STANFORD RESEARCH INSTITUTE

MENLO PARK, CALIFORNIA

File Copy

A MOBILE AUTOMATON: AN APPLICATION OF ARTIFICIAL INTELLIGENCE TECHNIQUES

by

Nils J. Nilsson
Stanford Research Institute
Menlo Park, California

January 1969

(Preliminary draft submitted for approval to the International Joint Conference on Artificial Intelligence, 7-9 May 1969, Washington, D.C.)

A MOBILE AUTOMATON: AN APPLICATION OF ARTIFICIAL INTELLIGENCE TECHNIQUES

by

Nils J. Nilsson

ABSTRACT

A research project applying artificial intelligence techniques to the development of integrated robot systems is described. The experimental facility consists of an SDS-940 computer and associated programs controlling a wheeled vehicle that carries a TV camera and other sensors. The primary emphasis is on the development of a system of programs for processing sensory data from the vehicle, for storing relevant information about the environment, and for planning the sequence of motor actions necessary to accomplish tasks in the environment. A typical task performed by our present system requires the robot vehicle to rearrange (by pushing) simple objects in its environment.

A novel feature of our approach is the use of a formal theorem-proving system to plan the execution of high-level functions as a sequence of other, perhaps lower level, functions. The execution of these in turn requires additional planning at lower levels. The main theme of the research is the integration of the necessary planning systems, models of the world and sensory processing systems into an efficient whole capable of performing a wide range of tasks in a real environment.

KEY WORDS

Robot

Robot System

Visual Processing

Problem Solving

Question Answering

Theorem Proving

Models of the World

Planning

Scene Analysis

Mobile Automaton

ACKNOWLEDGMENT

At least two dozen people at the Stanford Research Institute have made substantial contributions to the project that the author has the good fortune to describe in this paper. All of us express our appreciation to the Rome Air Development Center and the Advanced Research Projects Agency who supported this research.

I INTRODUCTION

At the Stanford Research Institute we are implementing a facility for the experimental study of robot systems. The facility consists of a time-shared SDS-940 computer, several core-loads of programs, a robot vehicle and special interface equipment.

Several earlier reports[1*] and papers[2-4] describing the project have been written; in this paper we shall describe its status as of early 1969 and discuss some of our future plans.

The robot vehicle itself is shown in Fig. 1. It is propelled by two stepping motors independently driving a wheel on either side of the vehicle. It carries a vidicon television camera and optical range-finder in a movable "head." Control logic on board the vehicle routes commands from the computer to the appropriate action sites on the vehicle. In addition to the drive motors, there are motors to control the camera focus and iris settings and the tilt angle of the head. (A motor to pan the head is not yet used by present programs.) Other computer commands arm or disarm interrupt logic, control power switches and request readings of the status of various registers on the vehicle. Besides the television camera and range-finder sensors, several "cat-whisker" touch-sensors are attached to the vehicle's perimeter. These touch sensors enable the vehicle to know when it bumps into something. Commands from the SDS-940 computer to the vehicle and information from the vehicle to the computer are sent over two special radio links, one for narrow-band telemetering and one for transmission of the TV video from the vehicle to the computer.

---

* References are listed at the end of this paper.

