Nilsson

May 1974

Progress Report

ARTIFICIAL INTELLIGENCE—RESEARCH AND APPLICATIONS

By: N. J. NILSSON, G. AGIN, H. BARROW, A. E. BRAIN, B. DEUTSCH, R. E. FIKES, T. D. GARVEY, D. C. LYNCH, E. D. SACERDOTI, J. M. TENENBAUM

Prepared for:

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY
ARLINGTON, VIRGINIA 22209

CONTRACT DAHC04-72-C-0008

STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U.S.A.

STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U.S.A.

Form Approved
Budget Bureau No. 22-R0293

May 1974

Progress Report
Covering the Period 9 October 1972 through 8 March 1974
Stanford Research Institute Project 1530

ARTIFICIAL INTELLIGENCE—RESEARCH AND APPLICATIONS

Submitted by

NILS J. NILSSON
Project Leader
(415) 326-6200, Ext. 2311

Authors

N. J. NILSSON, A. E. BRAIN, T. D. GARVEY, J. M. TENENBAUM, G. AGIN, B. DEUTSCH, D. C. LYNCH, H. BARROW, R. E. FIKES, E. D. SACERDOTI

CONTRACT DAHC04-72-C-0008
ARPA Order Number 1943
Program Code Number 2D30

Effective Date of Contract: 8 October 1971
Contract Expiration Date: 9 October 1974
Amount of Contract: $1,918,128.00

Prepared for

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY
ARLINGTON, VIRGINIA 22209

Approved by:

PETER E. HART, *Acting Director*
Artificial Intelligence Center

BONNAR COX, *Executive Director*
Information Science and Engineering Division

Copy No. ....

ABSTRACT

This progress report describes the work performed during the most recent year and a half of a program of research in the field of Artificial Intelligence (AI) being conducted at SRI. Our research program concentrates especially on the development of systems that can automatically generate and execute complex plans and that can obtain information about their environments through the sense of vision.

To provide a specific focus for our research, we have set ourselves the goal of building a "computer-based consultant" system that will serve as an expert consultant to a human apprentice. Together the system and the apprentice will be engaged in a task of checking out and repairing electromechanical equipment in a "workstation" domain. We have developed a plan in which we have specified the capabilities of a system that can be demonstrated after five years of work (1978), and in which we have identified and discussed the major research problem areas involved. These include modeling and representation, automatic planning and problem solving, machine vision, natural language communication and system integration. This report describes progress we have made in these areas.

In problem solving and modeling, we have begun implementation of a new hierarchical plan-generation and execution-monitoring system that uses a representation called a "procedural net." The system generates a network structure that allows a plan to be presented at varying levels of detail in order to match the apprentice's ability. We have also designed and implemented in QLISP a program that simulates the top-level behavior of an automatically controlled air compressor. This program models many of the cause-and-effect relations that determine the compressor's behavior, and can thus be used as an information source for programs that do trouble shooting and question answering.

In vision we have implemented an interactive scene interpretation system specifically designed for expressing basic perceptual concepts to a computer and experimenting with them using pictorial examples. Using this system, we build up more complicated structures from basic concepts using both symbolic and graphic means. We have also designed and implemented, in a preliminary fashion, a semantic scene-understanding system in which locally ambiguous scene interpretations are disambiguated using global constraints supplied by a user. The hallmark of the initial work is the redundancy of suitable constraints in typical scenes and the ease with which such constraints may be interactively specified.

The feasibility of a scanning, time-of-flight laser rangefinder that will develop a field of depth values was demonstrated with the completion of a prototype version.

In natural language we have begun analyzing apprentice-expert dialogs and will be using this information to begin the design of a natural language understanding component.

The report concludes with a brief discussion of the SRI Artificial Intelligence Center computer facility and a list of presentations and publications by the project staff.

CONTENTS

APPENDICES

ILLUSTRATIONS

TABLES

I INTRODUCTION

A. General

This progress report describes the work performed during the most recent year and a half of a program of research in the field of artificial intelligence (AI). The work reported here began in October 1972 and builds upon previous AI research performed at SRI and elsewhere.

During the course of the last year and a half we have documented our technical work in several reports, journal articles, and presentations. (These are listed in Section VII.) Our intention in this report is to provide an overview of the project, rather than to reproduce the contents of already published documents.

B. Background

1. The Mobile Robot Domain

For several years AI work at SRI has been directed especially at the development of systems that can automatically generate and execute complex plans and that can obtain information about their environments through the sense of vision. Both planning and vision rely heavily on large-scale models or knowledge bases about the domain of interest. To provide specific semantics for our research on planning, vision, and modeling we worked with the domain of a mobile robot in a laboratory environment consisting of rooms, doorways, and simple rectilinear objects.

Our work with the robot system reached a performance peak toward the end of 1972 with a series of experiments demonstrating rather sophisticated plan generation, execution, and learning abilities. This work is summarized in our last annual report [Ref. 1].* Rather than continue with more sophisticated robot systems we agreed with our ARPA sponsors to shift our experimental domain to one in which practical applications of our research might be more immediate.

2. The Computer-Based Consultant Domain

After intensive study of possible domains we selected one that is particularly relevant to DoD needs as well as having high scientific value. We set ourselves the goal of producing a computer-based consultant system that will serve as an expert consultant to a human apprentice. Together the system and the apprentice will be engaged in a task of checking out and repairing electromechanical equipment in a "workstation" domain.

We have specified the capabilities of a demonstration system that we think is achievable in five years, and we have identified the major research problem areas involved. These include modeling and

*References are listed at the end of this report.

representation, automatic planning and problem solving, machine vision, natural language communication, and system integration.

The demonstration system will give the human apprentice advice about how to diagnose equipment faults, how to repair them, and how to assemble and disassemble equipment. It will also give instructions about workstation procedures and information about the names of components, uses of tools, and so on. The level of detail of this advice and information will be automatically tailored to the individual needs of the apprentice. The apprentice will be able to ask specific questions or seek general guidance. An important means of communication will be through natural language. The system will have the ability to understand continuous speech about the domain of interest.

Another important perceptual channel will be computer based vision. The visual subsystem will be able to identify equipment components, inspect assemblies, and generally monitor progress in the workstation.

The demonstration system itself will serve to illustrate the feasibility of applying the constituent computer technology to problems of equipment operation, maintenance, and repair; to remote site and vehicle support; and to similar applications.

One of the tasks that we performed during this reporting period was to complete a comprehensive five-year research plan for the computer-based consultant project. This plan is thoroughly discussed in a document to be issued shortly (Ref. 2). It will guide our work on this project and its continuations for the next four years, ending in April 1978. The reader is referred to the plan document itself for a full description of this work.

Our work during this reporting period shows the transition from one domain to another. For the last several months, though, our project work has been devoted exclusively to research problems occasioned by the computer-based consultant system. We have essentially completed the first year of the five-year plan described in Ref. 2.

C. Organization of the Report

Our work during the past reporting period has been organized into five tasks. Task 1 has had as its goal the development of the research plan for the computer-based consultant. Task 2 has had as its goal the development of those problem solving, modeling, and model-access techniques that will be needed by the consultant system. Work on these problems is described in the next section. Task 3 concerns the development of natural language interface capabilities for the new system. This work is described in Section III. In Sections IV and V, we describe the work of Task 4: Vision. Section IV discusses the basic vision research that we are performing and that is necessary to develop specific capabilities for the consultant system. Section V describes the hardware inplementation of a scanning laser rangefinder being used in our vision research. Lastly, Task 5 includes various software and hardware developments and improvements to the AI Center computational facilities. These are discussed in Section VI. Section VII summarizes the publications and presentations by members of this project during the reporting period.

II PROBLEM SOLVING AND MODELING

A. Background

Our work during the last year and a half on problem solving and modeling has undergone a rather extensive evolution. At the beginning of the period, we had just completed our work on the STRIPS system [Refs. 3 and 4] and had begun to extend its capabilities with the ABSTRIPS system [Ref. 5] for generating plans in a hierarchical fashion. We had already concluded that a general-purpose, uniform planning system (such as STRIPS) was too simply organized and lacked the expressive power in which to encode the large quantities of domain-specific knowledge needed by more powerful systems. We had begun the task of designing a more complex system based on the QA4 problem-solving language [Ref. 6] (and its descendent, QLISP [Ref. 7]) when the orientation of the project shifted away from the robot domain to that of the computer-based consultant.

The new domain emphasizes the need for additional features in a planning system, especially functions like hierarchical planning, interaction with a user who is executing the plans, and interaction with a natural-language understanding component.

Our current work stresses the importance of techniques for representing planning information and the need for a multiplicity of local planning subsystems instead of a monolithic top-level planner.

First we shall describe the problem-solving subsystems that have been finished during the period, and then we shall discuss work now in progress. The order of the presentation also serves to illustrate the evolution of our problem-solving work. We will conclude this section with a brief description of QLISP, the language in which we are writing many of our problem solving routines.

B. Completed Projects

1. ABSTRIPS

The ABSTRIPS (Abstraction-Based STRIPS) system is a modification of the STRIPS problem solver.* It uses a representation of a problem domain as a hierarchy of abstraction spaces. Essentially, it works by solving a problem in an abstraction space, a representation of the problem space in which unimportant details are ignored. When a solution to the problem in the abstraction space is discovered, all that remains is to account for the details of the linkup between the steps of the solution. This can be regarded as a sequence of subproblems in the lower space. If they can be solved, a solution to the overall problem will have been achieved. If they cannot be solved, more planning in the abstraction space is required to discover an alternative solution.

*ABSTRIPS is described fully in Ref. 5. This section summarizes the recent work and assumes some familiarity with the STRIPS system (Ref. 3).

This process is applied recursively until a solution is developed in the problem space itself.

By considering details only when a successful plan in a higher level space gives strong evidence of their importance, ABSTRIPS investigates a greatly reduced portion of the search space.

a. Abstraction Spaces in the STRIPS Context

For a practical problem-solving system one would like to have an abstraction space differ from its ground space enough to achieve a significant improvement in problem-solving efficiency, but yet not so much as to make the mapping from abstraction space to ground space complex and time consuming.

