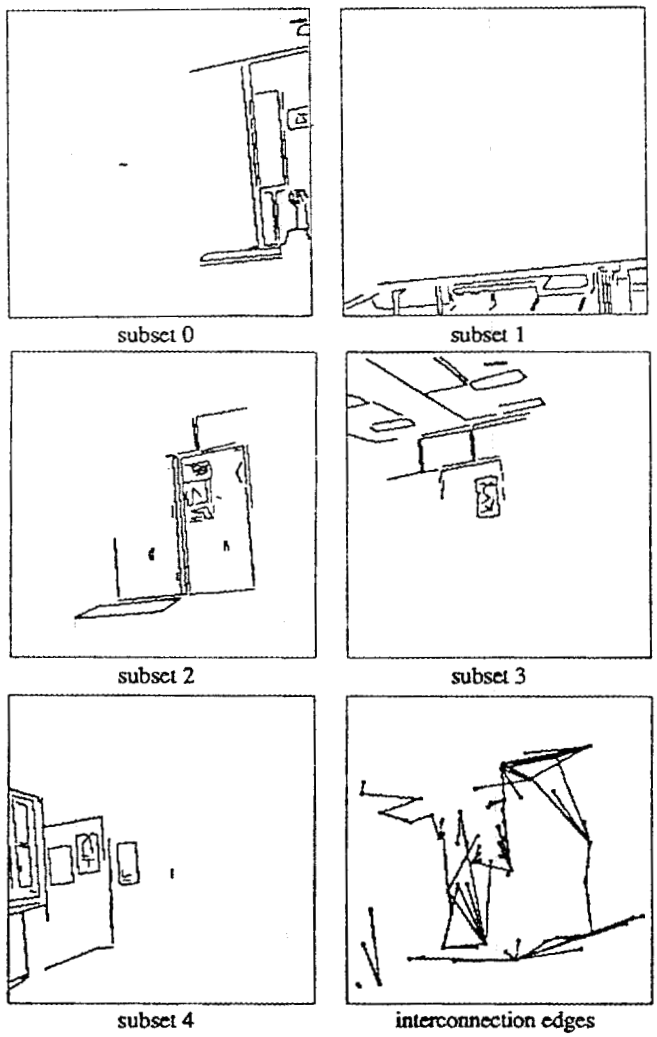


figure 16-a: Balanced 5-partitioning with α=2 of the left non homogeneous graph of the figure 2 provided by the simulated annealing algorithm. The interconnection cost is 74, the imbalance energy 116,8 the total energy 74,51. The initial temperature is 40 and the final temperature 0,172. The number of temperature steps is 127. The distribution of vertices is the following: 63 vertices in subset 0 and 1, 68 vertices in subset 2, 67 vertices in subset 3 and 62 vertices in subset 4.

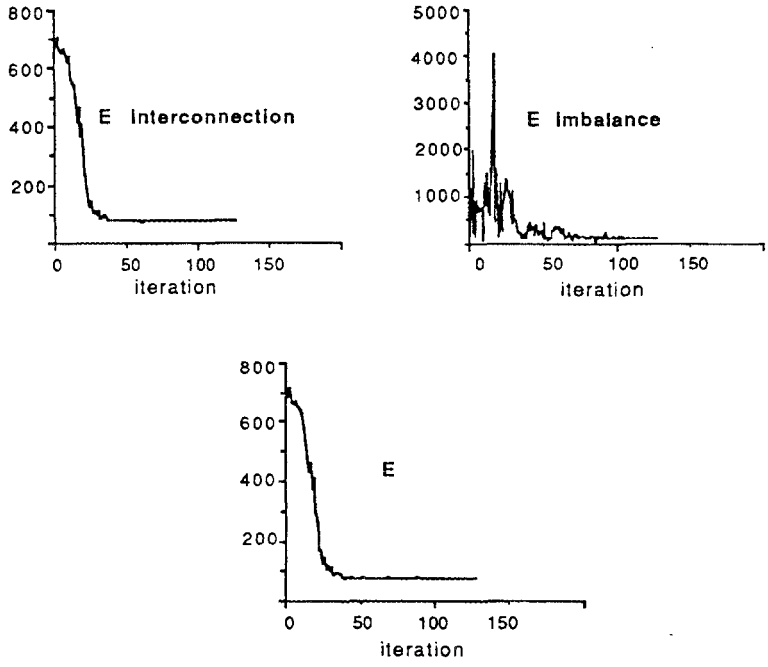


figure 16-b: Evolution of the system energies as a function of the temperature step number.

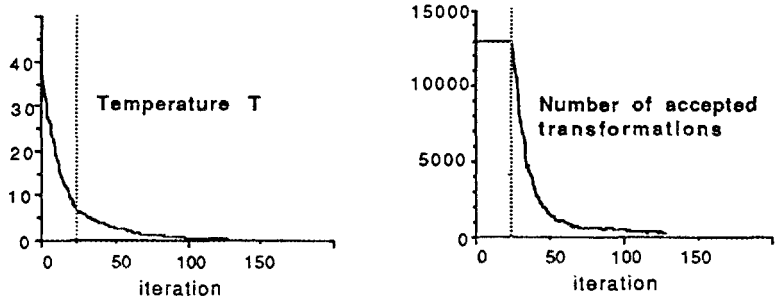


figure 16-c: Evolution of the temperature and of the number of accepted transformations as a function of the temperature step number.

Those results are to be compared with those provided previously by the generalized Kernighan method (see section 3.3): the simulated annealing algorithm experimentally improves by about 20% the interconnection cost and thus the global energy of the system.

The main drawback of this method is that the running time necessary to converge on a conventional sequential computer is very high (about 40 CPU hours on a VAX 11/780 for the graphs of figures 16 and 17). Nevertheless, we notice that the time complexity linearly increases as the problem size; in fact, it is imposed by the Markov chain length which is in o(N.K). In order to speed up the partitioning, we will use another approach, the so called mean field annealing algorithm (see section 4.3.3). Contrary to the stochastic and sequential nature of the simulated annealing, the system evolution here is deterministic and massively parallel. Results are nearly as good as those

provided by the simulated annealing algorithm and the CPU time is typically divided by 10 or 20 for simulations on a classical conventionnal computer (VAX 11/780).

4.3 Partition energy minimization by a network with analog neurons

In this neural network family, the internal state (i.e. the output) of each neuron has an analog internal state.

4.3.1 Hopfield network with analog neurons

One associates a numerical noise to the boolean transition function previously defined (see section 4.2.1). One would like to mimic the effect of this noise and additionally control the convergence process. So, one considers an Hopfield network with analog neurons as defined in (7). The new associated transition function is a sigmoid and depends on a parameter T which mimics the noise. One can take as a transition function (see figure 17):

$$f_T = \mathrm{th}\,(x/T). \tag{4.32}$$

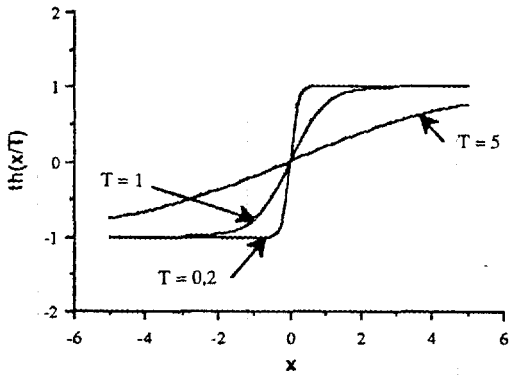


figure 17: Examples of possible transition functions associated to an analog neuron.

One forces T to tend to 0 during the convergence process. At this limit, we obtain the previous model (see section 4.2.1). As previously, one can show that the Hopfield energy defined in the part 4.2.1 converges to a minimum (7). A thresholding is made at the end of the process, when one estimates that the network has converged, by using the formula:

$$\forall\, i \in \langle 1,N\rangle,\ \forall\, k \in \langle 1,K\rangle,$$

$$V_i^k(t+1) = f_{T\to 0}\,(\,V_i^k(t)\,). \tag{4.33}$$

In this case, one has to add to the energy associated to our optimization problem the additional energy which tends to force the neuron outputs to be +1 or -1:

$$C.\sum_{i=1}^{N}\sum_{k=1}^{K}(1 - V_i^k).(1 + V_i^k). \tag{4.34}$$

This energy term is minimum when the neuron outputs are either +1 or -1. Then the global energy to minimize becomes:

$$E = -\frac{1}{2}.\sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N}[-2.(\lambda - a_{ij}).\delta_{kl} - 2.B.\delta_{ij} + 2.C.\delta_{ij}.\delta_{kl}].V_i^k.V_j^l$$
$$- \sum_{i=1}^{N}\sum_{k=1}^{K}[2.B.(2 - K) + 4.\lambda.N_k].V_i^k + B.N.(2-K)^2\,. \tag{4.35}$$

This energy can be written as an Hopfield energy by using the synaptic weights:

$\forall (i,j) \in <1,N>^2, \forall (k,l) \in <1,K>^2,$

$$T_{ik,jl} = -2.(\lambda - a_{ij}).\delta_{kl} - 2.B.\delta_{ij} + 2.C.\delta_{ij}.\delta_{kl} \ , \tag{4.36}$$

$$I_{ik} = 2.B.(2 - K) + 4.\lambda.N_k \ . \tag{4.37}$$

It is equivallent to add an excitatory synaptic term which tends to force every neuron output to be +1 or -1. The matrix of the synaptic weights is symmetrical. Additionally, with the condition:

$$C = B + \lambda \ , \tag{4.38}$$

all the diagonal coefficients of the matrix are 0. Therefore, the system converges to a minimum (7). With this condition, two parameters have to be fixed: B and λ.

As in the case of an Hopfield network with binary neurons, we have not validated this method on a conventional computer because of the large CPU times which are expected.

In the two types of Hopfield networks (with analog or binary neurons) and with the condition 4.27 or 4.38, every neuron is connected to N-1+K other neurons. Therefore the total number of synaptic connections in the network is N.(N+K-1) and thus is proportionnal to N^2. Consequently a small subset number compared to the vertex number doesn't noticeably influence the running time.

4.3.2 Mean field theory

The main drawback of the simulated annealing is its large running time on a conventional sequential computer (see section 4.2.2). A neural approach coming from statistical mecanics and named mean field theory (MFT) has been developped (17) to solve much more quickly some optimization problems. Here, the data used are scalar entities. We extend this approach to solve optimization problems having a lot of degrees of freedom such as the graph K-partitioning by using the previously defined vectorial entities (equation 4.1). Contrary to the simulated annealing , the convergence process is perfectly deterministic and is controlled by a dynamic system. At every temperature, a solution of this system is directly related to the vertex membership probabilities (between -1 and +1) of a subset in the K-partition. We show that this method gives very good results in a smaller CPU time than the one which is necessary in the simulated annealing. Additionally, it is intrinsically massively parallel by nature.