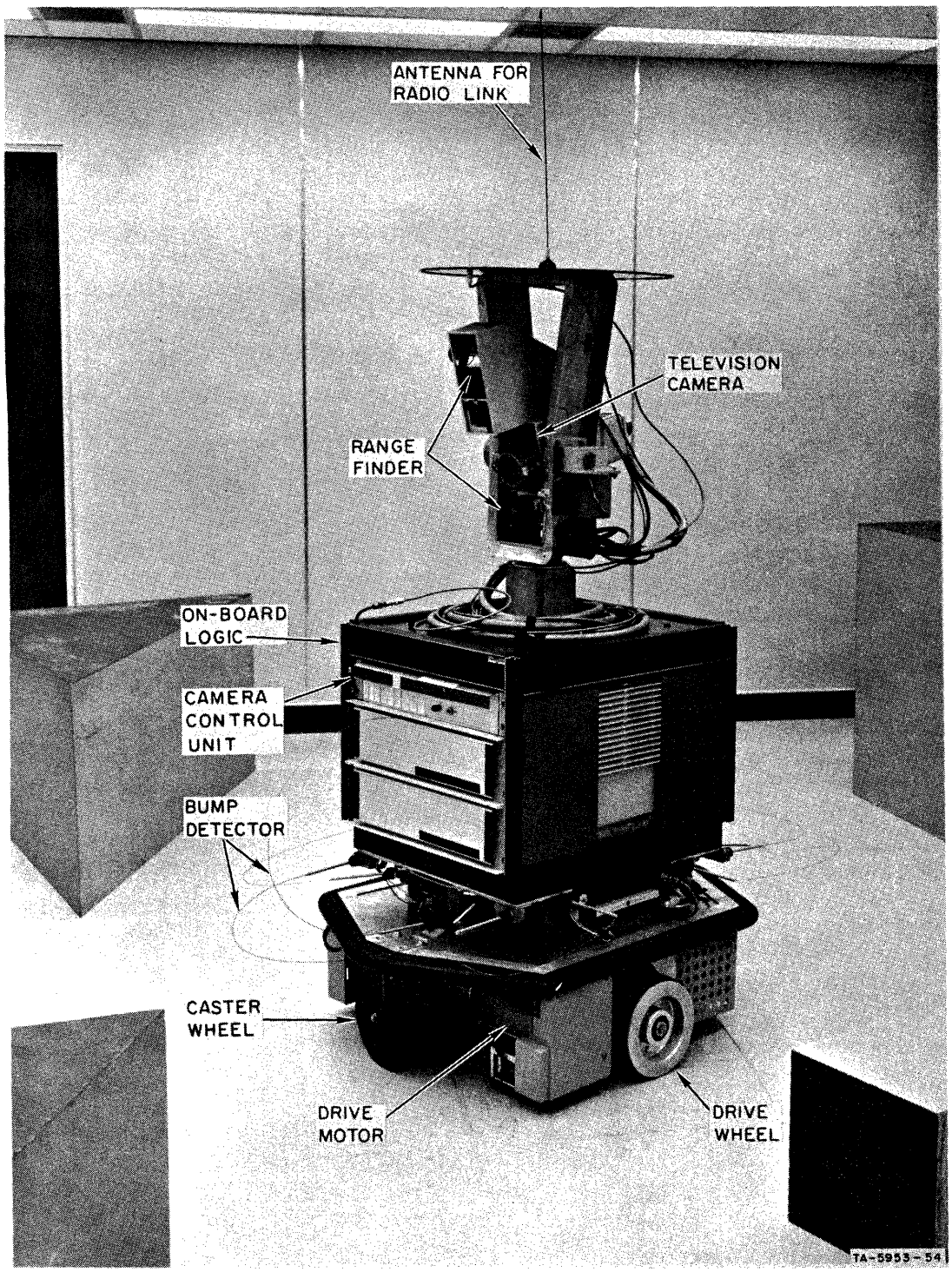


FIG. 1 THE ROBOT VEHICLE

The purpose of our robot research at SRI is to study processes for the real-time control of a robot system that interacts with a complex environment. We want the vehicle to be able to perform various tasks that require it to move about in its environment or to rearrange objects. In order to accomplish a wide variety of tasks rather than a few specific ones, a robot system must have very general methods. What is required is the integration in one system of many of the abilities that are usually found separately in individual Artificial Intelligence programs.

We can group most of the needed abilities into three broad classes: (1) problem-solving, (2) modelling, and (3) perception:

(1) Problem-Solving

A robot system accomplishes the tasks given it by performing a sequence of primitive actions, such as wheel motions and camera readings. For efficiency, a task should first be analyzed into a sequence of primitive actions calculated to have the desired effect. This process of task analysis is often called planning because it is accomplished before the robot begins to act. Obviously in order to plan, a robot system must "know" about the effects of its actions.

(2) Modelling

A body of knowledge about the effects of actions is a type of model of the world. A robot problem-solving system uses the information stored in the model to calculate what sequence of actions will cause the world to be in a desired state. As the world changes, either by the robot's own actions or for other reasons, the model must be updated to record these changes. Also new information learned about the world should be added to the model.

(3) Perception

Sensors are necessary to give a robot system new information about the world. By far the most important sensory system is vision, since it allows direct perception of a good sized piece of the world beyond the range of touch. Since we assume that a robot system will not always have stored in its model every detail of the exact configuration of its world and thus cannot know precisely the effects of its every action, it also needs sensors with which to check predicted consequences against reality as it executes its plans.

The integration of such abilities into a smoothly-running, efficient system presents both important conceptual problems and serious practical challenges. For example, it would be infeasible for a single problem-solving system (using a single model) to attempt to calculate the long chains of primitive actions needed to perform lengthy tasks. A way around this difficulty is to program a number of coordinating "action-units" each with its own problem-solving system and model and each responsible for planning and executing a specialized function. In planning how to perform its particular function, each action-unit knows the effects of executing functions handled by various of the other action-units. With this knowledge it composes its plan as a sequence of other functions (with the appropriate arguments) and leaves the planning required for each of these functions up to the action-units responsible for executing them at the time they are to be executed.

Such a system of interdependent action-units implies certain additional problems involving communication of information and transfer of control between units. When such a system is implemented on a serial

computer with limited core memory, obvious practical difficulties arise connected with swapping program segments in and out of core and handling interrupts in real time. The coordinated action-unit scheme serves as a useful guide in explaining the operation of our system, even though practical necessities have dictated occasional deviations from this scheme in our implementation. In the next section we shall discuss the problem-solving processes and models associated with some specific functions of the present SRI robot system.

II SOME SPECIFIC FUNCTIONS OF THE ROBOT SYSTEM AND THEIR ASSOCIATED PROBLEM-SOLVING PROCESSES AND MODELS

A. Low Level Functions

The robot system is capable of executing a number of functions that vary in complexity from the simple ability to turn the drive wheels a certain number of steps to the ability to collect a number of boxes by pushing them to a common area of the room. The organization of these functional action-units is not strictly hierarchical, although for descriptive convenience we will divide them into two classes: low level and high level functions.

Of the functions that we shall mention here, the simplest are certain primitive assembly language routines for moving the wheels, tilting the head, reading a TV picture and so on. Two examples of these are MOVE and TURN; MOVE causes the vehicle to roll in a straight line by turning both drive wheels in unison, and TURN causes the vehicle to rotate about its center by turning the drive wheels in opposite directions. The arguments of MOVE and TURN are the number of steps that the drive wheels are to turn (each step resulting in a vehicle motion of 1/32 inch) and

"status" arguments that allow queries to be made about whether or not the function has been completed.*

Once begun, the execution of any function either proceeds until it is completed in its normal manner or until it is halted by one of a number of "abnormal" circumstances such as the vehicle bumping into unexpected objects, overload conditions, resource exhaustion and so on. Under ordinary operation, if execution of MOVE results in a bump, motion is stopped automatically by a special mechanism on the vehicle. This mechanism can be overridden by a special instruction from the computer however, to enable the robot to push objects.

The problem-solving systems for MOVE and TURN are trivial; they need only to calculate what signals shall be sent to registers associated with the motors in order to complete the desired number of steps.

At a level just above MOVE and TURN is a function whose execution causes the vehicle to travel directly to a point specified by a pair of (x,y) coordinates. This function is implemented in the FORTRAN routine LEG. The model used by LEG contains information about the robot's present (x,y) location and heading relative to a given coordinate system and information about how far the vehicle travels for each step applied to the stepping motors. This information is stored along with some other special constants in a structure called the PARAMETER MODEL. Thus for a given (x,y) destination as an argument of LEG, LEG's problem-solving system

---

* Our implementation allows a program calling routines like MOVE or TURN to run in parallel with the motor functions they initiate.

calculates appropriate arguments for a TURN and MOVE sequence and then executes this sequence. Predicted changes in the robot's location and heading caused by execution of MOVE and TURN are used to update the PARAMETER MODEL.