For the ABSTRIPS system this criterion is met by having the abstraction spaces differ from their ground spaces only in the level of detail used to specify the preconditions of operators. Although the change in representation provided by this choice may seem intuitively insufficient, it satisfies the criterion well. The world model can remain unchanged; there is no need to delete unimportant details from it because they can simply be ignored. No operators need be deleted in their entirety; if all they do is achieve details, they will never be selected as relevant. Any change to the add or delete lists of the operators would cause the operators' effects to be very different in different spaces. Since the applicability of a particular operator at some intermediate state might depend on any effects of any previously applied operators, the mapping of plans among spaces would be rendered too complex.

Thus, an abstraction space in the ABSTRIPS context differs from its ground space only in the preconditions of its operators. The precondition wffs in an abstraction space will have fewer literals than those in its ground space. The literals omitted will be those that are "details" in the sense that a simple plan can be found to achieve them once the more "critical" literals have been achieved. For instance, consider a PUSHTHRUDR operator, which describes the effects of a robot pushing a particular object through a doorway into an adjacent room. In a high-level abstraction space, the operator would be applicable whenever the object was pushable and a doorway into the desired room existed. In a lower level space, it would also be required that the robot and the object be in the room connected by the doorway with the target room. In a still lower abstraction space, the door would also have to be open. Finally, in the original repesentation of the problem space, the robot would also have to be next to the object and the object would have to be next to the door.

For ABSTRIPS to be able to discriminate among various levels of detail, each literal within the preconditions of each operator in a problem domain is assigned a "criticality" value at the time the domain is first defined. Only the most critical literals will be in the highest abstraction space, whereas in lower spaces less critical ones will also appear.

b. Utilization of Abstraction Spaces in Planning

To take advantage of the hierarchical planning approach offered by the use of abstraction spaces, the ABSTRIPS system--whose flow of control is shown in Figure 1--has a recursive executive program. This program accepts two parameters. The first is a criticality value indicating the abstraction space in which planning is to occur. The second is a list of nodes from the search tree in the higher space, representing a skeleton plan. When a new problem is posed to ABSTRIPS, the external interface program sets the preconditions of a dummy operator to the goal wff. The domain's maximum criticality, which was determined when criticalities were assigned, is retrieved. The executive is called with the criticality set to the maximum and the skeleton consisting of the dummy operator.

Within the highest abstraction space, the executive plans to achieve the preconditions of the dummy step in the skeleton plan, i.e., the main goal. When a plan is found, the executive computes the criticality of the next lowest space in which planning is needed, and it builds a skeleton of nodes along the path of the successful plan. The executive then invokes itself recursively. The invocation solves in turn the subproblems of bridging the gaps between steps in the skeleton plan and of ensuring that the steps in the skeleton plan are still applicable at the appropriate points in the new plan. The final step in the skeleton is always the dummy operator, and so the final applicability check ensures that the original goal has been reached. When all subproblems have been solved, the executive invokes itself for planning in a still lower space. This recursion continues until a complete plan is built up on the problem space itself.

This search strategy might be termed a "length-first" search. It pushes the planning process in each abstraction space all the way to the original goal state before beginning to plan in a lower space.

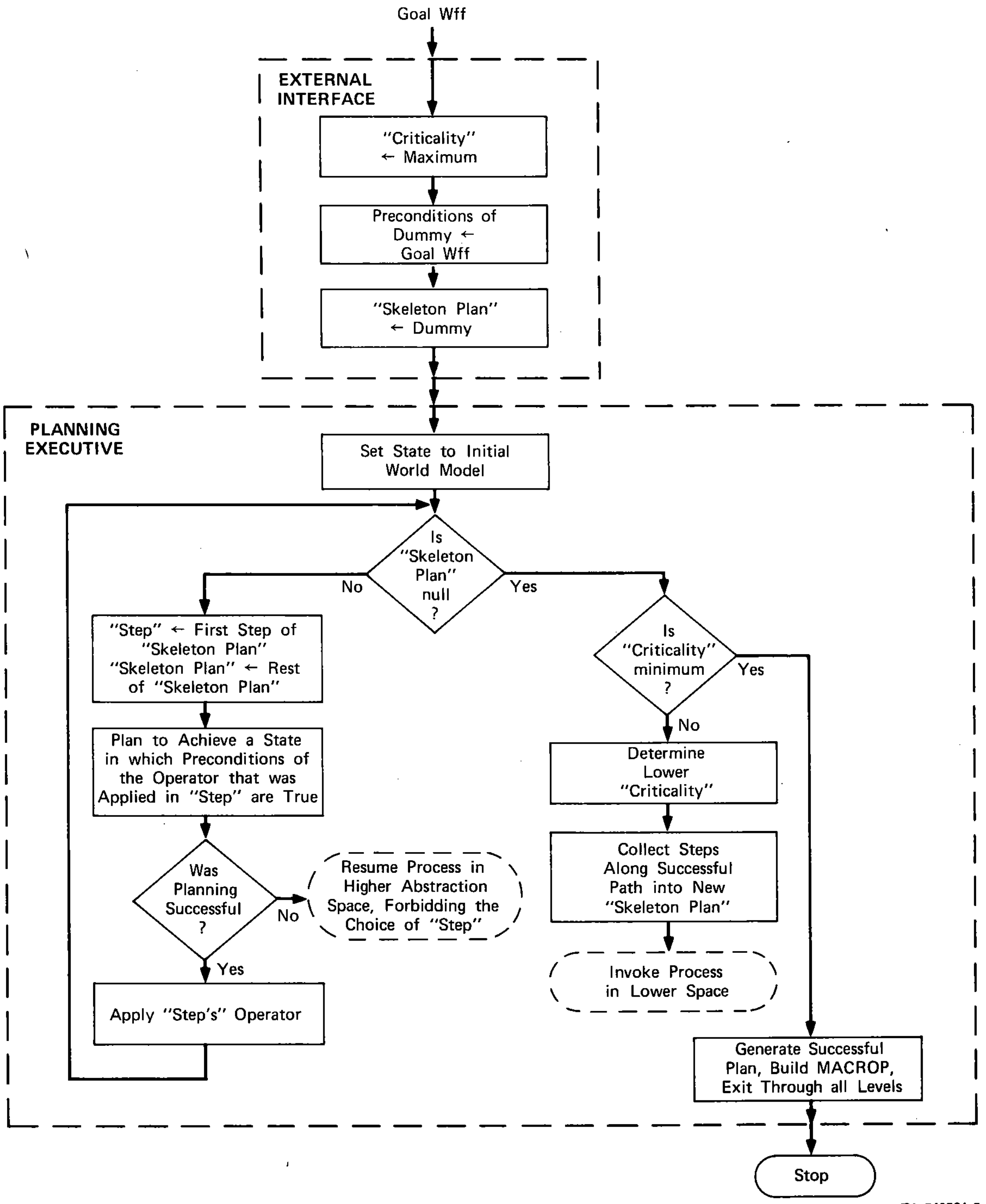

Goal Wff
EXTERNAL INTERFACE
"Criticality" ← Maximum
Preconditions of Dummy ← Goal Wff
"Skeleton Plan" ← Dummy
PLANNING EXECUTIVE
Set State to Initial World Model
Is "Skeleton Plan" null ?
No
Yes
"Step" ← First Step of "Skeleton Plan" "Skeleton Plan" ← Rest of "Skeleton Plan"
Plan to Achieve a State in which Preconditions of the Operator that was Applied in "Step" are True
Was Planning Successful ?
No
Resume Process in Higher Abstraction Space, Forbidding the Choice of "Step"
Yes
Apply "Step's" Operator
Is "Criticality" minimum ?
Yes
No
Determine Lower "Criticality"
Collect Steps Along Successful Path into New "Skeleton Plan"
Invoke Process in Lower Space
Generate Successful Plan, Build MACROP, Exit Through all Levels
Stop
TA-740524-5

FIGURE 1 FLOW OF CONTROL OF ABSTRIPS

This enables the system to recognize as early as possible the steps that would lead to dead ends or very inefficient plans.

If any subproblem in a particular space cannot be solved, control is returned to the process in its abstraction space. The search tree is restored to its state prior to the selection of the node that led to failure in the ground space. That node s ELIMINATED FROM CONSIDERATION, and the search for a successful plan at the higher level continues.

c. Experimental Results

A series of experiments was run to compare the performance of the STRIPS and ABSTRIPS systems.

The definition of the problem domain for these experiments is identical to the one used for the examples in our last annual report (Ref. 1), except for the addition of a few literals to the preconditions of operators. The set of tasks from that report was run on ABSTRIPS. The running times and the search trees are compared with those from the STRIPS system in Table 1. Figure 2 plots the planning time as a function of plan length for STRIPS and ABSTRIPS on an extended set of problems from the robot domain.

2. A Deductive Retrieval System

In our work with robot planning and more recently in our work with an expert consultant system, our programs need to maintain a model of

Table 1

COMPARISON OF PLANNING TIMES AND SEARCH TREES

	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5
ABSTRIPS					
Time to produce the plan (minutes and seconds)	1:54	2:55	2:24	2:30	6:41
Nodes in search trees					
Total	25	34	30	33	63
By spaces*	5,5,5,10	5,7,7,15	3,4,11,12	5,7,7,14	5,17,16,25
Nodes on solution path					
Total	24	32	28	32	54
By spaces*	5,5,5,9	5,7,7,13	3,4,10,11	5,7,7,13	5,11,15,23
Operators in plan	4	6	5	6	11
STRIPS					
Time to produce the plan (minutes and seconds)	1:40	5:44	4:34	9:47	>20:00†
Total nodes in search trees	10	33	22	51	--
Nodes on solution path	9	13	11	15	--
Operators in plan	4	6	5	7	--
STRIPS with MACROPs					
Time to produce the plan (minutes and seconds)	1:40	2:06	5:18	3:00	5:49
Total nodes in search trees	10	9	14	9	14
Nodes on solution path	9	9	9	9	14
Operators in plan	4	6	5	6	11

*The number of nodes from the search tree in each space, from the one of highest criticality to the problem space itself.