Let us define, for all vertex i and for all subset k:

$$h_i^k = -\sum_{j=1}^{N} (\lambda - a_{ij}).V_j^k \ , \tag{4.39}$$

$$h_{ext}^k = 4.\lambda.N_k \tag{4.40}$$

and

$$H_i^k = h_i^k + h_{ext}^k \ . \tag{4.41}$$

h^k_i may be considered as the kth component of the field vector created on the vertex i by the other k^{th} spins associated to the graph vertices. h^k_{ext} may be considered as the kth component of the external field in which the system is plunged. Thus H^k_i is the kth component of the total field existing on the vertex i. Then the system energy (equation 4.14) can be written:

$$E = -\sum_{i=1}^{N} \vec{H}_i.\vec{V}_i \ . \tag{4.42}$$

Let us consider a vertex which is isolated from the others which are supposed to be fixed. Then the energy associated to the vertex is:

$$E = -\sum_{k=1}^{K} H^k.V^k = -\vec{H}.\vec{V}. \tag{4.43}$$

In the spin vector of the vertex i, only one component is +1 and the others are -1. We can write the partition function associated to the mean behaviour of a vertex as:

$$Z = \sum_{\vec{v}} \exp\left\{-\frac{1}{T}.E(\vec{V})\right\} = \sum_{\vec{v}} \exp\left\{\frac{1}{T}.\vec{H}.\vec{V}\right\} . \tag{4.44}$$

In this expression, the configurations of minimum energy are predominant. After some algebra, we obtain:

$$Z = \sum_{k=1}^{K} \exp\left\{\frac{1}{T}.\left(H^k - \sum_{l=1,\, l\neq k}^{K} H^l\right)\right\} . \tag{4.45}$$

The mean vector of spins associated to a graph vertex has the following k^{th} component:

$$<V^k> = \frac{1}{Z}.\left[\exp\left\{\frac{1}{T}.\left(H^k - \sum_{l=1,\, l\neq k}^{K} H^l\right)\right\} - \sum_{l=1,\, l\neq k}^{K} \exp\left\{\frac{1}{T}.\left(H^l - \sum_{m=1,\, m\neq l}^{K} H^m\right)\right\}\right] . \tag{4.46}$$

After simplifications, it leads:

$$<V^k> = \frac{\exp\left\{\frac{2}{T}.H^k\right\} - \sum_{l=1,\, l\neq k}^{K} \exp\left\{\frac{2}{T}.H^l\right\}}{\sum_{l=1}^{K} \exp\left\{\frac{2}{T}.H^l\right\}} \tag{4.47}$$

$$= \frac{2}{\sum_{l=1}^{K} \exp\left\{\frac{2}{T}.(H^l - H^k)\right\}} - 1 \tag{4.48}$$

The mean field approximation consists in supposing that the field seen by a vertex is the mean field created on this vertex by the other vertices. Then, for all vertex i, we get:

$$<V_i^k> = \frac{2}{\sum_{l=1}^{K} \exp\left\{\frac{2}{T}.\left[4.\lambda.(N_l - N_k) - \sum_{j=1}^{N} (\lambda - a_{ij}).(<V_j^l> - <V_j^k>)\right]\right\}} - 1 . \tag{4.49}$$

The solutions of the equation (4.49) can be iteratively obtained thanks to the following equations:

$\forall\, i \in <1,N>,\ \forall\, k \in <1,K>,$

$$V_i^{k\,new} = \frac{2}{\sum_{l=1}^{K} \exp\left\{\frac{2}{T}.\left[4.\lambda.(N_l - N_k) - \sum_{j=1}^{N} (\lambda - a_{ij}).(V_j^{l\,old} - V_j^{k\,old})\right]\right\}} - 1 . \tag{4.50}$$

The desired values are also solutions of the dynamic system:

$\forall\, i \in <1,N>,\ \forall\, k \in <1,K>,$ (4.51)

$$\tau.\frac{d<V_i^k(t)>}{dt} = -<V_i^k(t)> + \frac{2}{\sum_{l=1}^{K} \exp\left\{\frac{2}{T}.\left[4.\lambda.(N_l - N_k) - \sum_{j=1}^{N} (\lambda - a_{ij}).(<V_j^l(t)> - <V_j^k(t)>)\right]\right\}} - 1.$$

Two running modes are possible. A new step in a synchronous running, every V^k component of each vertex is simultaneously updated by using the other V^k values which have been calculated at the previous step. In the case of an asynchronous running, one calculates the V^k associated only to one node. The V^k values of the other vertices will be calculated in another step. We can logically think that an asynchronous running mode produces best results because the convergence process is less subject to the oscillations which frequently exist in a synchronous

running mode. The algorithm is given in Appendix C.

Let us make some remarks about this algorithm. One can see that the choice of the final partition is obvious. When one estimates that the system has converged, one chooses for every vertex i the greatest V_i^k among all the positive components V_i^k. The corresponding component has a probability greater than 50% to be +1. All the other components are put to -1.

To determine the initial configuration of the system, let us notice that if $V^k(t=0) = 2/K-1$ for all vertex components, then the components V_i^k are solutions of the dynamic system. Practically, one determines the initial configuration of the system by adding noise on this trivial solution: for instance, the V_i^k are randomly choosen between the two values $(2/K-1-10^{-5}, 2/K-1+10^{-5})$.

We have tested this algorithm in the asynchronous running mode. We notice that the components V_i^k tend to experiment a damped oscillation during the convergence process. Thus it is our interest to scan the graph vertices a lot of times. The minimum scan number necessary to have a good solution depends on the graph homogeneity: the more non homogeneous the graph, the smaller the necessary scan number because the energetic slopes are then more abrupt. Practically, N/2 scans are sufficient for non homogeneous graphs such as monocular descriptions (see figure 2). As for homogeneous graphs, the system converges in less than N scans. We verify this assertion with the 5-partitioning of the following graphs:

- homogeneous graph and $\alpha=1$: regular hexagonal network. We show the partition provided by the mean field algorithm at two temperatures. In figure 18-a, the ambient temperature is 2. Figure 18-b shows the evolutions in each subset (k fixed) of the components V_i^k as a function of the scan number. The system needs less than N scans to converge. In figure 18-c and 18-d, the ambient temperature is 4. The partition energies of the figure 18-a and 18-c partitions are comparable but the solutions correspond to different valleys in the energy landscape.

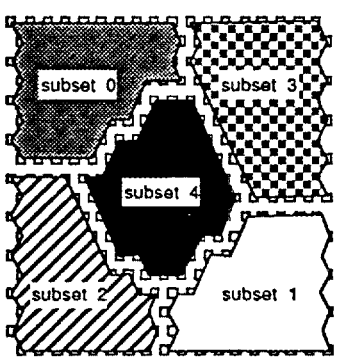


figure 18-a: Balanced 5-partitioning with $\alpha=1$ of a regular hexagonal network of 324 vertices and 901 edges provided by the mean field approximation (edges are not visualized). The ambient temperature is 2. The interconnection cost is 88, the imbalance energy 107,2 and the total energy 88,23. The distribution of vertices is the following: 66 vertices in subset 0, 64 vertices in subset 1, 65 vertices in subset 2, 68 vertices in subset 3, 61 vertices in subset 4.

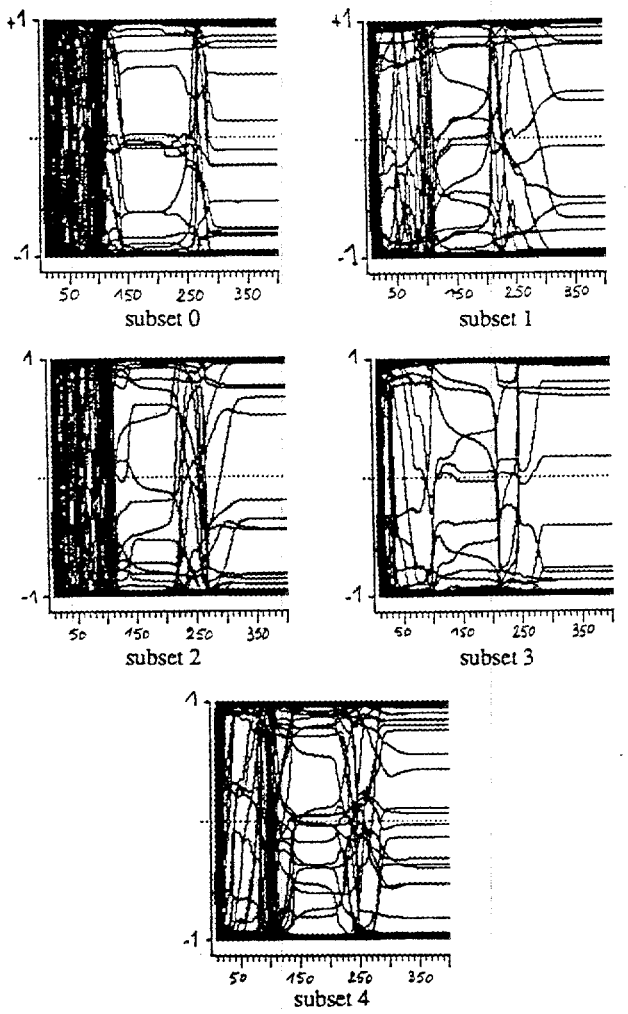


figure 18-b: At k fixed, curves giving V_i^k as a function of the scan number.

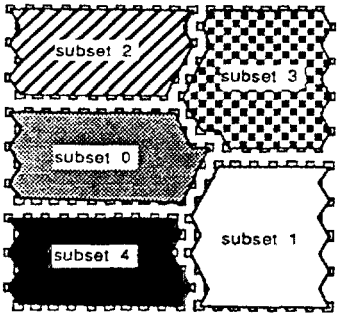


figure 18-c: Balanced 5-partitioning with α=1 of a regular hexagonal network of 324 vertices and 901 edges provided by the mean field approximation (edges are not visualized). The ambient temperature is 4. The interconnection cost is 86, the imbalance energy 259,2 and the total energy 86,56. The distribution of vertices is the following: 64 vertices in subset 0, 69 vertices in subset 1, 61 vertices in subset 2, 69 vertices in subset 3, 61 vertices in subset 4.

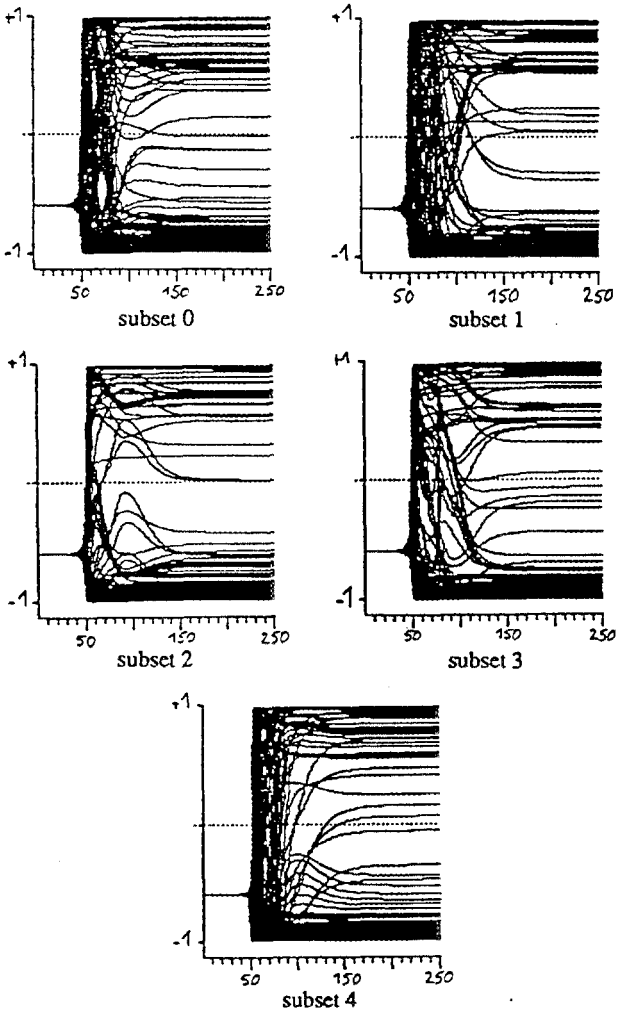


figure 18-d: At k fixed, curves giving V_i^k as a function of the scan number.