Ascending one more level in our system we encounter a group of FORTRAN "two-letter" routines whose execution can be initiated from the teletype. Our action-unit system ceases to be strictly hierarchical at this point since some of the two-letter commands can cause others to be executed.

One of these two letter commands, EX, takes as an argument a sequence of (x,y) coordinate positions. Execution of EX causes the robot to travel from its present position directly to the first point in the sequence, thence directly to the second, and so on until the robot reaches the last point in the sequence. The problem-solving system for EX simply needs to know the effect caused by execution of a LEG program and composes a chain of LEG routines each with arguments provided by the successive points specified in the sequence of points. Under ordinary operation, if one of these LEG routines is halted due to a bump, EX backs the vehicle up slightly and then halts. A special feature of our implementation is the ability to arm and service interrupts (such as caused by bumps) at the FORTRAN programming level.

Another two-letter command PI causes a picture to be read after the TV camera has been aimed at a specified position on the floor. The problem-solving system for PI thus calculates the appropriate arguments for a TURN routine and a head-tilting routine; PI then causes these to be executed, reads in a picture from the TV camera, and performs processing

necessary to extract information about empty areas on the floor. (Details of the picture processing programs of the robot system are described in Section III below.)

The ability to travel by the shortest route to a specified goal position along a path calculated to avoid bumping into obstacles is provided by the two letter command TE. Execution of TE involves the calculation of an appropriate sequence of points for EX and the execution of EX. This appropriate sequence is calculated by a special problem solving system embodied in the two-letter command PL.

The source of information about the world used by PL is a planar map of the room called the GRID MODEL. The GRID MODEL is a hierarchically organized system of four by four grid cells. Initially the "whole world" is represented by a four-by-four array of cells. A given cell can be either empty (of obstacles), full, partially full, or unknown. Each partially full cell is further subdivided into a four by four array of cells and so on until all partially full cells represent areas of some suitably small size. (Our present system splits cells down to a depth of three levels representing a smallest area of about 12 inches.)

Special "model maintenance" programs insure that the GRID MODEL is automatically updated by information about empty and full floor areas gained by either successful execution or interruption of MOVE commands.

The PL program first uses the GRID MODEL to compute a network or graph of "nodes." The nodes correspond to points in the room opposite corners of obstacles; the shortest path to a goal point will then pass through a sequence of a subset of these nodes. In Fig. 2 we show a complete GRID MODEL of a room containing three objects. The robot's position,

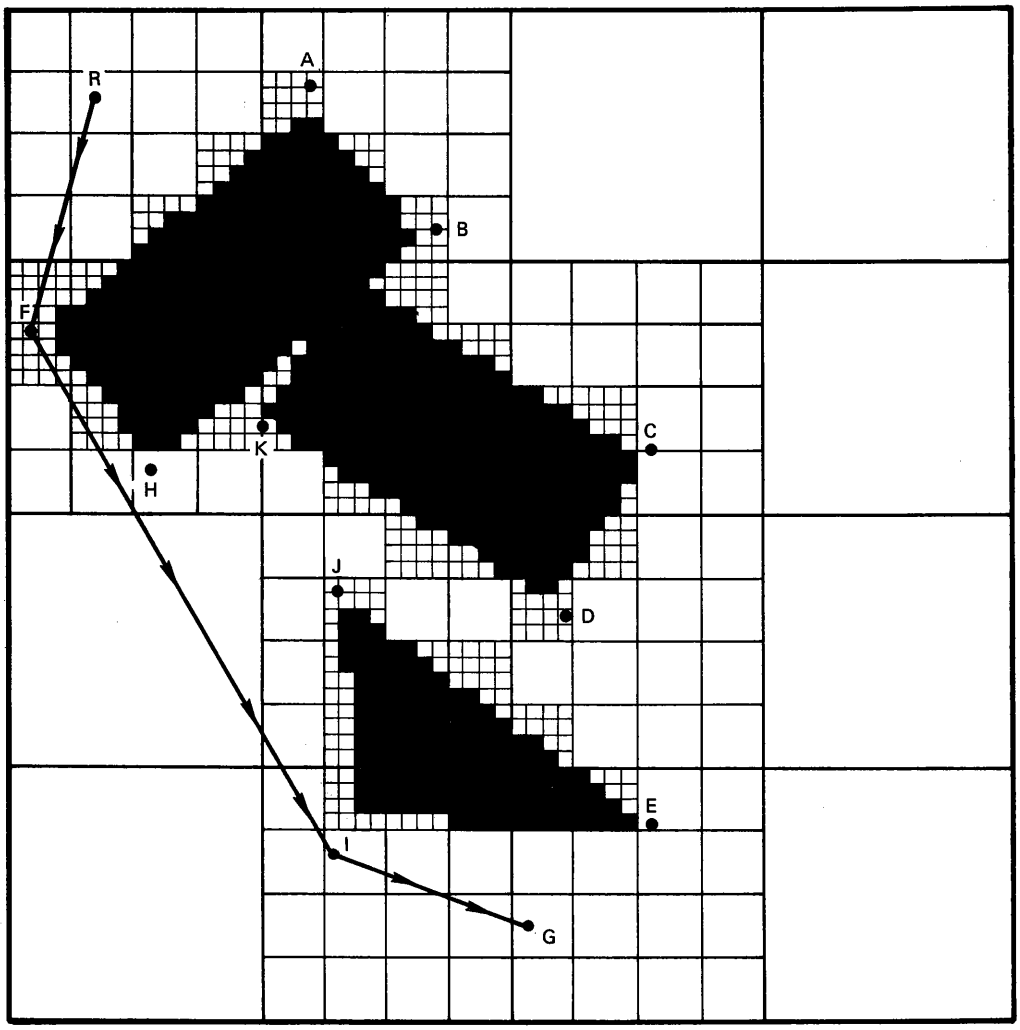


FIG. 2 A GRID MODEL OF A ROOM WITH THREE OBJECTS

marked "R," and the goal position, marked "G," together with the nodes A,B,C,D,E,F,H,I,J and K are shown overlain on the GRID MODEL. The program PL then determines that the shortest path is the sequence of points, R,F,I, and G by employing an optimal graph-searching algorithm developed by Hart, et al.[5]

If the GRID MODEL map of the world contains unknown space, PL must decide whether or not to treat this unknown space as full or empty. Currently, PL multiplies the length of any segment of the route through unknown space by a parameter k. Thus if k=1, unknown space is treated as empty; values of k greater than unity cause routes through known empty space to be preferred to possibly shorter routes through unknown space.