†STRIPS had not solved Problem 5 after 20 minutes.

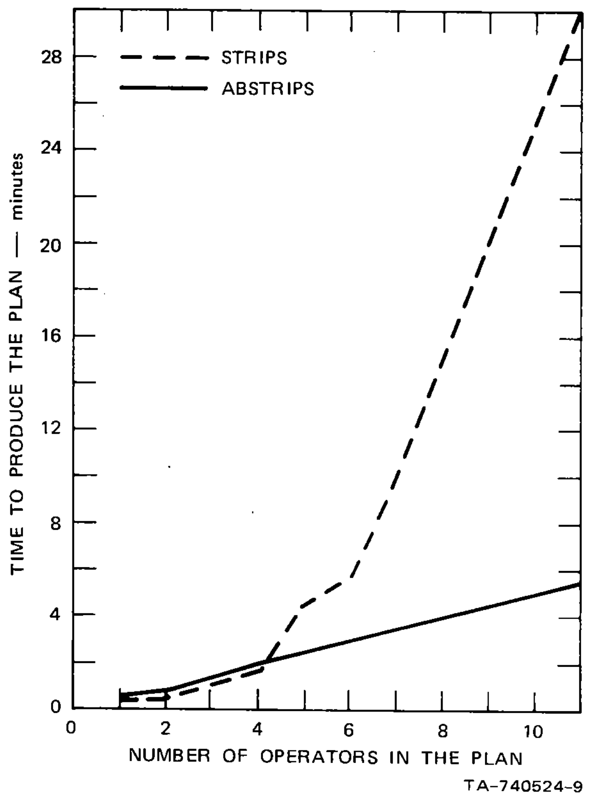


FIGURE 2 PLANNING TIME AS A FUNCTION OF PLAN LENGTH

the physical environment in which the robot activity or maintenance operation is taking place. Our planning, monitoring, and advice giving programs ask questions of this model to determine how they should proceed. Hence, we need a question-answering subsystem that can interface with the model. This subsystem should provide a deductive capability for dealing with questions whose answers are not explicitly stored in the model but whose answers can be deduced from information that is in the model. In addition, we want this subsystem interface to shield the rest of the system from concern about the actual representations used in the model. A new design for such a subsystem was tested during the last year with an implementation in QA4, and a current implementation in QLISP is one of the initial components of the expert consultant system.

The design began as an augmentation of the QA4 model retrieval statements EXISTS and INSTANCES. These statements retrieve from the model one or all true instances of a given expression containing variables (i.e., a pattern). For example, the statement (EXISTS (ON X DESK1)) could retrieve from the model one object that is on DESK1, and the statement (INSTANCES (ON X DESK1)) could retrieve all objects that are on DESK1. Now, consider as another example the statement (EXISTS (TOP:CLEAR DESK1)). This statement will look in the model for the expression (TOP:CLEAR DESK1) to indicate that no objects are on DESK1. If that expression does not have value TRUE in the model, then the query will fail even though a simple deductive process that looked in the model for expressions of the form (ON X DESK1) could determine an answer to the query. The new design attempts to make it easy to add such deductive processes to the system and thereby turn the retrieval mechanism into a question-answering mechanism.

The design allows the user to associate with each predicate in the system (e.g., ON, TOP:CLEAR, IN:ROOM) a list of deductive programs that will be called whenever true instances of an expression containing the predicate are being searched for in the model. Hence, one could associate a deductive program with the predicate TOP:CLEAR that would perform as described in the previous paragraph.

One common use of these deductive programs is to determine that an expression is false based on uniqueness properties of the predicate. For example, if there is a query asking whether OB1 is in room RM1, i.e., (EXISTS (IN:ROOM OB1 RM1)), a deductive program could look in the model for any true expression of the form (IN:ROOM OB1 X). If such a true expression is found and the room name that matched with X is not RM1, then the deduction can be made that OB1 is not in RM1.

Standard deduction programs are included in the system for NOT, AND, and OR. These programs allow one to ask questions such as: "Find a desk in either room RM1 or RM2 whose top is clear." The QA4 form of that query might be (EXISTS (AND (TYPE X DESK) (OR (IN:ROOM X RM1) (IN:ROOM X RM2)) (TOP:CLEAR X))). To answer this query an AND deductive program is first called. It would ask for an object of type DESK from the model. If one is found, say D1, then it asks if (OR (IN:ROOM D1 RM1) (IN:ROOM D1 RM2)) is true. This query calls an OR deductive program which might in turn call an IN:ROOM deductive program that could determine which room D1 is in by asking the model for the location of D1. If D1 is found to be in either room RM1 or RM2. then the AND program will ask if (TOP:CLEAR D1) is true. This might cause a TOP:CLEAR deductive program to be called that would ask the model for any objects X for which (ON X D1) is true; if no such objects are found, then D1 is returned as a desk that satisfies the conditions in the original query.

This question-answering process makes use of the QA4 backtracking capabilities when there is a need for more than one answer to a query. For instance, if our example query had been to find all the desks in either room RM1 or RM2 with clear tops, then each time such a desk was found it would be saved in a set, and the answering process would be backtracked to a point in the deductive programs where alternative choices were available. In this example, control would return to the search for objects of type DESK, and, if another desk is found, the process would proceed as before to check the room the desk is in and whether its top is clear. Each desk would be checked in this manner to form the desired set.

The backtracking mechanism is also used within the question-answering process when, for example, a desk that is found by the (TYPE X DESK) query is not in either RM1 or RM2. In that case the OR deductive program returns FALSE to the AND deductive program and control is backtracked into the (TYPE X DESK) query to find the next desk.

3. A General Tree-Searching Package

A set of programs was implemented in QA4 that provides a conceptual framework and executive program for conducting heuristic tree searches. These programs were motivated by an interest in exploring ways of using features in QA4 such as processes and contexts, and by the need for a general search package in our robot programs. We expect to reimplement this package in QLISP when the Bobrow-Wegbreit stack model [Ref. 8] is implemented in INTERLISP and then incorporate it into the computer-based consultant system.

This search package assumes that there is an initial node of the tree given and that the task is to find a path through the tree from that initial node to a node that satisfies a given goal test. Whenever the search reaches a node in the tree for the first time, an evaluation number is computed for the node. This evaluation number indicates to the search executive the desirability of the node for further consideration. At each step the executive selects the node with the best evaluation number, generates an offspring node of the selected node, computes an evaluation number for the new node, and tests whether the new node satisfies the goal test. If the goal test fails, then node selection occurs again and the process repeats.

This search package can be applied to a particular task domain by providing the programs for node evaluation, goal testing, and offspring generation. For example, it has been used to compute multiroom paths for a robot system, where each node in the tree represents a room to which the robot has traveled, and each offspring node of a given node represents an adjacent room. Hence, if the initial node represents the room in which the robot begins, then a path through the tree represents a room-to-room route that the robot can travel to reach a goal room. For many applications, including the robot path-finder, we wish the search to find the shortest path from the initial node to the goal. In such cases one often uses a node evaluation scheme that computes for a node an estimate of the length of the shortest path from the initial node to a goal-satisfying node passing through the node being evaluated. This evaluation scheme estimates that the node with the smallest evaluation number is on the shortest path to the goal and therefore should be the one considered by the searcher.

By using QA4 to implement this search package we obtained clear advantages for the user with regard to flexibility, power, ease of use. and search efficiency. Each node in the search tree is represented by a QA4 dynamic context. The search executive proceeds by growing a tree of contexts where the offspring of a given node is formed by doing a context push operation on the node; hence, any variable values and model information that were true in the parent node are automatically true in the offspring node.

The executive assumes that each node has associated with it a node-evaluation function, a goal-test function, and an offspring generating function. If these functions are to be the same for each node in the tree, as in the case for the robot path-finding example, then they merely need to be attached to the initial node and all other nodes will "inherit" them automatically (via the context mechanism) from their common ancestor, the initial node. But, if the problem domain is such that nodes in different parts of the tree should be evaluated, generated, and tested in different ways, then the nodes can be assigned their individualized functions as they are generated.

The evaluation, generation, and goal testing functions are called by the search executive in the dynamic context of the node being considered; hence, when these functions access model information or variable values, they are given the values that are current at the node. This design frees the writer of these functions from being burdened with concerns about obtaining the correct set of values for the node and handles much of the bookkeeping required in passing information from parent to offspring node.

When a node is selected by the search executive, the node generation function for that node is used to produce an offspring of the node.

The first time any given node is selected by the executive, its offspring generating function is transformed into a QA4 process. The executive assumes that each time this process is resumed it will form a new offspring and then suspend itself. This implies that the values of local variables in the generating function will be saved between generations so that the function can be written to determine the order in which offspring are generated and thereby significantly affect the amount of searching required to find a solution. For example, the offspring-generating function used in the robot path finder computes the distance from the goal room to an adjacent room before using that adjacent room to create an offspring. If the adjacent room is farther from the goal than the current room, then the room is put on a rejects list and considered again only after the offspring for all the adjacent rooms closer to the goal have been generated. For many problems this computationally inexpensive ordering algorithm virtually eliminates consideration of nodes that are not on a path to a solution.

4. A Control Regimen for Man-Machine Cooperation in Assembly-Disassembly Tasks

The computer-based consultant system entails man-machine cooperation on complex tasks. To achieve such cooperation we think it will be very important to be able to encode localized knowledge in a straightforward fashion. On the other hand, if the system is to be truly useful, it must be able to interact with its user along a wide range of generality, specifying steps of the task at a level no more detailed than the user needs to accomplish that task.

To achieve both of these goals, we need a hierarchically organized computer system whose control regimen allows instructions to the user and the maintenance of an internal model to be specified concurrently. The system must furthermore be very forgiving about missing details in the model, since the model can be only as detailed as the previous interactions with the user.