- non homogeneous graph and α=2: left monocular description of figure 2 (see figure 19-a and 21-b). The system needs less than N/2 scans to converge.

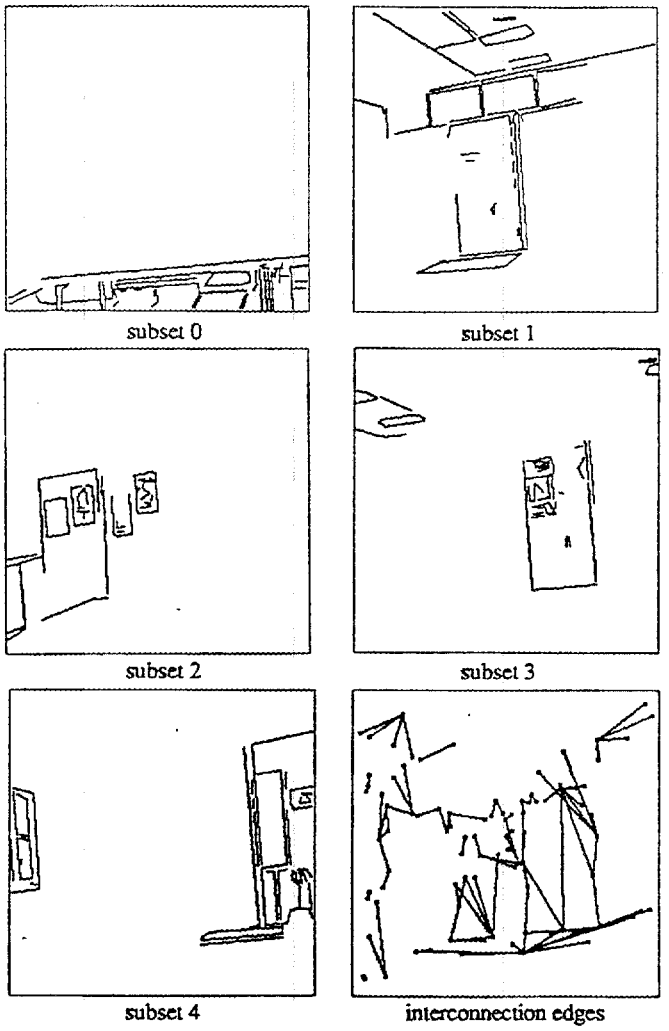


figure 19-a: Balanced 5-partitioning with α=2 of the left non homogeneous graph of the figure 2 provided by the mean field approximation algorithm. The ambient temperature is 3. The interconnection cost is 85, the imbalance energy 196,8 and the total energy 85,86. The distribution of vertices is the following: 63 vertices in subset 0, 66 vertices in subset 1, 63 vertices in subset 2, 61 vertices in subset 3 and 70 vertices in subset 4.

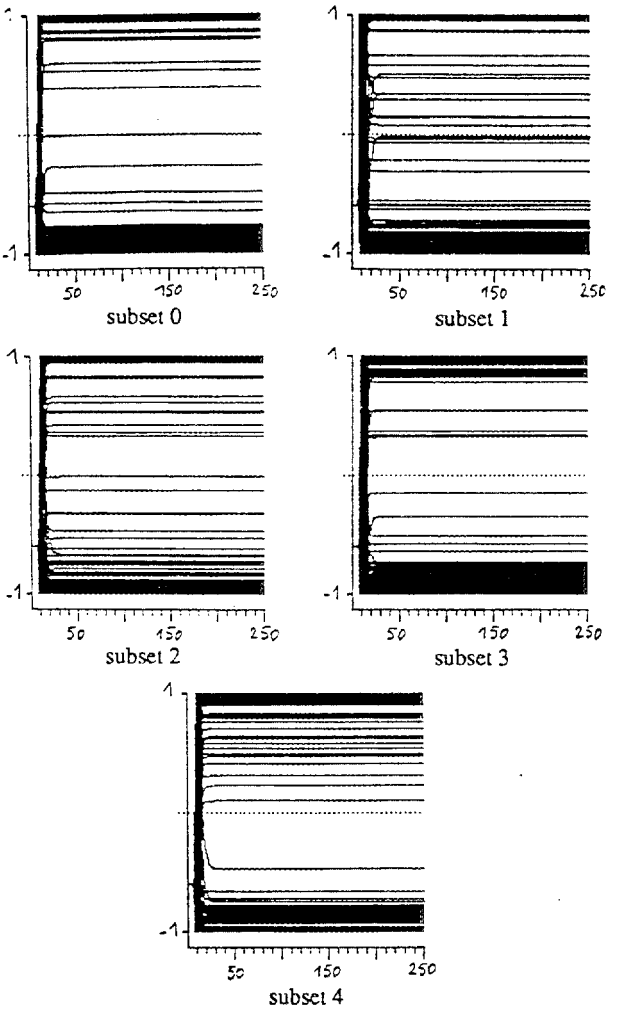


figure 19-b: At k fixed, curves giving V_i^k as a function of the scan number.

Those results are comparable to those given by the simulated annealing but are obtained in a CPU time 10 to 20 times smaller. The interconnection cost given by the generalized Kernighan method are about 20 % greater.

The difficulty of this method depends on the choice of the two parameters λ and T. λ is chosen without ambi - guity (equation 4.15). The choice of the temperature T has not a majour influence on the quality of the result when it is chosen in a certain range (between 1 and 4 for graphs having hundreds of vertices). Additionally, one notices that the range of possible temperature increases as the vertex numbers grows.

4.3.3 Mean field annealing

In the mean field approximation algorithm, the temperature is definitively fixed. Another possibility consists in doing an annealing during the convergence process. Consequently the convergence time is reduced: the smaller the temperature, the more rapid the convergence of the previous dynamic system (equation 4.51). Additionally, once the system has converged, the membership probabilities of a subset are more discriminant than previously obtained: all the V_i^k are forced to tend to +1 or -1 when the temperature decreases during the convergence process. The determination of the final partition is made without ambiguity concerning the vertex membership of a subset. The previous algorithm (see section 4.3.2) is slightly modified and is given in Appendix D.

Practically, the decreasing factor of the temperature (decT) between two scans must be slightly smaller than 1.

We give experimental results in figures 22 and 23:

- homogeneous graph , α=1, and decT = 0,995 : regular hexagonal network (see figure 20-a). Figure 20-b shows the evolutions in each subset (k fixed) of the V_i^k components as a function of the scan number. The system converges much more rapidly than previously (compared to figure 18-b).

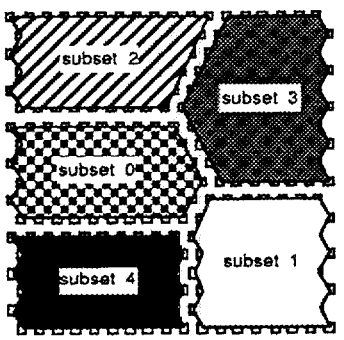


figure 20-a: Balanced 5-partitioning with α=1 of a regular hexagonal network of 324 vertices and 901 edges provided by the mean field annealing algorithm (edges are not visualized). The initial temperature is 5 and the decrease coefficient of the temperature is 0,995. The interconnection cost is 85, the imbalance energy 547,20 and the total energy 86,18. The distribution of vertices is the following: 63 vertices in subset 0, 63 vertices in subset 1, 63 vertices in subset 2, 75 vertices in subset 3 and 60 vertices in subset 4.

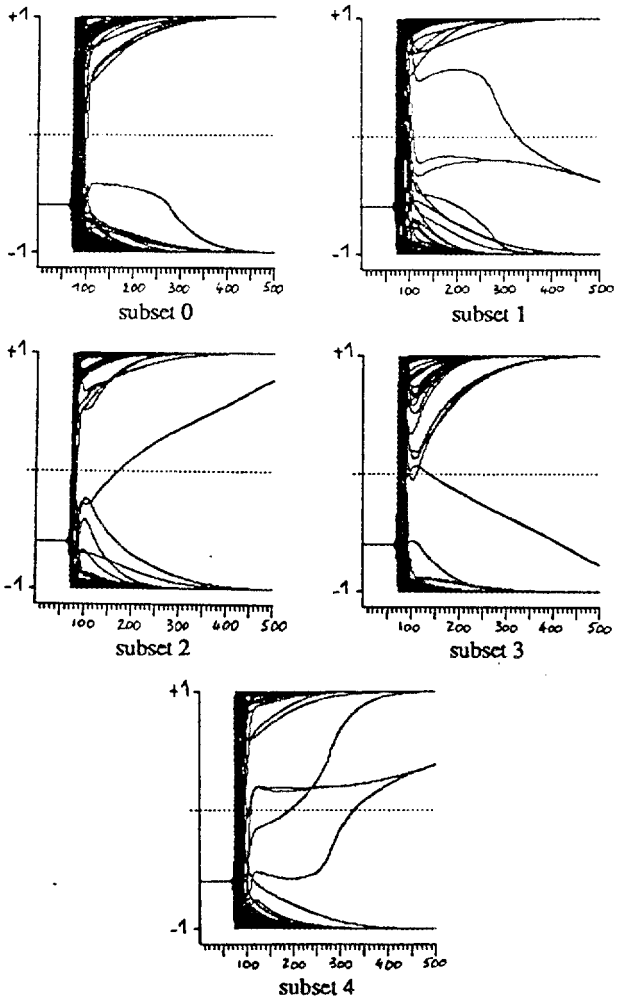


figure 20-b: At k fixed, curves giving V_i^k as a function of the scan number.

- non homogeneous graph, α=2, and decT = 0,995 : left monocular description of figure 2 (see figure 21-a and 21-b).

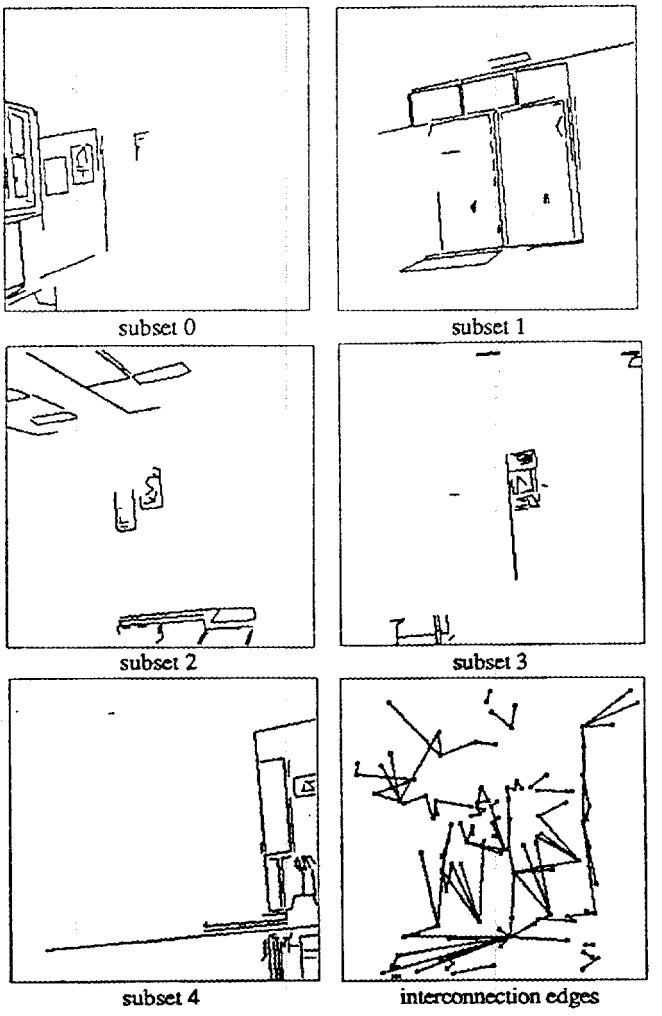


figure 21-a: Balanced 5-partitioning with α=2 of the left non homogeneous graph of the figure 2 provided by the mean field annealing algorithm. The initial temperature is 5 and the decrease coefficient of the temperature is 0,995. The interconnection cost is 107, the imbalance energy 532,8 and the total energy 109,33. The distribution of vertices is the following: 64 vertices in subset 0, 67 vertices in subset 1, 67 vertices in subset 2, 55 vertices in subset 3 and 70 vertices in subset 4.