Execution of TE is accomplished by first reading and processing a picture (using PI with the camera aimed at the goal position) and taking a range-finder reading. The information about full and empty floor areas thus gained is added to the GRID MODEL. A route based on the updated GRID MODEL is then planned using PL, and then EX is executed using the arguments calculated by PL. If the EX called by TE is halted by a bump, a procedure attempts to manuever around the interfering obstacle, and then TE is called to start over again. Thus, vision is used only at the beginning of a journey and when unexpected bumps occur along the journey.

Although our present robot system does not have manipulators with which to pick up objects, it can move objects by pushing them. The fundamental ability to push objects from one place to another is programmed into another two-letter FORTRAN routine called PU. Execution of PU causes the robot to push an object from one named position along a straight line path to another named position. The program PU takes five arguments:

the (x,y) coordinates of the object to be pushed, the "size" or maximum extent of the object about its center of gravity, and the (x,y) coordinates of the spot to which the object is to be pushed. The problem-solving system for PU assembles an EX, a TURN, and two MOVE commands into a sequence whose execution will accomplish the desired push. First a location from which the robot must begin pushing the object is computed. Then PL is used to plan a route to this goal location. The sequence of points along the route serves as the argument for EX which is then executed. (Should EX be stopped by a bump, PU is started over again.) Next PU's problem-solving system (using the PARAMETER model) calculates an argument for TURN that will point the robot in the direction that the object is to be pushed. A large argument is provided for the first MOVE command so that when it is executed, it will bump into the object to be pushed and automatically halt. After the bump and half the automatic stopping mechanism on the vehicle is overridden and the next MOVE command is executed with an argument calculated to push the object the desired distance.

B. Higher Level Functions

As we ascend to higher level functions, the required problem-solving processes must be more powerful and general. We want our robot system to have the ability to perform tasks possibly requiring quite complex logical deductions. What is needed for this type of problem-solving is a general language in which to state problems and a powerful search strategy with which to find solutions. We have chosen the language of first-order predicate calculus in which to state high level problems for the robot. These problems are then solved by an adaptation of a "Question Answering System" QA-3, based on "resolution" theorem-proving methods.[6-9]

As an example of a high level problem for the robot, consider the task of moving (by pushing) three objects to a common place. This task is an example of one that has been executed by our present system. If the objects to be pushed are, say, OB1, OB2, and OB3, then the problem of moving them to a common place can be stated as a "conjecture" for QA-3:

$$\exists p,s\{\text{POSITION (OB1,p,s)} \wedge \text{POSITION (OB2,p,s)} \wedge \text{POSITION (OB3,p,s)}$$

(That is, "There exists a situation s and a place p, such that OB1, OB2, and OB3 are all at place p in situation s.") The task for QA-3 is to "prove" that this conjecture follows from "axioms" that describe the present position of objects and the effects of certain actions.

Our formulation of these problems for the theorem-prover involves specifying the effects of actions in terms of functions that map situations into new situations. For example, the function PUSH (x,p,s) maps the situation s into the situation resulting by pushing object x into place p. Thus two axioms needed by QA-3 to solve the pushing problem are:

$$\forall x,p,s \text{ POSITION (x,p, PUSH (x,p,s))}$$

and

$$\forall x,y,p,q,s[\text{POSITION (x,p,s)} \wedge \sim \text{SAME (x,y)} \Rightarrow \text{POSITION (x,p,PUSH (y,q,s))}]$$

The first of these axioms states that if in an arbitrary situation s, an arbitrary object x is pushed to an arbitrary place p, then a new situation, PUSH (x,p,s), will result in which the object x will be at position p. The second axiom states that any object will stay in its old place in the new situation resulting by pushing a different object.

In addition to the two axioms just mentioned we would have others describing the present positions of objects. For example, if OB1 is at coordinate position (3,5) in the present situation, we would have:

POSITION (OB1, (3,5), PRESENT)

(This information is provided automatically by routines which scan the GRID MODEL giving names to clusters of full cells and noting the locations of these clusters.)

In proving the truth of the conjecture, the theorem-prover used by QA-3 also produces the place p and situation s that exist. That is, QA-3 determines that the desired situation s is:

s = PUSH (OB3,(3,5), PUSH (OB2,(3,5), PRESENT))

All of the information about the world used by QA-3 in solving this problem is stored in the form of axioms in a structure called the AXIOM MODEL. In general, the AXIOM MODEL will contain a large number of facts, more than are necessary for any given deduction.

Another LISP program examines the composition of functions calculated by QA-3 and determines those lower level FORTRAN two-letter commands needed to accomplish each of them. In our present example, a sequence of PU commands would be assembled. In order to calculate the appropriate arguments for each PU, QA-3 is called again, this time to prove conjectures of the form:

$$\exists p,w\{\text{POSITION (OB2,p,PRESENT)} \wedge \text{SIZE (OB2,w)}\}$$

Again the proof produces the p and w that exist, thus providing the necessary position and size arguments for PU. (Size information is also automatically entered into the AXIOM MODEL by routines that scan the GRID MODEL.)