To explore these ideas, a simple system was written in QLISP /Ref. 7/. The system's knowsedge about the task domain is encoded procedurally, using QLISP augmented by three new statements for interacting with the user and the internal model of the state of the task.

The first of these statements, VGOAL, is like a QLISP GOAL statement, but with an additional argument which is an instruction to the user. The statement works by first checking the data base to see if an expression matching the goal pattern has been asserted. If so, that expression is returned as the value of the VGOAL statement. If not, the instruction is typed to the user. If he responds in a positive fashion, the system assumes that the goal has been achieved. The user is asked to supply values for any unbound variables in the goal pattern or in the instruction. An appropriate expression is constructed from the goal pattern and these values. This expression is asserted in the model and returned as the value of the VGOAL statement.

If the user responds negatively to the instruction, a QLISP GOAL statement is constructed from the VGOAL statement, and it is evaluated to generate more detailed subgoals.

The second statement, VFOR, performs iterations in an interesting manner. It has two arguments: an instruction to the user, and a garden-variety iterative statement of the form: (FOR counter FROM start TO finish DO code). The statement operates by first issuing the instruction to the user. If the user's response is positive, he is asked to supply values for the unbound variables in the instruction. If the user's response is negative, counter is set to start, and the code is evaluated. Then, if start + 1 equals finish, the VFOR statement terminates. Otherwise, a new VFOR statement is constructed that is identical to the old one, except that start is replaced by start + 1. This new statement is then evaluated.

Thus, this statement enables the system to lead a user through the details of one cycle of an interative activity without forcing him to listen to details once he has mastered the activity.

The third statement, VIS, is a forgiving data base retrieval function. It operates like the QLISP IS statement, searching the data base for an expression that matches its pattern. But if no such expression was found, VIS constructs one and asserts it in the data base.

This statement is useful because it allows the model to be maintained at the level of detail of the interaction with the user. When the system's programs request more detailed information, using VIS, they receive an appropriate expression rather than failing.

Using these primitives, a small system was built to guide a very naive user through the fastening of two flat metal plates with four bolts and nuts. The system consists of fifteen functions. Several of the more complex ones are shown in Figure 3. A sample dialogue generated by this system is shown in Figure 4. Only seven of the functions of the system were actually invoked during this conversation. Although the interaction is rough and the domain is small, the system creates a surprising sense of richness from a small amount of simply encoded knowledge.

The concepts that were developed for this system are currently being used in the integrated planning and execution-monitoring system, which is described in Section II.C, "Work in Progress."

5. A Simulation Program for an Air Compressor

a. Introduction

A program has been designed and implemented in QLISP that simulates the behavior of a small air compressor. (See Appendix A for a description of the particular air compressor being considered.) The behavioral features simulated by the program include voltage and current levels in the power cord and motor cable, rotational rate of the pump and motor, binary status of the pressure switch and valves, pressure in the tank and aftercooler, speed of the belt, and so on. When a simulated action by the compressor operator (such as plugging in the power cord or using air from the tank) is entered into the program, a trace is produced indicating the sequence of changes that occur in these behavioral features. Table 2 provides an example of such a trace following the plugging in of the power cord when the pressure in the tank is initially zero. The design of this program was strongly influenced by Hendrix [Ref. 9].

```
(FILECREATED "23-SEP-73 11:56:06" EXAMPLE)


(DEFINEQ


(FASTENPLATE
  (QLAMBDA (FASTENED ←PLATE1
                     ←PLATE2)
           (MATCHQ ←SIZE
                   (CDR (ASSOC (QGET (PLATE $PLATE1)
                                     HOLESIZE)
                               $IZETABLE)))
           (MATCHQQ ←C
                    (CLASS))
           (VFOR (GET A CLASS ←C OF FOUR $SIZE BOLTS)
                 [FOR I FROM 1 TO 4
                    DO (PROGN (LOOKFORBOLT ('(ONHAND ←B)))
                              (MATCHQQ ←C
                                       (CLASS $B $$C)
                 ($C (CLASSOF (BOLT ←B)
                              SIZE $SIZE)))
           (MATCHQQ ←BOLTS
                    $C)
```

FIGURE 3 EXAMPLES OF FUNCTIONS FOR MAN-MACHINE COOPERATION

```
(VFOR (LOOSELY FASTEN ALL THE BOLTS IN $C)
      (FOR I FROM 1 TO 4
         DO (PROGN (MATCHQQ (CLASS ←B
                                   ←←C)
                            $C)
                   (STARTIN ('(BOLT $B/
(MATCHQQ ←C
         $BOLTS)
(VFOR (TIGHTEN THE BOLTS IN $C)
      (FOR I FROM 1 TO 4
         DO (PROGN (MATCHQQ (CLASS ←B
                                   ←←C)
                            $C)
                   (TIGHTEN ('(BOLT $B/
(ASSERT (FASTENED $PLATE1 $PLATE2))))
```

↑L

FIGURE 3   EXAMPLES OF FUNCTIONS FOR MAN-MACHINE COOPERATION (Continued)

```
(TIGHTEN
  (QLAMBDA (BOLT ←BOLT)
           (VIS (STARTEDIN ←NUT
                              $BOLT))
           (VGET (BOLT $BOLT)
                 SIZE ←SIZE)
           (VGOAL (CONTINUE SCREWING THE NUT ONTO BOLT $BOLT
                     UNTIL RESISTANCE IS FELT.)
                  (HANDTIGHTENED $NUT $BOLT)
                  APPLY NIL)
           (VGOAL (FIND A SCREWDRIVER.)
                  (ONHAND ←SCREWDRIVER)
                  APPLY
                  (LOOKFORSCREWDRIVER)
                  TYPE SCREWDRIVER)
           (VGOAL (FIND A $SIZE WRENCH)
                  (ONHAND ←WRENCH)
                  APPLY
                  (LOOKFORWRENCH USEANGLEWRENCH)
                  TYPE WRENCH)
           (VGOAL (USE WRENCH $WRENCH TO HOLD THE NUT FIRMLY.)
                  (HOLDING $WRENCH $NUT)
                  APPLY
                  (USEWRENCH))
           (VGOAL (SCREW THE BOLT IN FAIRLY TIGHTLY)
                  (TIGHTENED $NUT $BOLT)
                  APPLY
                  (SCREWIN))))
```

FIGURE 3 EXAMPLES OF FUNCTIONS FOR MAN-MACHINE COOPERATION (Continued)

```
(SCREWIN
  (QLAMBDA (TIGHTENED ←NUT
                      ←BOLT)
           (IS (ONHAND ←SC)
               TYPE SCREWDRIVER)
           (VGOAL (INSERT THE END
                     OF THE BLADE OF SCREWDRIVER $SCREWDRIVER INTO THE
                                          NOTCH
                     IN BOLT $BOLT %.)
                  (INSERTED $SCREWDRIVER $BOLT)
                  APPLY
                  (USESMALLERSCREWDRIVER))
           (VGOAL (TURN THE SCREWDRIVER CLOCKWISE
                     UNTIL FIRM RESISTANCE IS FELT.)
                  (TIGHTENED $NUT $BOLT)
                  APPLY NIL)))
)
  (LISPXPRINT (QUOTE EXAMPLEFNS)
              T)
  (RPAQQ EXAMPLEFNS (FASTENPLATE TIGHTEN SCREWIN))
STOP
↑L
```

FIGURE 3 EXAMPLES OF FUNCTIONS FOR MAN-MACHINE COOPERATION (Concluded)

```
>(FASTEN TOPPLATE TO BOTTOMPLATE)
<HOW?
>(GET A CLASS ←C OF FOUR #10 BOLTS)
< OK
>PLEASE SUPPLY BINDINGS:

< ←C:  (CLASS B1 B2 B3 B4)
>(LOOSELY FASTEN ALL THE BOLTS IN (CLASS B1 B2 B3 B4))
< CAN'T
>(FIND A NUT ←NUT)
< CHECK
>PLEASE SUPPLY BINDINGS:

< ←NUT:  N1
>(PUT BOLT B1 THROUGH A HOLE IN THE UPPER AND LOWER PLATES)
< HOW
>(PUT BOLT B1 THROUGH SOME HOLE IN THE UPPER PLATE.  THEN ALIGN THE
UPPER AND LOWER PLATES SO THAT THE BOLT CAN BE PUSHED THROUGH A HOLE
IN THE LOWER PLATE.)
< HOW?
>(IF YOU CAN'T ACHIEVE (INHOLE B1) BY NOW, I CAN'T HELP YOU.)
< OK
>(SCREW NUT N1 LOOSELY ONTO BOLT B1)
< YES
>(LOOSELY FASTEN ALL THE BOLTS IN (CLASS B2 B3 B4))
< OK
```

FIGURE 4 A SAMPLE DIALOGUE

```
>(TIGHTEN THE BOLTS IN (CLASS B1 B2 B3 B4))
< CAN'T
>(CONTINUE SCREWING THE NUT ONTO BOLT B1 UNTIL RESISTANCE IS FELT.)
< OK
>(FIND A SCREWDRIVER.)
< HOW?
>(TELL THE VISION PEOPLE TO FIND YOU A SCREWDRIVER.)
< OK
>PLEASE SUPPLY BINDINGS:

< ←SCREWDRIVER:  SC1
>(FIND A #10 WRENCH)
< OK
>PLEASE SUPPLY BINDINGS:

< ←WRENCH:  WR1
>(USE WRENCH WR1 TO HOLD THE NUT FIRMLY.)
< OK
>(SCREW THE BOLT IN FAIRLY TIGHTLY)
< HOW?
>(INSERT THE END OF THE BLADE OF SCREWDRIVER SC1 INTO THE NOTCH IN
BOLT B1 %.)
< CAN'T
>(FIND A SCREWDRIVER WITH A BLADE SMALLER THAN THAT OF SC1, WHICH
WILL FIT INTO THE NOTCH OF BOLT B1 %.)
< OK
>PLEASE SUPPLY BINDINGS:
```