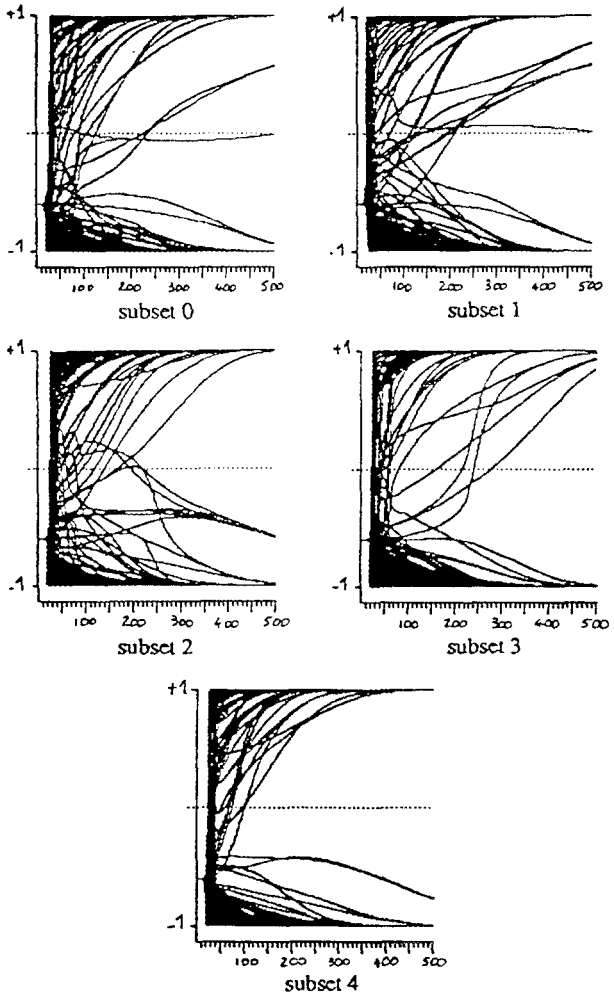


figure 21-b: At k fixed, curves giving V_i^k as a function of the scan number.

Those results are comparable to those obtained by using the mean field approximation but the convergence is more rapid and the discretisation which produces the final partition is less ambiguous.

5. Conclusion

We have shown how a NP-complete combinatorial optimization problem such as the graph K-partitioning can be treated as a minimization problem of a global quadratic energy thanks to the use of vectorial entities. We have proposed several neural methods to minimize this energy.

We have shown how to adapt the synaptic weights between the binary or analog neurons of an Hopfield network so that the system converges to energy minima which are good solutions of our problem.

We have extended the well known simulated annealing procedure (SA) to the use of our vectorial entities.

We have developed a deterministic and massively parallel method using the mean field theory (MFT) to handle our problem. This method, implemented on a conventionnal computer, gives very good results in a CPU time divided by an order of magnitude 10 to 20 compared to the simulated annealing.

Eventually, in the mean field annealing method (MFA), one makes an annealing during the convergence process of the MFT algorithm. This causes the system to converge more rapidly. Additionally, the final partition is determined with less ambiguity than with the mean field approximation.

Experimental results are given for the SA, MFT anf MFA methods.

Acknowledgements

The authors would like to thank Dr. R. Horaud and Dr. T. Skordas for enlightening discussions.

Appendix A

In this appendix, upper and lower bounds of the interconnection cost are given as a function of the desired subset number. Additionally, an approximation of the internal density of a subset is developped.

Upper and lower bounds of the interconnection cost

Let us consider a graph of N vertices and M edges with the density d = 2.M / (N.(N-1)). One wants to partition this graph in K subsets. Let us suppose that N(k) is the vertex number in the subset k and that d(k) is the internal density in this subset.
The number of vertices having its extremities in the subset k is d(k).N(k).(N(k)-1) / 2. Therefore, the number of interconnection edges is:

$$M - \sum_{k=1}^{K} \frac{d(k)}{2}.N(k).[N(k)-1] \tag{A.1}$$

Let us suppose that the partition is perfectly balanced ($N_1 = ... = N_K$) and that the density is the same for every subset. In the best case,the number of interconnection edges is 0. Therefore, it leads:

$$\forall\, k \in \langle 1,K\rangle,\ d(k) = K.d.\frac{N-1}{N-K}\ , \tag{A.2}$$

with the condition: for all k, d(k) is lower than 1. We notice that d(k) decreases when K increases.
Therefore, we obtain a limit value of K:

$$K_{limit} = E\left[\frac{N}{(N-1).d+1}\right]. \tag{A.3}$$

A lower bound of the interconnection cost is C_{min} defined by:

$$\forall\, K \le K_{limit},\ \ C_{min}(K) = 0\ , \tag{A.4}$$

$$\forall\, K > K_{limit},\ \ C_{min}(K) = \frac{N^2.d}{2}.\left[1 - \frac{1}{K}.\left(\frac{d-1}{N.d}.K + \frac{1}{d}\right)\right]. \tag{A.5}$$

In the worst case, for all k, d(k) equals to the graph density. Then an upper bound C_{sup} of the interconnection cost is obtained by replacing d(k) by d in the previous formula. It leads:

$$\forall\, K \le N,\ \ C_{sup}(K) = \frac{N^2.d}{2}.\left(1 - \frac{1}{K}\right). \tag{A.6}$$

Approximation of the density in a subset

We suppose that N is much greater than K. Let M(k) be the edge number in the subset k. If the partition is perfectly balanced, we have:

$$\forall\, k \in \langle 1,K\rangle,\ d(k) = \frac{2.M(k)}{\frac{N}{K}.\left(\frac{N}{K}-1\right)} \approx \frac{2.K^2.M(k)}{N^2}\ . \tag{A.7}$$

In first approximation, one can take, for all k, M(k) = M/K. Therefore:

$$\forall\, k \in \langle 1,K\rangle,\ d(k) \approx \frac{2.M}{N^2}.K \approx K.d\ . \tag{A.8}$$

Appendix B

In this appendix, we describe the simulated annealing algorithm. We use vectorial entities defined in section 4.1. The algorithm is the following:

1. Get an initial system configuration.

   Construct the associated $(\vec{V_i})_{i \in <1,N>}$.

2. Fix the initial ambient temperature T by using equation 4.31.
   Fix the length of the elementary transformation sequences so as to reach the equilibrium at any temperature T: L = 100.N.(K - 1).

3. Initialization of the number of accepted transformations at this temperature: NTaccept = 0.
   Repeat L times:
   3.1 Pick at random a vertex i of the graph (this vertex is in the subset k : $V_i^k = 1$).
   3.2 Pick at random a subset l which is different from k.
   3.3 Calculate the energy variation associated to the move of the vertex i from the subset k to the subset l by using equation 4.29.
   3.4 If the energy decreases:
   3.4.1 The elementary transformation is accepted: NTaccept -> NTaccept+1.
   3.4.2 Operate the transformation: $V_i^k = -1$ et $V_i^l = 1$.
   3.5 If the energy increases, then the elementary transformation is accepted with a probability given by equation 4.30.
   3.6 If NTaccept = L/10, then consider that the equilibrium is reached at T: stop (go to step 4.).

4. If NTaccept = L/10, update the ambient temperature ($T_{new} = 0{,}93\ T_{old}$) and go to step 3.
   If NTaccept is between N and L/10, le system is freezing, update the ambient temperature ($T_{new} = 0{,}965\ T_{old}$) and go to step 3.
   If NTaccept < N (the system is frozen), stop: the solution (final K-partition) is obtained.

Appendix C

In this appendix, we describe the mean field approximation algorithm. We use vectorial entities defined in section 4.1. The algorithm is the following:

1. Fix the running mode:
   - synchronous -> fct = 0,
   - asyncronous -> fct = 1.

   Fix the temperature T.
   Fix the scan number of the graph vertices: Nbscan.
   Get, for all i and k, an initial value V_i^k randomly choosen between the values
   $(2/K-1-10^{-5},\ 2/K-1+10^{-5})$.

2. Repeat Nbscan times:
   2.1 Randomly scan the graph vertices in such a way that every vertex is updated once.
   2.1.1 Update every vertex seen in the scan:
   2.1.1.1 Calculate, for all k, $V_i^{k\ new}$ (equation 4.50).
   2.1.1.2 If fct = 1 (asynchronous running mode), update for all k: $V_i^{k\ old} = V_i^{k\ new}$.
   2.2 If fct = 0 (synchronous running mode), update for all vertex i and for all subset k :
   $V_i^{k\ old} = V_i^{k\ new}$.

3. 3.1 Test if the system has converged into a configuration different from the initial one.
   3.2 If the system has not converged, either the temperature T is too high or Nbscan is too

small. Go to step 1.
3.3 If the system has converged, for all graph vertex i:
3.3.1 Determine k such that V_i^k is the greater.
3.3.2 Do $V_i^k = 1$ and, for all $l \neq k$, $V_i^l = -1$ -> the vertex i is in the subset k.

Appendix D

In this appendix, we describe the mean field annealing algorithm. We use vectorial entities defined in section 4.1. The algorithm is the following:

1. Fix the running mode:
   - synchronous -> fct = 0,
   - asyncronous -> fct = 1.

   Fix the initial temperature $T = T_0$.
   Fix the scan number of the graph vertices: Nbscan.
   Fix the decreasing coefficient of the temperature between two consecutive scans: decT.
   Get, for all i and k, an initial value V_i^k randomly choosen between the values
   $(2/K-1-10^{-5}, 2/K-1+10^{-5})$.

2. Repeat Nbscan times:
   2.1 Randomly scan the graph vertices in such a way that every vertex is updated once.
   2.1.1 Update every vertex seen in the scan:
   2.1.1.1 Calculate, for all k, $V_i^{k\ new}$ (equation 4.50).
   2.1.1.2 If fct = 1 (asynchronous running mode), update for all k: $V_i^{k\ old} = V_i^{k\ new}$.
   2.2 If fct = 0 (synchronous running mode), update for all vertex i and for all subset k :
   $V_i^{k\ old} = V_i^{k\ new}$.
   2.3 T -> decT * T.