In transferring control between LISP and FORTRAN (and also between separate large FORTRAN segments), use is made of a special miniature monitor system called the VALET. The VALET handles the process of dismissing program segments and starting up new ones using auxiliary drum storage for transferring information between programs.

The QA-3 theorem proving system allows us to pose quite general problems to the robot system, but further research is needed on adapting theorem-proving techniques to robot problem-solving in order to increase efficiency.* The generality of theorem-proving techniques tempts us to use a single theorem-prover (and axiom set) as a problem-solver (and model) for all high level robot abilities. We might conclude, however, that efficient operation requires a number of coordinating action-unit structures each having its own specialized theorem-prover and axiom set and each responsible for relatively narrow classes of functions.

Another LISP program enables commands stated in simple English to be executed. It also accepts simple English statements about the environment and translates them into predicate calculus statements to be stored as axioms. English processing by this program is based on work by L. S. Coles.[10] English commands are ordinarily translated into predicate calculus conjectures for QA-3 to solve by producing an appropriate sequence of subordinate functions. For some simple commands, the theorem-prover is bypassed and lower level routines such as PU, TE, etc., are called directly.

---

* We can easily propose less fortuitous axiomatizations for the "collecting objects task" that would prevent QA-3 from being able to solve it.

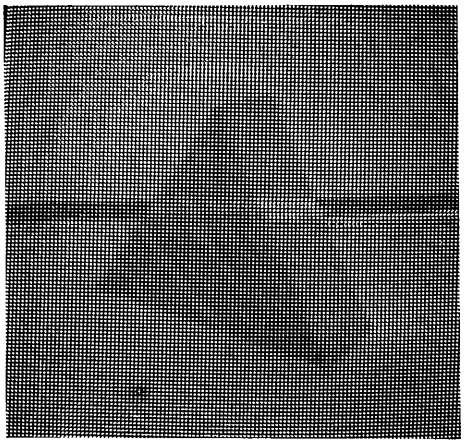
The English program also accepts simple English questions that require no robot actions. For these it uses QA-3 to discover the answer, and then it delivers this answer in English via the teletypewriter. (Task execution can also be reported by an appropriate English output.) Further details on the natural language abilities of the robot system are described in a paper by Coles[11] published in this Proceedings.

III VISUAL PERCEPTION

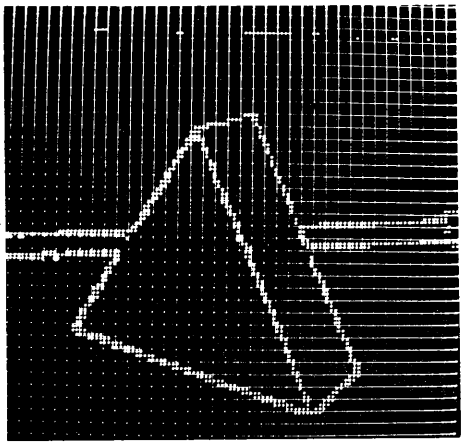
Vision is potentially the most effective means for the robot system to obtain information about its world. The robot lives in a rather antiseptic but nevertheless real world of simple objects--boxes, wedges, walls, doorways, etc. Its visual system extracts information about that world from a conventional TV picture. A complete scene analysis would produce a description of the visual scene, including the identification and location of all visible objects. While this is our ultimate goal, our current vision programs merely identify empty floor space, regions on the floor into which the robot is free to move. This is done by first producing a line drawing representation of the scene, and then by analyzing this line drawing to determine the empty floor space. In this section we shall describe briefly how this is done; further details can be found in other reports and papers.[1,4]

A. Production of a Line Drawing

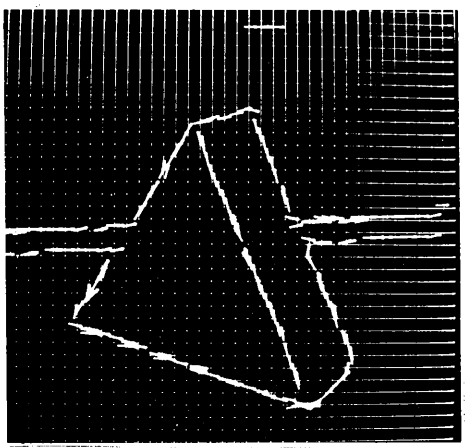
The line drawing is produced from the TV picture by a series of essentially local operations. The first step is to read the TV picture into the computer. The picture, obtained from a conventional vidicon camera, is digitized and stored as a 4-bit (16 intensity levels) 120 x 120 array. This digitized representation can be displayed for visual inspection, and Fig. 3a shows a digitized version of a scene containing a wedge-shaped object.