FIGURE 4 A SAMPLE DIALOGUE (Continued)

```
>(TURN THE SCREWDRIVER CLOCKWISE UNTIL FIRM RESISTANCE IS FELT.)

< OK

>(TIGHTEN THE BOLTS IN (CLASS B2 B3 B4))

< HOW?

>(CONTINUE SCREWING THE NUT ONTO BOLT B2 UNTIL RESISTANCE IS FELT.)

< OK

>(USE WRENCH WR1 TO HOLD THE NUT FIRMLY.)

< OK


>(SCREW THE BOLT IN FAIRLY TIGHTLY)

< OK

>(TIGHTEN THE BOLTS IN (CLASS B3 B4))

< OK


↑L
```

FIGURE 4 A SAMPLE DIALOGUE (Concluded)

Table 2

TRACE OF SIMULATION OF THE COMPRESSOR

```
Initial State:  tank pressure 0 and power cord unplugged.
Simulation begins by scheduling an event that asserts
(STATUS POWER:CORD PLUGGED:IN).

    SIM TIME IS 0 [sec]

    ASSERT (STATUS POWER:CORD PLUGGED:IN)
    ASSERT (VOLTAGE POWER:CORD 110)
    ASSERT (VOLTAGE MOTOR:CABLE 110)
    ASSERT (VOLTAGE MOTOR 110)
    ASSERT (CURRENT MOTOR 9) [amps]
    ASSERT (CURRENT MOTOR:CABLE 9) [amps]
    ASSERT (CURRENT OUTLET 9) [amps]
    ASSERT (CURRENT POWER:CORD 9) [amps]
    ASSERT (RPM MOTOR 1725)
    ASSERT (SPEED BELT 37.63363) [ft/sec]
    ASSERT (RPM PUMP 802.3256)
    ASSERT (STATUS CHECK:VALVE OPEN)

    SET SIM TIME TO 430.5 [sec]

    ASSERT (STATUS PRESSURE:SWITCH OPEN)
    ASSERT (VOLTAGE MOTOR:CABLE 0)
    ASSERT (VOLTAGE MOTOR 0)
    ASSERT (CURRENT MOTOR 0) [amps]
    ASSERT (CURRENT MOTOR:CABLE 0) [amps]
    ASSERT (CURRENT OUTLET 0) [amps]
    ASSERT (CURRENT POWER:CORD 0) [amps]
    ASSERT (RPM MOTOR 0)
    ASSERT (SPEED BELT 0.0) [ft/sec]
    ASSERT (RPM PUMP 0.0)
    ASSERT (STATUS CHECK:VALVE CLOSED)
    ASSERT (PRESSURE AFTER:COOLER 125.0) [psi]
    ASSERT (PRESSURE TANK 125.0) [psi]

    SET SIM TIME TO 440.5 [sec]

    ASSERT (PRESSURE AFTER:COOLER 0) [psi]
```

This simulation program was developed to be used by other parts of the computer-based consultant system, including those concerned with planning, question answering, and trouble shooting. It is useful primarily because it embodies the cause-and-effect relationships that determine the state changes and behavioral features of the compressor while it is operating.

In this section we will provide an overview of the design of the simulation program and then discuss the ways in which we anticipate that other modules of the system might use it.

b. Structural Description of the Simulation Program

The top-level function in the simulation program is a monitor that maintains a simulated clock and initiates events that have been scheduled to occur during the simulation. Each event on the monitor's scheduling queue has associated with it the time at which it is to be initiated and a function whose evaluation will cause the event to be simulated. The monitor proceeds by simply finding the scheduled event that is to occur next, setting the simulated clock to the time at which the event is to be initiated, calling the function that simulates the event and, when control is returned to the monitor, repeating the process with the next scheduled event. Hence, the simulation is said to be "event driven" and the simulated time proceeds in arbitrary sized steps depending on when events are scheduled. Examples of events include those created outside the simulation program, such as a command that the power cord be plugged in, and those created by functions inside the simulation program, such as the opening of the pressure switch scheduled for the time that the pressure in the tank is predicted to reach 125 psi.

The simulation program maintains a model of the current simulated state of the compressor, and the primary effect of an event is to update that model to reflect the simulated state of the compressor after the event has occurred. This updating process is driven by a set of "demons" that are activated when a change is made in the model and typically have the effect of initiating further changes in the model. For example, there is a demon that is activated whenever the status (i.e., plugged or unplugged) of the power cord changes. This demon responds to the new status by changing the power cord's voltage level in the model.

The function associated with an event will typically make a small number (often 1) of changes in the model and those changes will initiate activation of a succession of demon functions. All the model changes made by the demons are considered to be part of the event and to have occurred at the same point in simulated time as the event. Hence, for example, if the power cord is unplugged while the motor is running, the voltage and current changes and the stopping of the motor and pump are all modeled as having occurred instantaneously. An event function or demon can indicate that a model change should occur at some later point in simulated time by scheduling an event to be initiated at that time.

The simulation mechanism we have described up to this point is capable of modeling discrete events. To handle state changes that are continuous over time our program utilizes the deductive retrieval system described earlier. This retrieval system is used by the simulation program for all accesses to the model. So, for example, if a function needs to know the tank pressure, a statement of the form (DEDUCE (PRESSURE TANK ←P)) would be executed. If the tank pressure is not stored explicitly in the model in the form given, then a set of deductive functions associated with the predicate PRESSURE would be executed in an attempt to deduce the tank pressure value. This retrieval mechanism can be exploited to represent continuously changing model values by removing any explicit mention of the value in the model and adding a deductive function that can compute the value as a function of the simulated time. For example, when the compressor pump is not running and no air is being used from the tank, the tank pressure is fixed and can therefore be represented by an explicit model entry. However, when the pump is running or when air is being used from the tank, the tank pressure varies as a function of time and is represented by removing the explicit tank pressure entry from the model and adding a function to the predicate PRESSURE that can compute the tank pressure value whenever it is needed. The adding and deleting of the explicit model entry and the deductive function are done by demons that are activated when the pump changes speed or when the tank valve is opened or closed.

We have been describing the simulation of what might be called the "top-level behavior" of the compressor. We are also interested in simulating the internal workings of components such as the motor, pump, pressure switch, check valve, and the like. These simulations will work with a more detailed model and with smaller increments of time than does the top level simulation, but they should interact smoothly with each other and with the top level simulation. We are currently designing and implementing an interface for this hierarchy of simulations using a simulation of the internal workings of the pump as a test case. We expect a user of this hierarchical system to be able to specify the level of detail at which each component is to be simulated, and that this specification can be dynamically changed as the simulation proceeds.

c. Use of the Simulator in the Computer-Based Consultant System

The design and implementation of the simulation program was motivated by the hypothesis that such a program could act as the information source for many of the system's activities. Although it is too early in the development of the consultant system for this integration to occur, we can indicate how we expect the simulation program to be used.

First of all, the simulation program can interact directly with the human apprentice to provide a description of the behavior of a device (e.g., the compressor). Such an interaction might occur while the apprentice is learning how to operate the device or during a checkout procedure where a comparison would be made between actual and predicted behavior.

The simulator could also be an information source for the question-answering section of the system. Each of the demons in the simulation program expresses a functional characteristic of one of the components of the device. For example, in the compressor simulation the demon that changes the rotational rate of the motor whenever the voltage level changes in the motor cable is representing a functional characteristic of the motor. If each device component has associated with it the demons that represent it, then descriptions of these demons can serve as answers to queries of the form "Tell me about the motor," or "What does the motor do?" Descriptions of demons could be generated from comments in the demon programs or perhaps directly from the programs themselves. The simulator can also be used to answer questions such as "What would happen if the line voltage dropped to 100 volts?" or "Does the compressor have enough power to drive this air tool?"

The simulation program will be an integral part of plan generation and execution monitoring. During plan generation, simulation of "operators" is typically done to determine the anticipated effects of candidate plans. If we allow our plans to contain steps such as "Plug in the power cord" or "Open the tank valve," then the simultion program will be needed to model their results. Hence, the simulation program can be thought of as simply a more elaborate version of the simulations normally done by a planning program. Similarly, during plan execution the systen must maintain a model of the current situation so that it can monitor progress, answer questions, and recognize failures. This model maintenance is accomplished primarily by simulating the operations that the apprentice is performing.

The simulator may also be useful during trouble shooting operations, since a trace of its actions can be used to construct a behavioral tree such as that shown in Figure 5. Each path through this tree represents a cause-and-effect chain in the behavior of the compressor. For example, voltage at the motor causes the motor to