3. 3.1 Test if the system has converged into a configuration different from the initial one.
   3.2 If the system has not converged, Nbscan is too small. Go to step 1.
   3.3 If the system has converged, for all graph vertex i:
   3.3.1 Determine k such that V_i^k is the greater.
   3.3.2 Do $V_i^k = 1$ and, for all $l \neq k$, $V_i^l = -1$ -> the vertex i is in the subset k.

References

(1) AYACHE, FAVERJON : " Efficient registration of stereo images by matching graph descriptions of edge segments" *International Journal of Computer Vision, 1(2), 1987*

(2) S. BOKHARI : "Assignement Problems in Parallel and Distributed Computing" *Kluwer Academic Publishers, 1987*

(3) M.BURSTEIN : "Algorithms for Partitionning of VLSI Networks" *IBM Technical Disclosure Bulletin, Vol.25, No.11A, April 1983*

(4) W.E.DONATH,A.J.HOFFMAN: "Algorithms for Partitionning of Graphs and Computer Logic Based on Eigenvectors of Connection Matrices" *IBM Technical Disclosure Bulletin, Vol.15, No.3, August 1972*

(5) M.R.GAREY, D.S. LOHNSON : "Computers and Intractability" *W.H.Freeman and Company, New York, 1979*

(6) S.GEMAN and D.GEMAN : "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images " *IEEE PAMI, 6(6):721-41,November 1984*

(7) J.J.HOPFIELD :"Neurons with graded response have collective computational properties like those of two-states neurons" *Proc. Natl. Acad. Sci. USA, Vol 81, May 1984, Biophysics*

(8) J.J.HOPFIELD :"Neural networks and physical systems with emergent collective computational abilities" *Proc. Natl. Acad. Sci. USA, Vol 79, April 1982, Biophysics*

(9) R.HORAUD and T.SKORDAS: "Stereo-correspondence through feature grouping and maximal cliques" *IEEE PAMI to appear*

(10) B.KERNIGHAN : " Some Graph Partitionning Problems Related to Program Segmentation" *Princeton University, Ph.D., 1969*

(11) B.KERNIGHAN, S.LIN :"An Efficient Heuristic Procedure for Partitionning Graphs" *The Bell System Technical Journal, February 1970*

(12) S.KIRKPATRICK, C.D. GELATT, M.P. VECCHI : "Optimization by Simulated Annealing" *IBM Thomas J.Watson Research Center, Yorktown Heights, NY, 1982.*

(13) E.L. LAWLER : "Electrical Assemblies with a Minimum Number of Interconnections" *IEEE Transactions Electronic Computers, Vol EC-11, February 1962*

(14) J.A. LUKES : "Combinatorial Solution to Partitionning Problems" *Technical Report No 32, Stanford Electronics Laboratories, May 1972*

(15) J.A. LUKES : "Combinatorial Solution to the Partitionning of General Graphs" *IBM J. Res.&.Dev. (USA) Vol 19, No.2, March 1975, pp 170-80*

(16) "Procedure for Partitionning the Nodes of a Graph" *IBM Technical Disclosure Bulletin, Vol 28, No.9, February 1986, pp 4030-4034*

(17) C.PETERSON, J.ANDERSON : "Neural Networks and NP-complete Optimization Problems : A Performance Study on the Graph Bisection Problem" *Complex Systems 2 (1988)*

(18) G.RAO, H.STONE, T.HU : " Assignement of Tasks in a Processor System with Limited Memory" *IEEE actions on Computers, Vol C-28, no 4, April 1979*

(19) H.STONE : "Multiprocessor Scheduling with the Aid of Network Flow Algorithms" *IEEE Computer, vol.16, no 1, January 1983*

(20) G.A.TAGLIARINI, E.W.PAGE : "A Neural-Network Solution to the Concentrator Assignment Problem" *IEEE Conf. on "Neural Information Processing Systems - Natural and Synthetic", Denver, Colorado, November 8-12, 1987*

(21) D.W TANK, J.J.HOPFIELD :"Simple Neural Optimization Networks : An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit" *IEEE Transactions on Circuits & Systems, Vol.CAS-33, No 5, May 1986*

COMPUTER-AIDED TELEOPERATION (CAT)

CLASSICAL TELEOPERATION

Fixed relationship between operative/sensory units of the workspace and control/restitution devices in the master station

CAT

"Programmable coupling" of the master and slave devices, allowing the operator to dynamically select the system configuration according to the task

Teleassistance concept

Analog control

Implements the coupling algorithms

Symbolic control

Management of the system configurations

Control modes

Geometric scaling

facilitates the performing of large or fine movements on the slave arm

Position indexing

allows the operator to keep his hand in a restricted and comfortable zone irrespective of the slave arm position

Force scaling

brings down physical stress or increases force feedback feeling

Force indexing

weight balancing

DOF locking

helpfull for tool operation like drilling

Sensor referenced modes

collision avoidance, surface following, target tracking, assisted grasping (reflex or hybrid scheme)

Model referenced modes

generation or attractive or repulsive force feedback relatively to immaterial potential zones

I. Symbolic control (Supervision)

multi-mode and multi-device features of CAT → increased operator mental workload

The availability of numerous control and restitution modes requires:

- the determination of the appropriate machine behaviours, according to the task;

- the formulation of a strategy in terms of functions which can be executed by the real time control system.

The problem has therefore a decisional aspect (what is the optimal behaviour of the system with respect to the current sub-task?) and a communication aspect (how to implement it?).

Some symbolic assistance is needed.

Problems associated with supervision

1. Programming strategy

Remote tasks are generally:
- non repetitive;
- poorly defined;
- performed under conditions of weakened perception.

Task execution must then rely, at least partly, on the on-line decision making capabilities of the man-machine system.

→ off-line preparation focuses on a generic model of the considered task in order:

1. to make profitable the programming effort;

2. to take into account the on-line processing of:
   - possible events;
   - recovery procedures;
   - alternative strategies;
   - parameter setting.

2. Multi-level Dialogue

On-line, man-machine cooperation is supported by a dialogue which can be more or less verbose and tedious according to the task analysis capabilities of the supervision computer. As these capabilities change with the considered task and the encountered incidents, the system has to provide some facilities for a multi-level dialogue.

In practice, the relevant dialogue levels are:

the object level
where the task is described as a sequence of actions directed towards the environment objects;

the effector level
where the task is described in terms of the end-effector motions;

the system level
which deals with the successive states of the CAT system.

3. Task modeling

Symbolic man-machine cooperation is closely linked to the respective knowledge the operator and the CAT system possess about the task.

We must therefore consider the following models which characterize the supervision process:

- the CAT machine model of the task;
- the operator model of the task
  - at the analog level (when the human manually pilots the manipulator using the master arm);
  - at the symbolic level (when the human acts as a supervisor);
- The cooperation model.

Furthermore, if a multi-level dialogue is implemented, these models are probably hierarchically structured.

II. The SARAH II test-bed

In order to study the symbolic control aspects of CAT, a complete system is being developed by UGRA.

Its main features are:

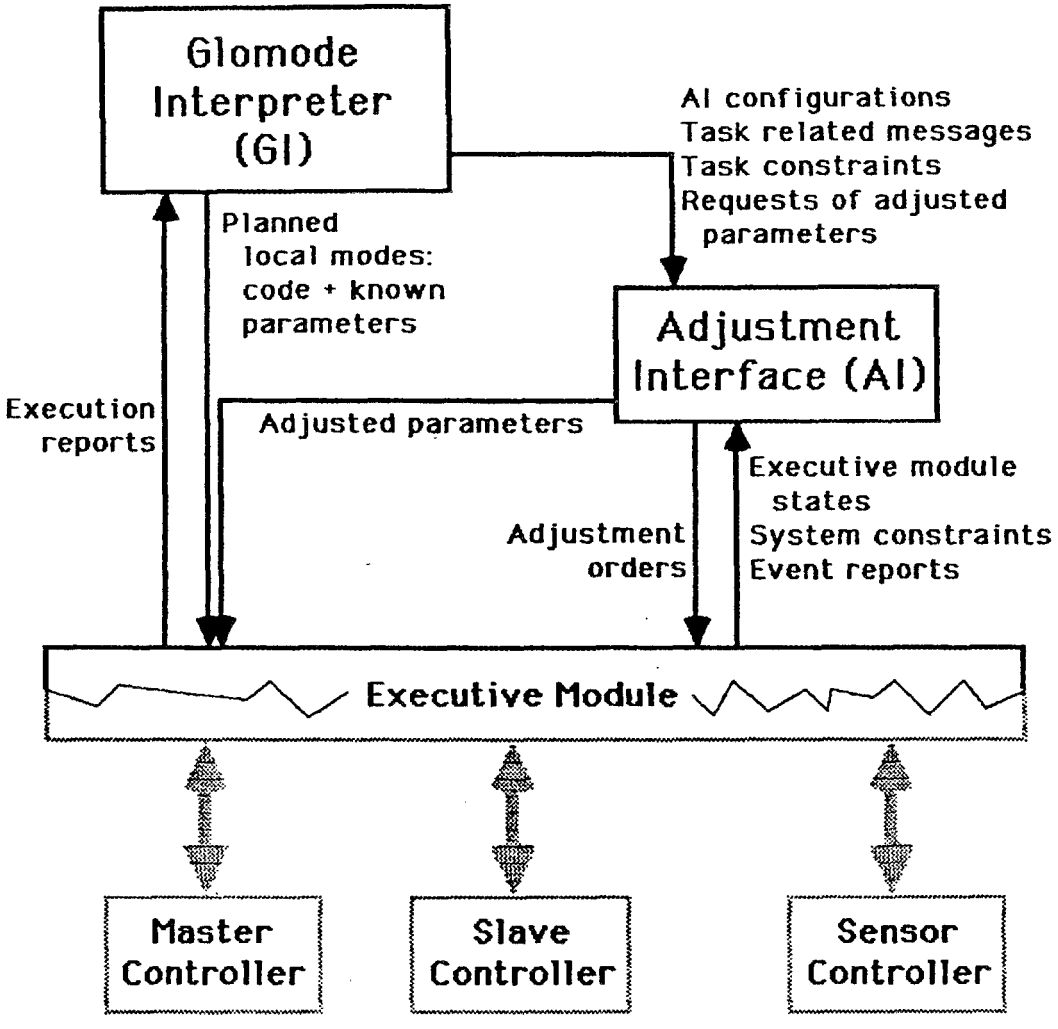
- an extended set of manual, automatic and mixed control modes

- a symbolic control level composed of 3 modules:

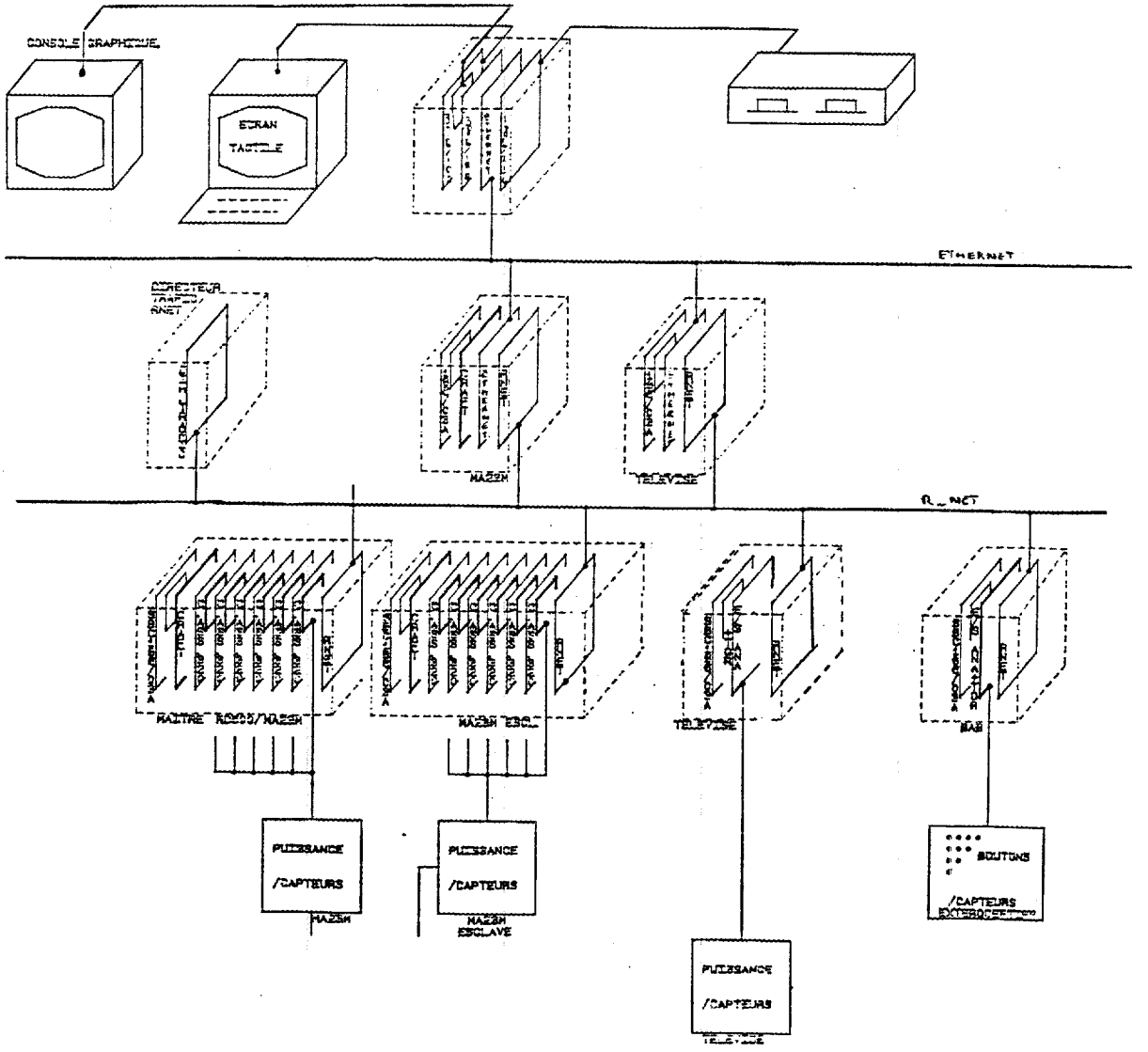
  * the Execution Module integrating a teleoperation language which can be used off-line for task programming (effector level), as well as on-line for operator direct inputs

  * the Glomode Interpreter (GI) which implements a task oriented object level man-machine dialogue

  * the Adjustment Interface (AI) which supports the on-line system level operator interventions



Functional structure of the SARAH II system



Computer architecture of the SARAH II system

III. The Executive Module

The Executive Module supports the SPARTE Teleoperation language and implements the basic control loops corresponding to the specified CAT behaviours. It communicates with the Master and Slave Arm Controllers through the R-Net high rate transmission link.

SPARTE main features:

- off-line task programming which specify the control modes required by manual, automatic or mixed execution

- on-line inputs of individual instructions (called direct effect instructions or DEI) which allow the operator to modify any programmed CAT behaviour

- definition of hybrid control modes (manual/ position/force)

- management of external events

- learning facilities to help the programmer specifying relevant positions in the workspace

SPARTE describes a CAT behaviour as the combination of:

- a "fundamental instruction" specifying the control law applied to each of the slave DOFs (ex: force control for X, other DOFs manual)

- some "harmonic parameters" which values can be set or modified (possibly on-line) by adjustment or activation/deactivation instructions (ex: force value, weight suppression state, ...)

- some configuration parameters (as the control cordinate frame)

Fundamental behaviours and configuration parameters generally depend of the task and can thus be off-line programmed. On the other side, harmonic parameters may vary according to the operator or the environment and must be at least adjusted on-line.

This description is completed with instructions for event management which may:

- activate/deactivate the watch of a particular event (operator interruption, mechanical stop, ...)

- define how the system must react to the occurrence of a watched event (send a message, stop the current fundamental mode and interpret the next instruction, ...)

IV. The Glomode Interpreter (GI)

At the Glomode Interpreter level, the tasks are modeled using off-line compiled structures called glomodes.

A glomode is a generic description of a remote sub-task stated in a form which is:

redundant

in the sens that it provides a number of alternative strategies and recovery procedures applicable to incident situations;

incomplete

as certain parameters are not known and the actual sequence of actions is determined on-line.

modular

a given glomode may be interpreted on-line or used off-line in order to define a more complicated glomode.

Glomode Interpreter functions

The Glomode Interpreter implements a task-level dialogue with the operator and thus interactively processes the glomode sub-task descriptions in order to generate:

- sequences of relevant CAT behaviours according to the task:
  * at the executive level, each of these behaviours is described by a SPARTE procedure called through a GI order
  * such an order specifies a list of parameters, some of which can be adjusted by the operator and are therefore requested to the adjustment Interface

- Adjustment Interface configurations defining the contents and the layout of the AI interactive screen

- messages related to the task progession

- task constraints applied to the adjustment process and intended to avoid conflicts between the off-line programmed behaviours and the on-line operator inputs

V. The Adjustment Interface (AI)

The Adjustment Interface allows the operator to master the part of symbolic control which cannot be deduced from an off-line (and often summary) analysis of the task.

Functions of the Adjustment Interface:

1. Inform the operator about:
   - the current behaviour of the system
   - the occurrence of external events and the system reactions
   - messages related to the task

2. provide the operator with some facilities:
   - to complete or modify the CAT behaviours specified off-line
   - to request system data as the end effector position or the gripper state
   - to stop the current behaviour or to signal an unexpected event
   - to re-initialize the Executive Module

3. use of the system without Glomode Interpreter (in the case of very simple or completely unpredictable tasks)

Adjustment principles

* Numerical and boolean parameters of the Executive Module can be modified using Direct Effect Instructions.

* The human acts on task related parameters which are not always defined at the executive level.

* The adjustable parameters of the Executive Module (ex: controlled forces/torques) are associated with one or several Adjustable Objects (AO; ex: drilling or grinding forces) which values can be modified by the operator.

* The AOs are managed by the Adjustment Interface according to the task constraints fixed by the Glomode Interpreter.
  For exemple, the operator can be allowed or not to change the value of an AO, and in the former case, the AO can react or not by sending a DEI to the Executive Module.

* As a DEI can be forbidden or have no visible effect over a given CAT behaviour, the adjustment process takes into account some system contraints.

Interfaces

The Adjustment Interface is therefore connected to:

1. the operator who perceives the displayed data and can modify the value of the activated AOs

2. the Executive Module which:
   - processes the DEIs sent by the AOs
   - informs the Adjustment Interface about:
     * the current behaviour of the executive level
     * the occurrence of external events
     * the system constraints applied to parameter adjustment

3. the Glomode Interpreter which sends to the Adjustment Interface:
   - configuration orders (AO creation, screen selection, ...)
   - messages related to the task progression
   - task constraints applied to the AOs
   - request of adjusted parameters to complete its own orders to the Executive Module

Conclusion

SARAH II is an open system which can be used to experiment different man-machine cooperation strategies.

We particularly intend to study the following points:

- what is the best trade-off between off-line programming and on-line adjustment?
- what knowledge must we put into the system in order to enhance its comprehension of the external world, including the human operator?
- how can we represent that knowledge?
- how does the human behave when faced with a symbolic control task in CAT?
- how can we design an efficient multi-level man-machine dialogue?
- what are the guidelines for the design of a good man-machine cooperation strategy?

APPENDIX A

Preliminary Report on the Base-Case Scenario Definition in Preparation of the 1989 CESAR/CEA Workshop on Autonomous Mobile Robot

Antoine Cossic
August 1988

1. INTRODUCTION

This report was prepared by Antoine Cossic of CEA following a one-month term at CESAR during the Summer of 1988. The objectives of Mr. Cossic's visit at CESAR were to (1) initiate with the CESAR team the definition of the base-case scenario which would serve as the basis for testing and coupling mobile robot navigation algorithms during the forthcoming 1989 CESAR/CEA Workshop on Autonomous Mobile Robot and, (2) obtain all geometric information and specifications relating to HERMIES-IIB and its CESAR laboratory surroundings so that the robot and its demonstration environment could be accurately simulated using the CEA's ARES system.

2. BACKGROUND

2.1 ARES ACTIVITIES

ARES group (Atelier de Robotique Et Simulation – Robotics and Simulation Environment Tool) was created in March 1986. The original idea was to provide a large software environment for processing and testing third generation robot algorithms (control, collision avoidance, navigation, sensor data analysis) and a help for end-users in computing complex robotic tasks (description of the world and the robot(s), definition of tasks using a high-level programming language). The means consist of a simulation bed of virtual robots, or systems of cooperating ones, in a virtual environment using workstations, for now IBM 6150 RT PC operated by AIX (UNIX System V) and IBM 9370-60 computer running VM, coupled with an IBM 5080 graphic workstation, those systems providing powerful 3D graphics facilities, multi-tasking facilities, fast and efficient floating point operations processors.

ARES intends to provide world models that are suitable for 3D graphic animation and an increasable set of robotic algorithms. It also intends to provide an easy access for any program via the "ARES library" (a set of fully documented functions including geometric and updating operators) and to give the capability for any user to generate any arbitrary complex, solid objects by bonding and jointing primitives, so that a 3D environment can be completely described.

The world model, which is performed, is based on a constructive solid geometry (CSG) tree where leaves are fitted with boolean operators (union, intersection, difference) leading to rigid solids. Those solids may have their own attributes, not only geometric ones. What we call jointed solids in the arborescent description of an object are rigid or jointed solids fitted together with mechanical joints. An object is a solid, either a rigid or jointed one, which values have been affected to geometric parameters. The universe is no more than a collection of such objects which have been given a location, a velocity too, in the world. Objects may be bonded together within the world by "natural" bonds such as gravity or contact (lying object, hanging on one) or the result of a robot's action, grasping for instance. Rigid solids are described by a wire-frame model, allowing fast 3D computation or a polyhedral one, external surfaces arising from CSG operations can be computed and thus a lot of properties such as volume, center of mass, matrices of inertia, inside/outside location, too. This is also quite suitable for modifications of the universe, its graph, sensor data analysis, and ray-tracing methodology.

The basic algorithmic components, which are already developed or under development, are mainly (1) a geometric/kinematic/dynamic control command of manipulator units with any degrees-of-freedom, (2) a sensor data analysis scheme for sensors, laser range-finders, cameras based on a map rebuilding algorithm, accounting for feedback information analysis and updating, (3) navigation algorithms, local and global strategies, and (4) a kinematic model of the motion of a mobile vehicle on an arbitrary surface.

According to the required robotic tasks, several processed may be performed at the same time, virtually, and/or in a specific way; activation of one process may depend on information left by others. That is why a task scheduler has been

intended to be developed for monitoring the execution of concurrent processes, making them communicate with each other, synchronizing them, and managing mutual interactions (conflicts). UNIX S5 V3 multi-task operating system has been chosen to develop this scheduler (in C) though some real-time facilities are lacking.