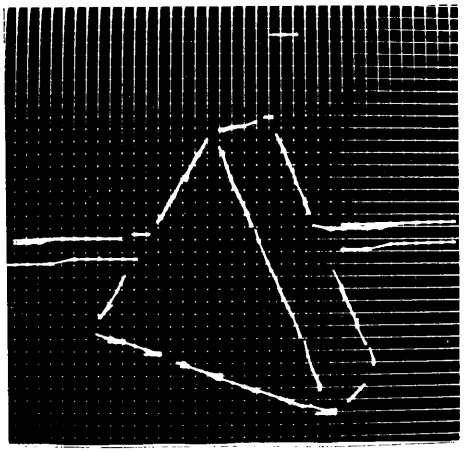
(a) Digitized Image



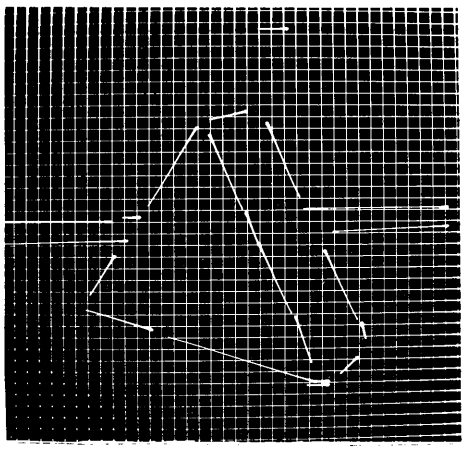
(b) Differentiated Image



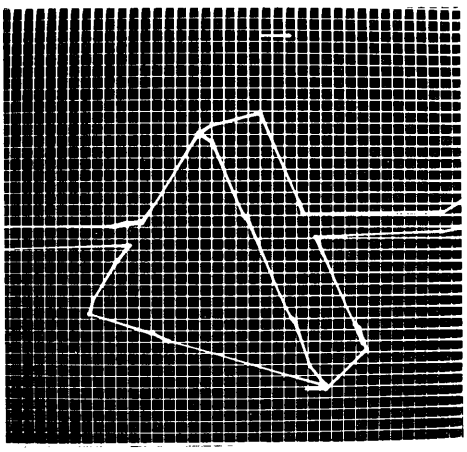
(c) Line-Segment Mask Responses



(d) Grouped Line Segments



(e) Long-Line Fits



(f) Joined Lines

TA-5953-52

FIG. 3 EXAMPLE OF VISUAL PROCESSING STEPS

The digitized image is then processed to determine which pisture points have intensities that are sufficiently different from thos of its immediate neighbors. Several techniques have been described in the literature to produce such a "differentiated" or outline-enhanced pisture; we are using an approximation to a method proposed by Roberts.[12] After "differentiation" the image is as shown in Fig. 3b.

The next step is to attempt to determine locally the direction of outlines of the picture. To do so we use a set of "feature-detecting" masks. Each mask covers a 7 x 7 sub-frame of the picture; when a sufficient number of picture points of the differentiated image lie along a short line segment, then a particular mask matched to a line segment of that direction responds. We use 16 masks matched to 16 different segment directions and test for responses with masks centered everywhere on the picture. The result of this short-line segment detecting operation is shown in Fig. 3c. In that figure we have used short line segments to represent the corresponding mask responses.

The next stage of processing, called "grouping," fills in some of the gaps and throws away isolated line segments. Whenever line segments are both sufficiently close in location and sufficiently the same in direction they are linked together in a "group." Line segment groups having too few numbers are then thrown away. The result of grouping for our example image is shown in Fig. 3d.

Next, each group is fitted by a single long straight line. The result is shown in Fig. 3e. Note that gaps still exist, particularly near corners. These are largely taken care of by a routine called JOIN that in effect manufactures special masks to see which of several candidate

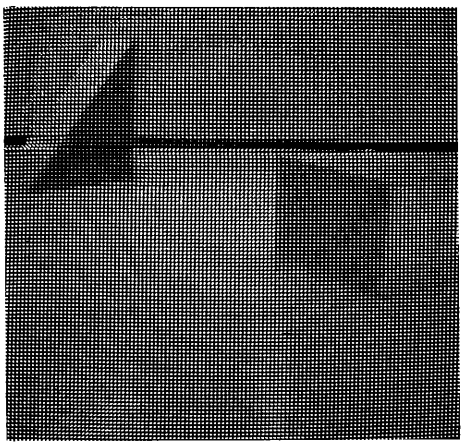
methods for joining end points is best supported by the original picture data. After JOIN, our example image is as shown in Fig. 3f. In Fig. 4 we show a corresponding sequence of images for a slightly more complicated scene.

B. Analysis of the Line Drawing

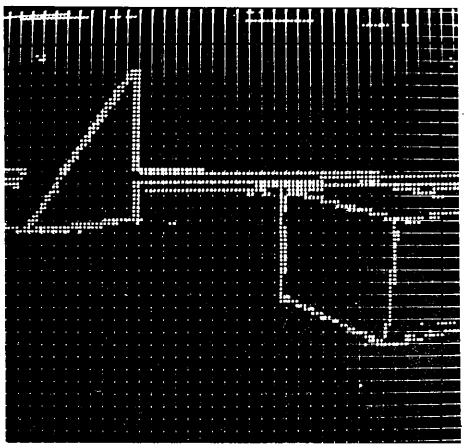
The line drawing produced by JOIN preserves much of the information in the quantized picture in a very compact form. However, the line drawing often contains flaws in the form of missing or extra line segments, and to circumvent these flaws during analysis requires knowledge of or hypotheses about the nature of the robot's world.

The only information currently being extracted from the line drawing is a map of the open floor space. A program called FLOOR BOUNDARY analyzes the line drawing to find the places where the walls or other objects meet the floor. The FLOOR BOUNDARY program first checks to be sure that the area along the extreme bottom of the picture is indeed "floor." It then uses a special procedure to follow along the lines nearest the bottom of the picture (filling gaps where necessary) to delineate a conservative estimate of this region of floor. In Fig. 5 we show the floor boundaries extracted from the scenes of Figs. 3 and 4.

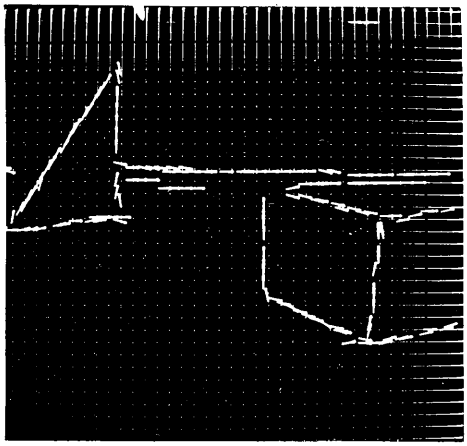
Because we know that the floor that the robot "sees" is an extension of the same floor on which it rests, and because we know certain parameters such as the acceptance angle and height of the camera, and the pan and tilt angles, we can compute the actual location in three-dimensional space of a line corresponding to the bottom of the picture. Similarly, we can compute lines corresponding to the sides of the picture and of the floor blundary. This computation gives us an irregular polygon on the floor that is known to be empty. It is this empty area that is then finally entered into the GRID MODEL.