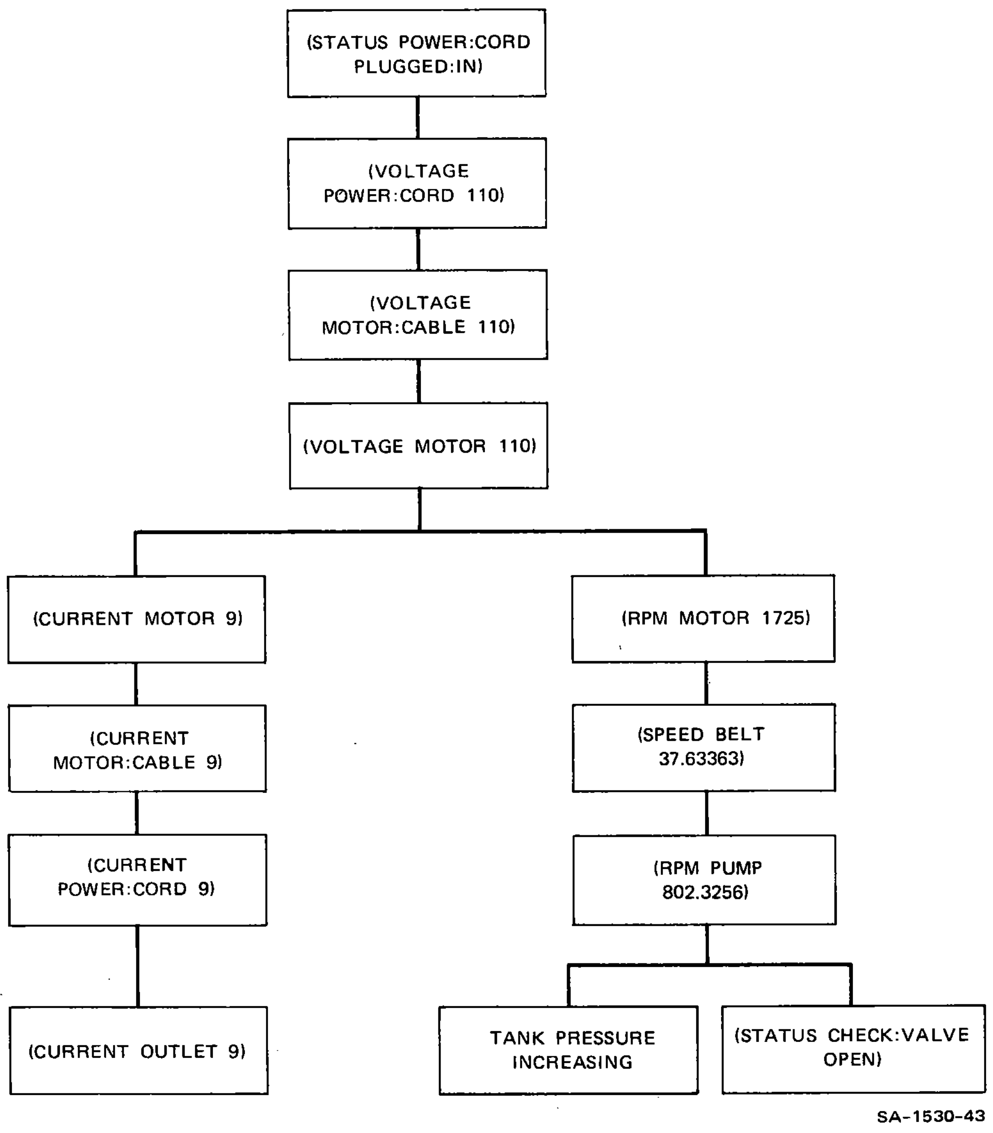


FIGURE 5 BEHAVIOR TREE PRODUCED BY THE SIMULATOR

begin turning, which causes the belt to begin moving, and so on. A "signal tracing" troubleshooting algorithm could use this tree to generate a sequence of questions for the apprentice with the desired result of finding where breaks occur in the anticipated behavioral paths. For example, when the compressor's power cord is plugged in, the system could ask if the pressure in the tank is increasing.

If the answer is no, then another question could be generated along the path in the tree that connects these two events; perhaps, "is the motor turning?" The answer to this question could be generated to test it. Finally, a single faulty link would be identified and replacement of the device component associated with that link could be indicated as a repair procedure.

C. Work in Progress

The major effort now under way is the design and implementation of a planning system that can generate instruction sequences for converting a device from one arbitrary state of assembly into another state. The system has two major aspects, the plan construction operation itself and the representation of the planning steps and associated information. We have been pursuing these two aspects somewhat separately and will thus be discussing them as individual pieces of work. We are just beginning to merge them into an integrated planning system.

1. Plan Construction

Two of the basic problems being considered in this effort are (1) the design of a model that will contain the information needed by a planner, and (2) the design of an effective planning program that will allow us to input easily a human expert's knowledge about assembling and disassembling a device.

One thrust of the model design effort is to define a set of predicates that can be used in an intuitive manner to build a model for the planner. A preliminary set of such definitions for modelling the connections between the "top-level" components of the compressor is shown in Table 3. Using these definitions, we can trace the relationship between two components, say the pump and the platform, during an assembly operation as follows: (REMOVED pump platform), meaning the pump is not on the platform; (POSITIONED pump platform), meaning the pump has been positioned on the platform ready for insertion of the mounting bolts; (CONNECTED pump platform), meaning the mounting bolts have been started but not tightened; and (RIGIDLY:CONNECTED pump platform), meaning the mounting bolts have been tightened. In addition, it is useful to think of the RIGIDLY:CONNECTED and CONNECTED predicates as forming a graph as shown in Figure 6. The task of removing some component X (perhaps as a preliminary to replacing X), can be expressed as the goal of breaking each of the arcs in this connection graph that link to X. For example, we see from Figure 6 that removing the pump entails disconnecting it from the platform, aftercooler elbow, pump pulley, pump brace, and oil drain nipple.

Progress is also being made on an attempt to model the inside of the compressor's pump (see Figure A-3 of Appendix A) so that assembly and disassembly sequences can be planned at that level also. The relationship between components is much more complex inside the pump than for the top level of the compressor we were discussing above. We expect our model to reflect this increased complexity by containing such information as orientation of components and degrees of freedom allowed by connections.

Table 3

MODELING PREDICATES

(CONNECTED B C)	Components B and C are fastened to each other.
(LINKED B C)	There is a chain of connections linking components B and C.
(RIGIDLY:CONNECTED B C)	Components B and C have inflexible shapes and are fastened to each other in such a manner as to form an inflexibly shaped subassembly.
(RIGIDLY:LINKED B C)	There is a chain of rigid connections linking components B and C.
(POSITIONED B C)	Components B and C are in position to be fastened together.
(REMOVED B C)	Components B and C have been disconnected from each other, they are not rigidly linked to each other, and the rigid subassembly of which B is a part and the rigid subassembly of which C is a part are free of each other except for possible nonrigid connections between them.
(INSIDE B C)	C is an identifiable portion of the device that surrounds component B.

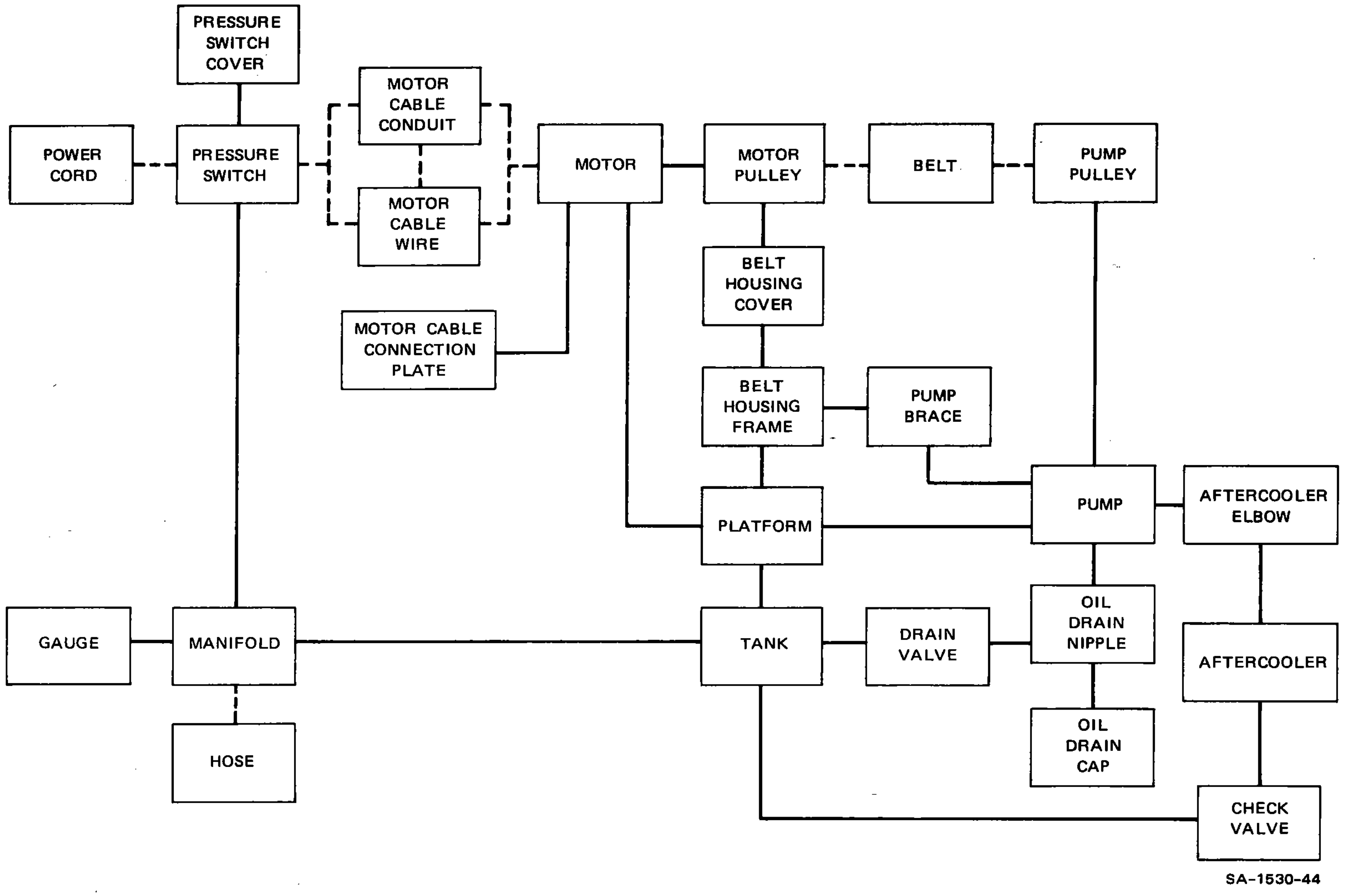


FIGURE 6 DIAGRAM OF COMPRESSOR COMPONENTS

Our work on the design and implementation of an assembly-disassembly planner is concentrating on mechanisms that will allow us to encode easily semantic knowledge about a device as given to us by a human expert. For example, the idea of a planning operator (as used in STRIPS) [Ref. 3] composed of a precondition formula, a list of additions, and a list of deletions, has been generalized to an arbitrary QLISP program that serves the function of a subplanner. Hence, the "operator" for unlinking the motor from the pressure switch is a program that searches the connection graph to find paths between the motor and the pressure switch and calls the planner recursively to plan a set of disconnections that will break all the paths. The use of such subplanners allows us to include in them strategy information, ordering preferences, and other information typically supplied by an expert. Also, it seems that most of the search and selection heuristics typically found in planning executive programs will be contained in two subplanners that are the "operators" for disjunctions and for conjunctions.