ARES team is looking for portability and focusing on standards. Portability involves not only the programming language but also the operating system which operates. Most of ARES routines are written in FORTRAN 77, a small part is coded in C. ADA language seems a good middle term candidate, strong typing and abstraction (object oriented), recursivity, and reliability for simulation developments. UNIX operating system is an issue for it is supported by most of machine manufacturers, international users, and manufacturers working groups, too. As standards, ARES' choice has been a 3D graphic software package, PHIGS, which is very efficient and satisfactory for its capability to manipulate 3D objects, XAWINDOWS as a multi-screen/multi-task/multi-processors man-machine interface, a robotic language, LMAC, developed by the University of Besancon, France, which sources are available and which is well accepted by the whole robotic French community, and UNIX, once again, for those reasons detailed above.

2.2 CESAR ACTIVITIES

CESAR (Center for Engineering Systems Advanced Research) is a center of excellence in the study of intelligent machines. It is involved in the RISP (Robotics and Intelligent Systems Program) activities, since the middle of 1984.

The center's series of mobile robot research vehicles, called HERMIES (Hostile Environment Robotic Machine Intelligent Experiment Series), are self-powered systems consisting of a wheel-driven chassis, dual manipulators, and a directionally controlled sensor platform. HERMIES-IIB has been operational since June 1987. An on-board VME (Versa Module European) rack provides the link to HERMIES-IIB hardware for controlling the robot's effectors and sensors. This system is loaded with Motorola 68020 series microprocessors. Another on-board computer system is loaded with three four-node NCUBE hypercube parallel computers on which the image processing programs are executed and where resides the CLIPS expert system shell. The host computer is an industrial version of an AT PC. Communication between the VME computer system and the host computer is by an 8 megabauds parallel link.

HERMIES-IIB is a research tool. Flexibility has been incorporated into its design so that features can be added or modified as research requires, additional sensors can be mounted and the computer architecture can be upgraded if needed.

The first research topic concerns the capability of an autonomous mobile robot to continually monitor its, never fully predictable, environment to manage unexpected occurrences. The world's dynamic requires that the robot periodically adjusts its plans, in real time. That is why HERMIES-IIB is equipped with powerful parallel NCUBE processors. Besides this research goal, an important research activity is the optimization of overall computer architecture.

The CESAR team developed and implemented a demonstration to focus on its research, to verify the accuracy of its approach, and to better identify areas for further experimentations. Typically, the mobile robot is setting in an unknown

environment, knowing only its initial location A and the location of an intermediate goal B. It must navigate from A to B, avoiding small obstacles that may be on its path. From location B, HERMIES-IIB must find a control panel, move up to it and read analog meters. It is assumed that there are no more obstacles between subgoal B and the panel. Navigation strategy is performed by making wide-angle sonar scans and collision-free path planning. One sensor is dedicated to scanning the area ahead of the robot, allowing it to stop within two feet of an unexpected obstacle. An expert system rule base makes the robot take appropriate actions. All routines are written in C or CLIPS language.

Further research goals intend to integrate, that is called multi-sensor fusion, several techniques for navigation. Two methodologies for instance, one based on sonar sensor analysis, another one using vision. Another area of ongoing research concerns learning environmental properties: the aim is to make the robot able to discover a system's dynamic, the mocked-up control panel, by manipulating its components and observing the changes that result on panel meters.

With the HERMIES-III series, facilities for more realistic experimentation will be provided. Features include two CESAR manipulators, CESARm, 7-degrees-of-freedom lightweight units, which are suitable for dynamics of robotic dexterous manipulation, and a laser range-finder. CESAR research manipulator is another research goal at ORNL. It intends to provide an ideal robotics testbed.

3. OBJECTIVES

The objectives of these studies are testing the various algorithms that may be implemented, designing benchmarks to stress those algorithms according to precision criteria, and defining themes for the first workshop between CEA and ORNL which will occur at the end of the year or the beginning of 1989.

3.1 ALGORITHMS

One of the objectives concerns testing algorithms in order to compare them with each other. On one hand, there will be algorithms simulated on ARES' graphic workstation, and on the other hand algorithms that are implemented on HERMIES-IIB system and computed by it. The ARES simulation will also enable the programmer to test several algorithms and to analyze their results.

Mainly, two kinds of algorithms may be invoked. First, navigation routines for an autonomous mobile robot in unknown and probably hazardous terrain. ARES intends to simulate the 2D path planning methodology which runs on HERMIES-IIB's on-board computer system. But not only that, such a simulation will be enhanced by processing other various schemes; for instance, ARES may simulate 3D collision-free path planning, area which has been previously investigated by M. Goldstein among others. Potential field strategies might be introduced, too.

The second type of algorithms concerns sensor data analysis and/or vision processing. All those techniques aim to enable the robot to build a reliable internal spatial representation of the world that is a navigation map. ARES routines, of course, will not be able to process the variety of systematic errors which, for instance, Polaroid transducers, as those which are mounted on top of HERMIES-IIB, give rise to, but in some way, will approximate those physical uncertainties. Whatever, ARES will process either sonar or laser range-finder or camera geometric data analysis. Map building may include those techniques together, for instance, sensor data analysis (which are more suited) for local planning method and laser range-finder data analysis used for global planning strategy.

3.2 BENCHMARKS

What we call benchmarks are testcases and experiments that have been designed to test the aspects of robotic algorithms. This approach may include motion precision, repeatability, consistency, uniqueness, completeness, and so on. Focusing on HERMIES-IIB's demonstration, a typical benchmark may be an experiment defining location of obstacles, providing unexpected occurrences, an obstacle which has not previously been seen because it was hidden or it was suddenly appearing in the world and moving towards the robot or crossing its path, positioning the mocked-up control panel in such a manner that the robot must perform exploration and pattern recognition, in order to stress the various algorithms and verify their accuracy. On ARES' side, the same experiment will be simulated and probably enhanced by running other navigation routines and/or other map building techniques based on ARES world modeling. Comparison of results will be helpful to show the differences between the very real behavior of HERMIES-IIB and its simulated one. The improvement a powerful 3D graphic simulation can offer will reside in the capability of providing several simulation ways.

3.3 SPECIFICATIONS

These specifications, essentially, deal with geometric information about HERMIES-IIB's environment and configuration. Sensor performances will also be needed.

In order to describe HERMIES-IIB's world and to put it in a realistic manner, measurements and dimensions are to be provided: geometric features of the room, dimensions of obstacles and control panel, characteristics of HERMIES-IIB itself.

The vision system used for navigation and pattern-recognition is also to be described: its geometric potentialities, range resolution and physical features.

3.4 WORKSHOP THEMES

Themes that are proposed for the first workshop between ORNL and CEA on robotics and intelligent systems will be divided in four classes:

* the simulation of CESAR's navigation algorithms and the comparison with other collision-free path plannings. This study will mainly be a qualitative one;
* the simulation of HERMIES-IIB experiment, a benchmark has to be defined, without learning aspects that means no Artificial Intelligence;
* the area of geometric modeling, map building and self-location, 2D or 3D maps;
* the utilization of advanced vision systems: a 3D camera or a laser range-finder.

3.5 ACTIVITIES COVERED IN THIS REPORT

The main activities which are covered in this report essentially deal with information processing, navigation (for an autonomous mobile robot), map building, and uncertainty.

4. SPECIFICATIONS

This section describes the features of the HERMIES-IIB research mobile robot, its geometry, its environment, its computer architecture, and the vision system used for navigation. Software environment considerations are also dealt with.

4.1 ENVIRONMENT

The environment of HERMIES-IIB consists of a room which is approximately 7.8 × 6.6 meters by 2.10m tall. Obstacles are parallelepipedic solids with dimensions .60m × .60m × 1.20 tall or .30m × .30m × 1.20m. Control panel at which the robot is supposed to be close enough to read the meters is a metal box .61m × .61m wide by 1.m high containing two 10. × 10.cm analog meters, a row of four 13.cm square push buttons and two horizontal solid levers.

4.2 HERMIES-IIB ITSELF

HERMIES-IIB is a self-powered robot system consisting of a wheel-driven chassis, dual manipulators, on-board distributed concurrent processors, and a control sensor platform at its head. Each manipulator is a five-degree-of-freedom unit. The torso assembly for the arms includes a shoulder pitch motion for each arm.

The sonar system consists of 25 individual Polaroid range-finders. Maximum range is 7.75 meters and range resolution is equal to 2.5 centimeters, 24 of these sonar transducers are mounted in six 2 × 2 matrix clusters. The effective sonar beam from approximately 20 degrees for each transducer is reduced to 10 degrees for each phased-array cluster. Five of these clusters are mounted in a ring at the head of the robot. The sixth remaining cluster is mounted on a tiltable platform attached to the head.

A CCD video camera module is also mounted on this platform. It transmits images into a form usable to the robot. A vision application of this stereo camera system is for controlling HERMIES' arms, in front of the control panel, to take appropriate actions on the buttons and the levers.

4.3 COMPUTER ARCHITECTURE

The architecture of HERMIES-IIB consists of two computer systems, an IBM 7532 (industrial version of the IBM AT PC) loaded with NCUBE hypercube nodes, and a VME computer rack.

Four of the eight slots available on the AT computer's back plane are used for NCUBE parallel processing boards, at present, two NCUBE boards, each containing four processor nodes. The host computer is loaded with an INTEL 80286 CPU. Expansion to 16 NCUBE nodes can be realized by utilizing the remaining two slots.

The VME rack provides the link to HERMIES-IIB hardware for monitoring the robot's effectors and sensors. The transfer rate between the VME system and the host computer is approximately of 1 Mbytes per second.

4.4 COMPUTER PROGRAMS ON HERMIES-IIB

The host IBM computer may either run MS-DOS or AXIS operating system, UNIX like, which operates NCUBE hypercube computers. Computer programs are written in C and divided in four classes:

* the HERMIES routines (commands of motion, path planning);
* the CLIPS expert system shell (rule base of high-level decisions);
* the image analysis routines (including vision and sensor data);
* integration programs that reside on the NCUBE/AT host computer.

The image processing programs are executed on the NCUBE nodes.

4.5 ARES SOFTWARE ENVIRONMENT

ARES group's computer architecture consists of an IBM 9370 (series 60) host computer connected to a 3D graphic workstation, IBM 5080, via a channel through an IBM 5085 graphic processor. The host computer is operated by VM/SPoperating system.

Three devices are available within the graphic workstation:

* a mouse allowing stroking, picking and locating functions;
* a choice device, with Light Program Function Keypad, of 32 keys;
* a valuator with eight programmable turn-buttons allowing for instance extension, rotation or zoom functions.

The 3D images are computed by the graPHIGS software routines which can be called either in C, FORTRAN, Pascal or PL/1 routines. Most of the programs that are being developed, so far, by ARES are written in FORTRAN 77.

ARES computer programs run also on another host computer which is an IBM 6150 RT Personal Computer operated by the AIX, UNIX System V, operating system. Thus, a small part of the routines are written in C language. The 5080 workstation may be opened by either IBM mini-computer or the 6150 RT PC. GraPHIGS package is installed on both machines.