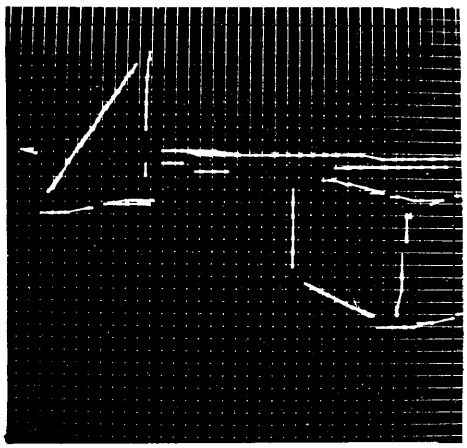
(a) Digitized Image



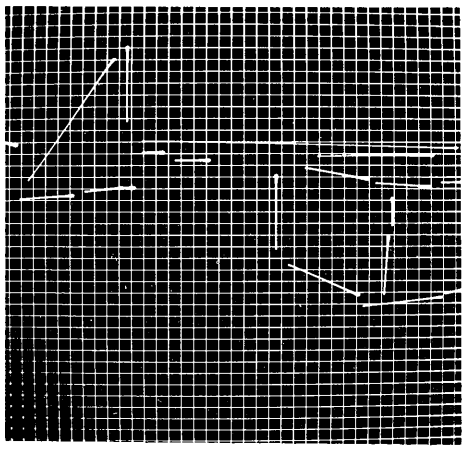
(b) Differentiated Image



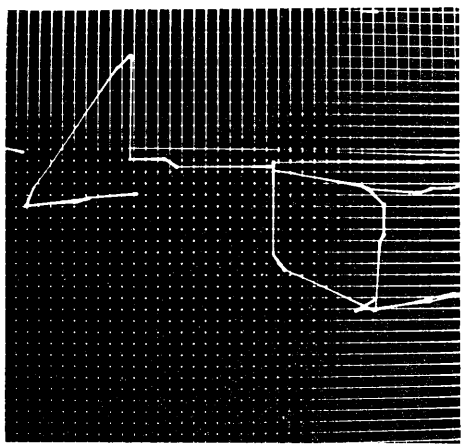
(c) Line-Segment Mask Responses



(d) Grouped Line Segments



(e) Long-Line Fits

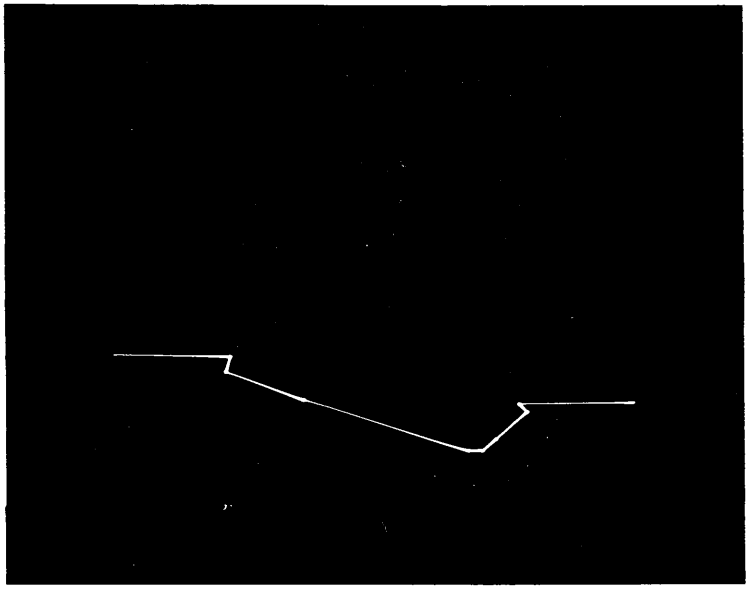


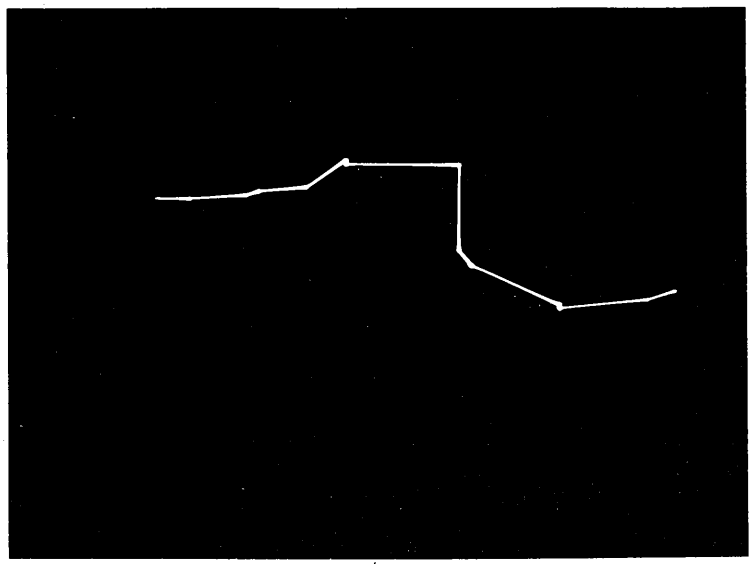
(f) Joined Lines

TA-5953-53

FIG. 4 A SECOND EXAMPLE OF VISUAL-PROCESSING STEPS

(a)





(b)

FIG. 5 FLOOR BOUNDARIES

Although information about known empty space is very useful, it is clear that much more information can be extracted from a visual scene. Much of our current vision system research is being directed at locating and identifying various objects, whether partially or totally in the field of view. Some of the approaches we are taking are described in the next section.

IV CONCLUSIONS

There are several key questions that our work has helped to put into focus. Given that a robot system will involve the successful integration of problem-solving, modelling, and perceptual abilities, there are many research questions concerning each of these. Let us discuss each in turn.