An important basic design decision regarding the planner is to structure its planning activity in a hierarchical manner (as in ASTRIPS, described earlier). Hence our system will create "top level" plans with steps of the form (POSITION pump platform) or (DISCONNECT motor platform). These steps, in turn, will be expanded into more detailed plans that deal with loosening bolts and unscrewing screws. By structuring the planning efforts in this manner, we expect to reduce drastically the complexity of each planning task and therefore REDUCE THE NEED FOR SOPHISTICATED SEARCH TECHNIQUES.

2. The Procedural Net: A Representation for Reasoning about Actions

For a number of years there has been considerable controversy in the artificial intelligence community over the way in which knowledge should be represented within a computer memory. One group argued the virtues of a declarative representation; for instance, a collection of predicate calculus wffs in a homogeneous data base (Green, Ref. 10) or a set of relational assertions in a structured data base (Quillian Ref. 11). The declarative representation allows all the knowledge in the system to be processed in a uniform way; but, it requires that the knowledge be processed by some rather general data manipulator which will either be simple and therefore lack goal-directedness, or else be complex and therefore hard to update and improve.

On the other hand, proponents of a procedural representation (e.g., Winograd, Ref. 12) point out that procedures allow efficient goal-directed processing, and are easy to update. However, it is difficult for a procedurally based knowledge system to characterize what knowledge is available to it. Furthermore, it is difficult to leave an appropriate record of the invocations of the procedures that represent the system's knowledge. A knowledge system that is engaged in a mixed-initiative interaction must know what it knows and know what it is doing if it is to have us believe that it "understands."

To encode the knowledge of our computer consultant, we are taking an approach that will give us, we believe, the benefits of both approaches to the representation issue. We call our representation a "Procedural Net." It is a tightly structured network of nodes, each of which may contain both procedural and declarative information.

a. The Structure of a Procedural Net

Each node in a procedural net represents an action at some level of detail. By an action, we mean any operation that affects the system which the net is modeling. (For the purposes of the computer consultant, this means an action taken by the user or by a functioning electromechanical device.) The nodes are linked together to form hierarchical descriptions of operations, and to form plans of action. Table (4) displays the fields of a node as they are currently formulated.

A plan may be defined as a sequence of successor nodes. A procedural net may contain several plans at different levels of detail.

The procedural net may be altered in at least three ways. The first of these is the simulation of the most detailed plan in the net. This is done by evaluating the body of code associated with each node in the plan. The result of the evaluation of each node will be the generation of offspring nodes.

Table 4

THE FIELDS OF A NODE IN THE PROCEDURAL NET

TYPE	May be GOAL, BUILD (up a class of objects), BREAK (down a class of objects), IS (for data retrieval), QLAMBDA (for an arbitrary chunk of code)
QUERY	Expression to be typed to the apprentice, requesting him to perform the action that this node is modeling
PATTERN	A template expression that describes to the system the basic effect of the action this node is simulating
ARGUMENT	The particular expression that caused the activation of this node
BODY	The code that may be evaluated to generate a more detailed plan to achieve the effects of the action modeled by this node.
CONTEXT	The data context in which this node is to be evaluated
EFFECTS	A list of expressions that are added to the data base by this action
CHILDREN	A list of nodes in the net that represent a more detailed simulation of the action in question
PARENT	A pointer to the node that has this node as a child (This node is part of a more detailed simulation of the action modeled by the PARENT node)
PREDECESSOR	The node modeling the preceding action in the plan
SUCCESSOR	The node modeling the succeeding action in the plan
EXECUTED	A flag indicating whether the action modeled by this node was executed

The second way the net may be altered is by criticism of the most detailed plan in the net. Criticism is accomplished by analyzing the precondition and goal structure of the net and attempting to optimize the most detailed plan by altering the predecessor and successor pointers. The critics will have much the same effect as the critics in Sussman's HACKER program (Ref. 13), but we expect that the declarative aspects of the procedural net, together with the explicit representation of goal-subgoal and goal-precondition relationships, will enable our critics to operate more effectively and more efficiently.

The third kind of net alteration is by execution of the actions that the net represents. In the computer consultant domain, execution is the process of walking an apprentice through the steps of a plan. The level of detail at which a step in the plan is presented may be easily varied by dropping down to the step's offspring or rising to the level of its parent. Thus, it should be possible to present the plan to the apprentice at a varying level of detail, based on a model of his competence at individual tasks, and on how well he is progressing on the task at hand. As steps are executed, their modeled effects are asserted in a context representing "reality," and the steps of the plan are checked off.

b. Growing the Procedural Net

The algorithm for generating a procedural net is simple. The top node in the net represents a global goal; e.g., (REPLACED PUMP1 PUMP2). The node will also represent a one-step plan to achieve the goal when no details are considered. Starting from this one-step plan, then, the net is expanded by the following algorithm:

1. Simulate each step in the most detailed plan. This will generate a new, more detailed plan. Each step of the new plan will have some node in the old plan as its parent.

2. Criticize the new plan. This may cause the new plan to be altered or reordered.

3. Determine (based on some model of the difficulty of the task and the competence of the user) whether more planning is necessary. (We will need to expand the net at least several levels beyond our model of the user's capabilities so that we can warn him of any special problems he might encounter.) If more planning is needed, return to step 1. Otherwise, go into "execution mode" and begin talking the apprentice through the task.

c. Advantages of the Procedural Net Representation

The procedural net will constitute a common representation for both the planning and execution-monitoring phases of the consultation task. This will allow for more effective response to failure during execution, since what amounts to a trace of the planner's activities will still be around for analysis. Furthermore, monitoring of serendipitous achievement of aspects of the task at hand will be made simpler by the explicit representation of goal-subgoal relationships. Finally, the use of a common representation for planning and execution allows for effective use of information-gathering operations during planning.

The explicit structuring of goal-subgoal and precondition-goal relationships within the net will provide a facility by which the speech-understanding portions of the computer consultant system will be able to better predict the course of discourse about the ongoing task.

The use of critics will allow the system to deal with problems arising from interactions between steps in the plan. By relying on the critics to do this work, we can encode the knowledge of our task domain by focussing on one local area of knowledge at a time. The system can be built up from a collection of microconsultants which are encoded modularly. The system will, by use of the critic approach, ensure that the microconsultants are working in harmony.

The hierarchical structure of the procedural net will allow the system to model and to plan with conditional operators and operators with loops. It will allow the system to model the progress of the ongoing task at the level of the interaction with the apprentice. In addition, the hierarchically structured declarative knowledge stored in the net will allow the system to answer questions about its reasons and motivations for suggesting particular courses of actions. The system will be, in a real sense, self-aware.

D. QLISP: A Language for the Interactive Development of Complex System

Since January, 1973, we have been engaged in the development of a high-level programming language called QLISP. This work has been carried out with the support of NASA (Contract NASW-2086). The QLISP language provides most of the features of QA4 [Ref. 6] embedded within the INTERLISP system. Thus, QLISP provides a rich variety of data types, a data base for content-directed retrieval of expressions, a powerful pattern matching capability, pattern directed function invocation, and a mechanism for manipulating data contexts. These are all available in a programming environment that provides a versatile, LISP-oriented editor, an easy-to-use file package for maintaining symbolic files, a "programmer's assistant" that allows commands to be undone or altered, and an error correction system that can fix many simple user errors automatically.

We have used QLISP for some of the work already described in this report, and we intend to use it as our basic programming language for the problem-solving portions of the computer consultant.

During the period covered by this report, the basic features of the language were implemented and incorporated in a rather reliable and easy-to-use package. Programs written in QLISP run about 15 to 30 times faster than the corresponding QA4 programs. During the coming year, we plan to install a new pattern matcher that can perform unification (matching two patterns, both of which contain variables) as well as do simple matches more quickly. We will be developing a compiler for QLISP, which will speed up programs considerably. Finally, when the Bobrow-Wegbreit stack model [Ref. 6] is implemented in INTERLISP, we will incorporate facilities for pseudo-parallel processing into QLISP.

QLISP is available over the ARPANET, and has already been used on an experimental basis at several sites around the network.

III NATURAL LANGUAGE INTERFACE

A. Dialogs

We have been taping dialogs between two people working to complete repair tasks in the workstation environment. The device we have used for our experiments is the small air compressor described in Appendix A. One person plays the role of an expert adviser; the other acts as an apprentice.

Our initial experiments were done with the expert and the apprentice in the same room. In some of the experiments the expert and the apprentice could see each other; in other experiments they could not. A major characteristic of these dialogs was a cooperative completion of ideas: one of the dialog participants would start a sentence, and the other would complete it. That is, as soon as the listener thought he knew what the speaker was trying to say, he would indicate this by completing the thought. Looking over these dialogs we discovered that sentence fragments were at least as common as complete sentences. Communication is seen as a subpart of the whole task -- namely, cooperating to get the task done.

For a speech system to be able to understand speech like this, it would have to have an extremely strong semantic component. Before trying to build such a system, we decided to run a series of experiments to see how communication would be affected if the expert and the apprentice were not allowed to interrupt each other.

A second observation from our first set of experiments was that the amount of vision available has a large effect on the language used. When the two participants can see each other, there is a much larger amount of deictic (i.e., pointing) reference. Since our system will only have limited vision capabilities, we also designed the second set of experiments so that visual information was restricted. Thus, this second set of experiments serves as a closer simulation of the system for which we are building a speech component.