The simulation of HERMIES-IIB demonstration might include simulation of the hardware communication between the VME rack and the host NCUBE computer, using message queues for instance. That supposes that more than one process will run within the simulator. The IBM 9370, because of VM/CMS is not suitable for concurrent processing and exchanging data between routines. Yet, it is quite possible on the 6150 RT PC operated by UNIX System V. For UNIX offers low-level tools (SuperVisor Calls) which allow C-programmers to use system routines close to the kernel. Five interprocess communication techniques are available: (1) pipes, (2) signals, (3) message queues, (4) shared memory, and (5) semaphores. Only (1), (3), and (5) deal with communication of amounts of various data.

5. BENCHMARK (TESTCASE)

5.1 COMPARISON CRITERIA

Data and results will have to be compared to show the differences which may occur between the HERMIES-IIB demonstration and its simulation on a 3D graphic workstation.

These comparison criteria may be:

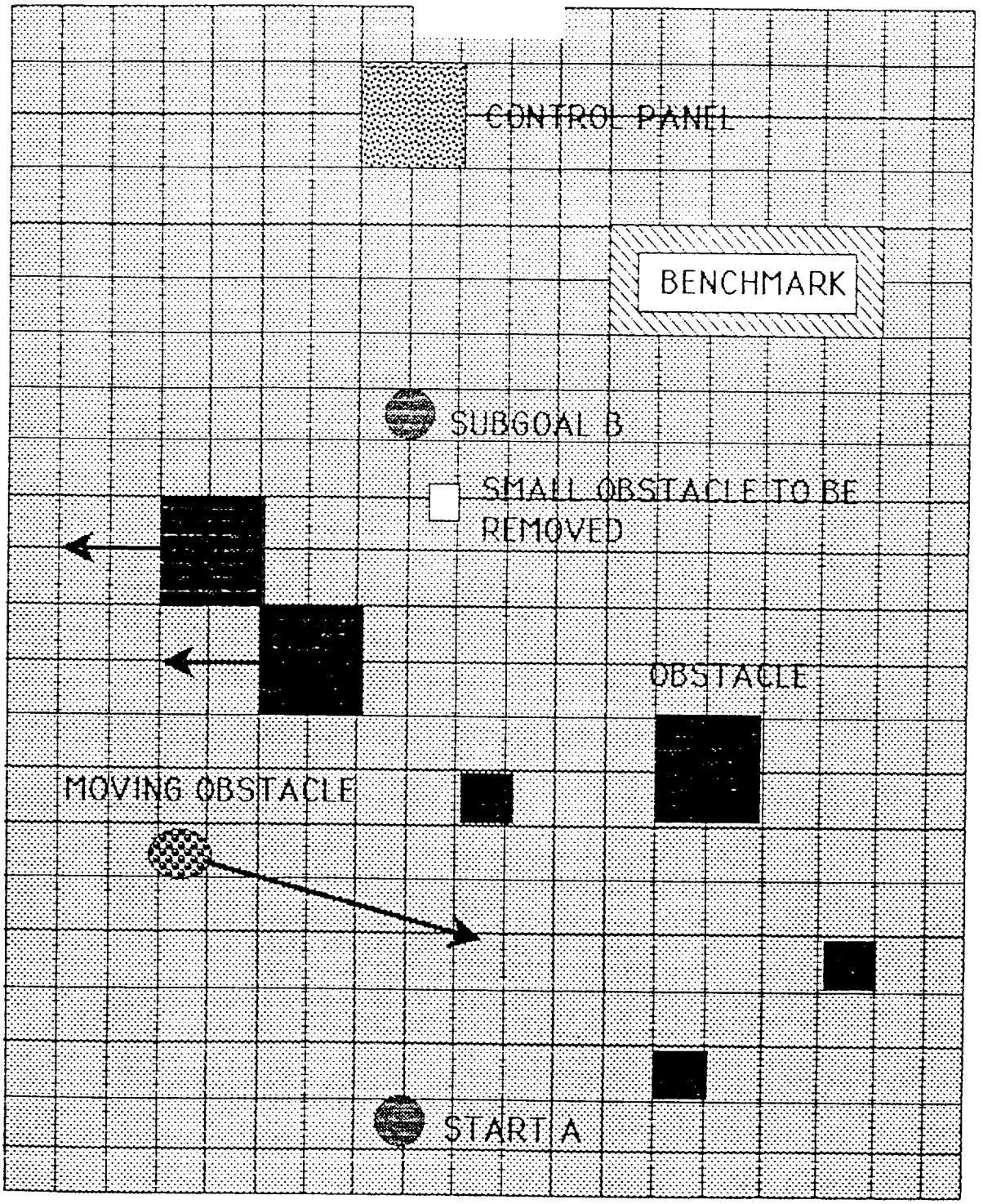
* First precision ones: precision of input data, on one hand, and on another hand, accuracy and uncertainty of results, output data from vision system analysis, exact location of the mobile robot.
* Consistency and usability of input/output data: are the results coherent, qualitatively speaking, or, to put it in another way, are they those expected, what is the likelihood of the robot's behavior. One can wonder also if all amounts of data are needed for processing, that means which data are useful at a given time.
* Completeness will also be an area of investigation: does the navigation algorithm work for many different configurations, in which case(s) is it not successful, what can be the reasons of wavering or not be successful to find a path.
* One has to look for uniqueness and repeatability, too. Is the path chosen the only one or the optimal one. Will the mobile robot be able to perform the same path under meaningless changes.
* Runtime considerations will be invoked. Efficiency and expensiveness will be taken care of. A response assurance will be needed. For instance, excessive response times will be prohibited.

5.2 TESTCASE

A typical benchmark is shown on the following sheet. The dimensions of the room are respected, each square is 0.30m × 0.30m wide. This is a 2D representation of HERMIES-IIB environment. Three large obstacles 0.60m × 0.60m) and three small ones (0.30m × 0.30m) are set in the room, but the robot does not know, *a priori*, their location. One moving obstacle, figuring a human being envelope, is moving ahead HERMIES-IIB. Another small obstacle, a cube which edge is 0.20m, is located close to subgoal B and will have to be removed by HERMIES-IIB. The obstacles are positioned in such manner that HERMIES-IIB, which width is assumed to be 4 feet (1.20m), will not be able to move up to point B. To reach this point, after HERMIES-IIB has sought enough, two of the large obstacles will be moved to another location, horizontal extension of two feet (0.60m) on the left, so that it will have enough place to perform its path between the obstacles.

Such an experiment will test:

* the ability of HERMIES-IIB to avoid a moving obstacle, crossing its path;
* the utilization of its two arms to remove a small fixed obstacle which has been detected ahead;
* the collision-free path-planning strategy (sonars will be used) in a complex, dynamic and unknown environment.


CONTROL PANEL
BENCHMARK
SUBGOAL B
SMALL OBSTACLE TO BE REMOVED
OBSTACLE
MOVING OBSTACLE
START A

6. WORKSHOP THEMES

This section deals with several topics that are proposed for the first workshop between the CEA/DEDR and ORNL/CESAR, which will be held probably in Spring 1989.

6.1 NAVIGATION ALGORITHMS

This theme will involve the simulation of CESAR's navigation algorithms and other collision-free path plannings, 2D and 3D ones. The environment is assumed to be unknown and non-static:

* the mobile robot only knows its start location and the location of the goal it has to move to;
* it must, by it own sensor data analysis system, find out its path between the obstacles, some of them may move or be removed.

The comparison between several ways of computing navigation will be the main topic. The aim is to outline their performances and determine in which conditions such or such algorithm is more suited according to precision criteria (precision, time, accuracy) discussed above.

6.2 HERMIES-IIB DEMONSTRATION

This topic will focus on the HERMIES-IIB demonstration. Particularly, it will deal with the so designed benchmark which is presented in this paper. On one hand, the real demonstration will show the capabilities to move in a complex terrain and the difficulties which arise. On another hand, the simulated experiment will show or not the depth which separates the real behavior of HERMIES-IIB and its computed one.

This section aims to show what simulation can offer, what a major contribution it can be to help development on a real robot. By testing not only qualitative rules of navigation, is the action taken by the mobile robot appropriate or not, but also, according to precision criteria, testing quantitatively path planning methodologies. Without a simulator, one cannot predict if the navigation algorithm will work in all cases and prevent the robot to harm it. By taking some rather large margins when simulating, and testing a great number of possible paths, most of these problems will be avoided and a certain completeness be performed, by off-line programming.

6.3 GEOMETRIC MODELING/MAP BUILDING

This topic will deal with the area of geometric modeling, an important basic research field, and map building techniques.

A representation of the world, either 2D or 3D, though 3D modeling is much more complicated, is absolutely needed to perform map building algorithms. ARES's choice has been a representation based on a CSG tree. For 3D (or 2D) graphic animation, solids are modeled with polyhedral frames which involves many advantages as discussed previously.

Then, depending on this representation of the universe, map building schemes will take account for features of such or such sensor data analysis or vision system processing. The model will take account, too, for world's dynamics: some objects in the scene which location may have changed or which may have disappeared.

6.4 ADVANCED VISION SYSTEMS

At the CESAR laboratory, HERMIES-III will be equipped with a 3D laser range-finder. At Saclay, ARES lab has purchased a 3D camera to be coupled with a research tool which is a five-axis manipulator equipped with a laser beam. This arm is intended to work in hot cells and cut radioactive pieces of material for dismantling.

We propose this section to focus on the utilization of such advanced 3D vision systems: what are their main features (depth, reflectance, noise), how to simulate them and what does simulation provide.

This interesting topic will be the opportunity to share documents and results about these two advanced vision systems.

INTERNAL DISTRIBUTION

1. B. R. Appleton
2. J. E. Baker
3. D. L. Barnett
4. M. Beckerman
5. P. F. R. Belmans
6. P. L. Butler
7. G. de Saussure
8. J. R. Einstein
9. K. Fujimura
10. C. W. Glover

11–15. K. S. Harber

16. J. P. Jones
17. F. C. Maienschein

18–22. R. C. Mann

23–27. F. G. Pin

28. D. B. Reister
29. P. F. Spelt
30. F. J. Sweeney
31. M. A. Unseren
32. H. A. Vasseur
33. EP&MD Reports Office

34–35. Laboratory Records Department

36. Laboratory Records, ORNL–RC
37. Document Reference Section
38. Central Research Library
39. ORNL Patent Section

EXTERNAL DISTRIBUTION

40. Office of Assistant Manager, Energy Research and Development, Department of Energy, Oak Ridge Operations, P.O. Box 2001, Oak Ridge, TN 37831
41. James S. Coleman ER-15, Director of Engineering and Geosciences, Office of Basic Energy Sciences, U.S. Department of Energy, Washington, DC 20545
42. John J. Dorning, Department of Nuclear Engineering and Physics, Thorton Hall, McCormick Rd., University of Virginia, Charlottesville, VA 22901
43. Robert M. Haralick, Boeing Clairmont Egtvedt Prof., Department of Electrical Engineering, Director, Intelligent Systems Lab, University of Washington, 402 Electrical Engineering Bldg., FT-10, Seattle, WA 98195

44–46. Alain Kavenoky, Reactor Research and Development Division, CEA/IRDI/DEDR, 91191 Gif-Sur-Yvette Cedex, FRANCE

47. James E. Leiss, 13013 Chestnut Oak Dr., Gaithersburg, MD 20878
48. Lynne E. Parker, 125 Slade Street, Belmont, MA 02178
49. Neville Moray, Department of Mechanical and Industrial Engineering, University of Illinois, 1206 W. Green St., Urbana, IL 61801
50. Mary F. Wheeler, Mathematics Department, University of Houston, 4800 Calhoun, Houston, TX 77204-3476

51–60. Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831