1. Problem-Solving

Our somewhat hierarchical organization of problem-solvers and models seems a natural, even if ad hoc, solution to organizing complex behavior. Are there alternatives? Will the use of theorem-proving techniques provide enough generality to permit a single general purpose problem solver or will several "specialist" theorem-provers be needed to gain the required efficiency?

Other questions concern the use of theorem-proving methods for problem-solving. How do they compare with the "production methods" as used by the General Problem Solver (GPS) or with the procedural language approach as developed by Fikes?[13] Perhaps some combination of all of these will prove superior to any of them; perhaps more experience will show that they are only superficially different.

Another question is: To what level of detail should behavioral plans be made before part of the plan is executed and the results checked

against perceptual information? Although this question will not have a single answer we need to know upon what factors the answer depends.

Our problem-solving research will also be directed at methods for organizing even more complex robot behavior. We hope eventually to be able to design robot systems capable of performing complex assembly tasks requiring the intelligent use of tools and other materials.

2. Modelling

Several questions about models can be posed: Even if we continue to use a number of problem-solvers, must each have its own model? To what extent can the same model serve several problem-solvers? When a perceptual system discovers new information about the world, should it be entered directly into all models concerned? In what form should information be stored in the various models? Should provisions be made for forgetting old information? Can a robot system be given a simple model of its own problem-solving abilities? Ensuing research and experience with our present system should help us with these questions.

3. Visual Perception

The major difficulty we have encountered in extending the capability of the vision system has been the cascading of errors during the various stages of processing. The lowest level program inevitably makes errors, and these errors are passed up to the next higher level. Thus, errors accumulate until the highest level program is asked, among other things, to correct the compounded errors of all the programs below it.

To circumvent these problems, we have begun experimenting with a quite different program organization in which a high-level driver program,

endowed with knowledge of the robot's world, actively seeks information from low-level subroutines operating directly on the pictorial data. When a given subroutine is exercised, the driver program checks to see if the results are consistent with the information already accumulated. If not, other subroutines may be called, or the results of previously called subroutines may be reconsidered in the light of current information. We anticipate that this organization will lessen the compounding effect of errors and will provide a more graceful means of recovering from the errors that are committed.

A number of obvious questions come to mind. How can information about the world best be incorporated in the driver program? How can the driver use facts about the world obtained from the model? What strategy should the driver use to explore the picture with its repertoire of subroutines? Since "facts" obtained from either the model or the subroutines are subject to error, it is natural to accompany them by some confidence or probability measure. How should these be computed? How should the results of several subroutines be combined, since, loosely speaking, we have strong statistical dependence? How can we augment the current repertoire of subroutines with others to make use of such properties as color, texture, and range? We are presently actively involved in seeking answers to these and related questions. Early results with this approach have been very encouraging, and we hope to provide more details in a future paper.

The main theme of the project has been and will continue to be the problem of system integration. In studying robot systems that interact with the real world, it seems extremely important to build and program

a real system and to provide it with a real environment. Whereas much can be learned by simulating certain of the necessary functions (we use this strategy regularly), many important issues are likely not to be anticipated at all in simulations. Thus questions regarding, say the feasibility of a system of interacting action-units for controlling a real robot can only be confronted by actually attempting to control a real robot with such a system. Questions regarding the suitability of candidate visual processing schemes can most realistically be answered by experiments with a system that needs to "see" the real world. Theorem-proving techniques seem adequate for solving many "toy" problems; will the full generality of this approach really be exploitable for directing the automatic control of mechanical equipment in real-time?

The questions that we have posed in this section are among those that must be answered in order to develop useful and versatile robot systems. Experiments with a facility such as we have described appears to be the best way to elicit the proper questions and to work toward their answers.

REFERENCES

1. N. Nilsson, et al, "Application of Intelligent Automata to Reconnaissance," Contract AF 30(602)-4147, SRI Project 5953, Stanford Research Institute, Menlo Park, California. Four Interim Reports and one Final Report dated December 1968.

2. C. A. Rosen and N. J. Nilsson, "An Intelligent Automaton," IEEE International Convention Record, Part 9 (1967).

3. B. Raphael, "Programming a Robot," Proc. IFIP Congress 68, Edinburgh, Scotland (August 1968).

4. G. E. Forsen, "Processing Visual Data with an Automaton Eye," in Pictorial Pattern Recognition (Thompson Book Company, Washington, D.C., 1968).

5. P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Trans. on Systems Science and Cybernetics, Vol. SSC-4, No. 2, pp. 100-107, (July 1968).

6. C. Green and B. Raphael, "The Use of Theorem-Proving Techniques in Question-Answering Systems," Proc. 1968 ACM Conference, Las Vegas, Nevada (August 1968).

7. C. Green and B. Raphael, "Research on Intelligent Question-Answering Systems," Scientific Report No. 1, Contract AF 19(628)-5919 SRI Project 6001, Stanford Research Institute, Menlo Park, California (May 1967).

8. B. Raphael, "Research on Intelligent Question-Answering Systems," Final Report, Contract AF 19(628)-5919, SRI Project 6001, Stanford Research Institute, Menlo Park, California (May 1968).

9. C. Green, "Theorem-Proving by Resolution as a Basis for Question-Answering Systems," Machine Intelligence 4, B. Meltzer and D. Michie, (Eds.), Edinburgh University Press (Edinburgh, Scotland; to appear 1969).

10. L. S. Coles, "An On-Line Question-Answering System with Natural Language and Pictorial Input," Proc. 1968 ACM Conference, Las Vegas, Nevada (August 1968).

11. L. S. Coles, "Talking with a Robot in English," Proc. of the International Joint Conference on Artificial Intelligence, Wash. D.C., (May 1969).

12. L. G. Roberts, "Machine Perception of Three-Dimensional Solids," Optical and Electro-Optical Information Processing (MIT Press, 1965).

13. Fikes, R., "A Study in Heuristic Problem-Solving: Problems Stated as Procedure," Proc. of Fourth Systems Symposium, held at Case Western Reserve University, (Nov. 1968) (to be published).