The design of the experiment is shown in Figure 7. The apprentice, the expert, and a monitor who served as a link between them were all in separate rooms. The apprentice and the monitor were connected by microphone and earphone links. The expert and the monitor were linked

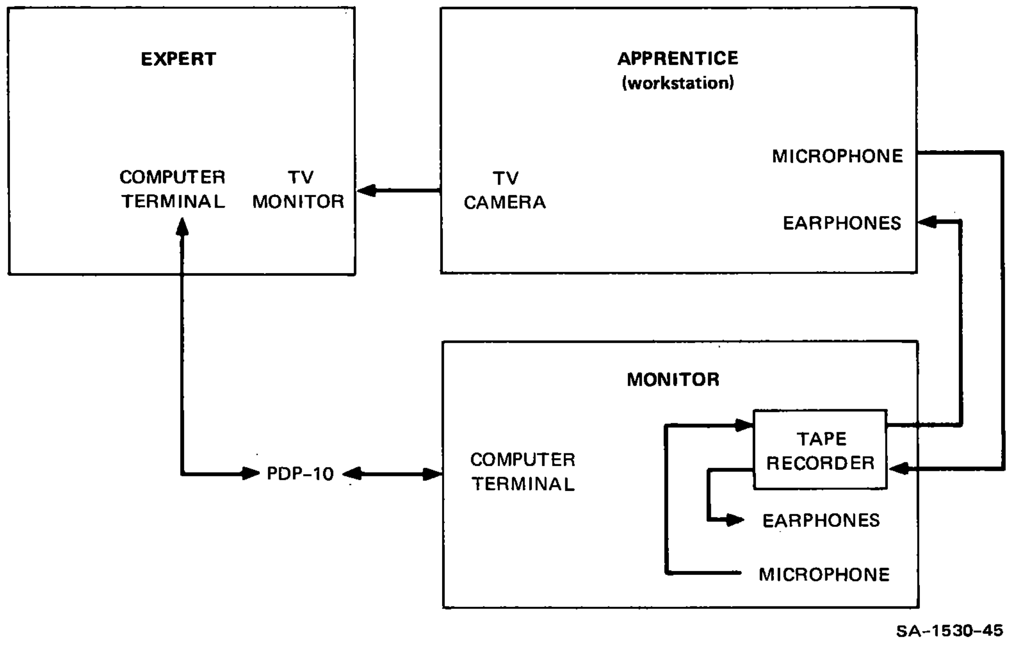


FIGURE 7 CONFIGURATION FOR DIALOG EXPERIMENTS

through computer connections to a PDP-10. The monitor typed what the apprentice said to the expert and read what the expert typed to the apprentice. In addition, the monitor was responsible for seeing that the apprentice did not speak while the expert was typing. Since only one person could type at a time (a feature of the linking program), the expert could not interrupt the apprentice. The whole dialog was taped and a typescript file was kept. When the expert wanted to see something he had to request that a TV picture be taken and transmitted over the television channel; only still shots were allowed. A camera operator was used so the apprentice would not be disturbed when the expert asked for a picture of something. This also allowed the apprentice to point at parts and tools when he wanted to identify them for the expert.

Placing a monitor between the expert and the apprentice had the effect of slowing down the dialog. However the response time did not seem to be much greater than what we can expect in the next few years from a speech understanding system. The main effect of this slowdown seemed to be that the apprentice would go ahead and try things while waiting for a response. Aside from the slowdown caused by messages being typed, the participants did not seem at all hampered by this means of communication. In fact, what appears to have happened is that the listener would act on his understanding (before an utterance was completed) rather than finishing the utterance (i.e., the speech act) and then acting.

It would be reasonable to expect that the speech input from someone taking directions would be very limited, consisting mostly of: "yes", "no", and "I don't understand." Our experience indicates this is not the case. The apprentice often takes the initiative in a task. He may need to explain a problem, or ask a question about an instruction he has just been given; he may want to propose the next step, or report on what he has been doing. Answers to questions are often far more complicated than a simple yes or no. The amount and kind of speech vary with the level of expertise of the apprentice. The dialog with a naive apprentice is filled with definitions of terms (establishing a common vocabulary) and questions about how certain operations should be done. With an experienced apprentice there is less talk in general: the dialog consists mostly of reports of what has been done or explanations of what the apprentice intends to do next.

Figure 8 is part of a dialog with a naive apprentice who was replacing the pump and the belt. Figure 9 shows part of a dialog with an experienced apprentice (i.e., someone who had experience working with mechanical devices -- he had not worked with the air compressor before) whose task was to assemble the compressor.

We are continuing these experiments using different experts and different apprentices. Different experts supply us with ideas about alternate ways of working on a task and different terminology. Different apprentices with varying levels of skill and experience give us some data on the adaptability which will be required of the system. We are also getting data on how the language, planning, and vision requirements vary with the skill level of the apprentice.

```
E:NOW REMOVE THE PUMP

E:WHAT ARE YOU GOING TO DO FIRST

A:HOW DO I REMOVE THE PUMP

E:FIRST YOU HAVE TO REMOVE THE FLYWHEEL

A:  HOW DO I REMOVE THE FLYWHEEL

E:FIRST, LOOSEN THE 2 ALLEN HEAD SETSCREWS HOLDING IT TO THE SHAFT,
THEN PULL IT OFF

A:OK

A:I CAN ONLY FIND ONE SCREW   WHERES THE OTHER ONE

E:ON THE HUB OF THE FLYWHEEL

A:THATS THE ONE I FOUND WHERES THE OTHER ONE

E:ABOUT 90 DEGREES AROUND THE HUB FROM THE FIRST ONE

A:I DONT UNDERSTAND  I CAN ONLY FIND ONE

A:OH, WAIT YES I THINK I WAS ON THE WRONG WHEEL

E:SHOW ME WHAT YOU ARE DOING

A:I WAS ON THE WRONG WHEEL AND I CAN FIND THEM BOTH NOW

A:THE TOOL I HAVE IS AWKWARD  IS THERE ANOTHER TOOL THAT I COULD
USE INSTEAD

E:SHOW ME THE TOOL YOU ARE USING

A:OK

E:ARE YOU SURE YOU ARE USING THE RIGHT SIZE KEY

A:I'LL TRY SOME OTHERS

A:I FOUND AN ANGLE I CAN GET AT IT

A:THE TWO SCREWS ARE LOOSE BUT IM HAVING TROUBLE GETTING THE WHEEL OFF

E:USE THE WHEEL PULLER,  DO YOU KNOW HOW TO USE IT ?

A:NO

E:DO YOU KNOW WHAT IT LOOKS LIKE ?

A:YES

E:SHOW IT TO ME PLEASE

A:OK

E:GOOD, LOOSEN THE SCREW IN THE CENTER AND PLACE THE JAWS AROUND THE
HUB OF THE WHEEL, THEN TIGHTEN THE SCREW ONTO THE CENTER OF THE
SHAFT, THE WHEEL SHOULD SLIDE OFF

A:WHATS THE HUB OF A WHEEL

E:AHHHHH, THATS THE CENTER PORTION OF THE WHEEL,  POINT AT WHERE
YOU THINK IT IS, SHOW ME PLEASE


A:OK JUST A SECOND

E:VERY GOOD, YOU ARE HOLDING IT WITH YOUR HAND

A:ONE OF THE JAWS IS STUCK AND I'M GOING TO LOOSEN IT

E:OK

A:I HAVE THE JAWS AROUND THE HUB  HOW SHOULD I TAKE IT OFF NOW

E:TIGHTEN THE SCREW IN THE CENTER OF THE PULLER AGAINST THE CENTER
E:OF THE SHAFT,  THAT SHOULD SLIDE THE WHEEL OFF  THE SHAFT

A:OK ITS OFF

A:A LITTLE METAL SEMICIRCLE FELL OFF WHEN I TOOK THE WHEEL OFF

E:THAT IS CALLED A WOODRUFF KEY,  IT KEEPS THE FLYWHEEL FROM TURNING
ON THE SHAFT,  BE SURE YOU DON'T LOSE IT.  NOW UNDO THE AIRLINE
CONNECTION AT THE PUMP
```

FIGURE 8   EXAMPLE OF A DIALOG WITH NAIVE APPRENTICE

```
10:57:16 E:THE PUMP PULLEY SHOULD BE NEXT.

10:57:32 A:YES UH DOES THE SIDE OF THE PUMP PULLEY WITH THE HOLES FACE
         AWAY FROM THE PUMP OR TOWARDS IT?

10:58:12 E:AWAY FROM THE PUMP.

10:58:22 A:ALL RIGHT.

10:58:35 E:DID YOU INSERT THE KEY, I.E. THE HALF-MOON SHAPED PIECE,?

10:59:04 A:YES I DID.

11:00:08 E:BE SURE AND CHECK THE ALLIGNMENT OF THE TWO PULLEYS BEFORE
         YOU TIGHTEN THE SET-SCREWS.

11:00:41 A:YES I'M JUST NOW FIDDLING WITH THAT.

11:00:51 E:OK

11:01:04 A:TIGHTENING THE ALLEN SCREW NOW.

11:01:17 E:OK. THANK YOU.

11:01:53 A:THAT'S FINISHED.

11:02:12 E:BY THE WAY, THERE ARE TWO SET SCREWS.

11:02:34 A:THANK YOU.

11:02:55 E:THE BELT CAN BE INSTALLED NEXT.

11:03:13 A:THE SECOND SET SCREW IS TIGHTENED AND I PUT ON THE BELT
         WHEN I  PUT THE PULLEY ON BECAUSE I DIDN'T KNOW HOW TIGHT IT
         WOULD BE.

11:03:54 E:OK, MAKE SURE THE MOUNTING BOLTS FOR THE MOTOR ARE TIGHT.

11:04:24 A:ALL RIGHT.

11:04:51 A:THE MOUNTING BOLTS ARE TIGHT AS IS ALL THE PLUMBING.

11:05:05 E:GOOD. ALL THAT REMAINS THEN, IS TO ATTACH THE BELT HOUSING
         COVER TO THE BELT HOUSING FRAME.

11:05:36 A:ALL RIGHT . I ASSUME THE HOLE IN THE HOUSING COVER OPENS TO
         THE PUMP PULLEY RATHER THAN TO THE MOTOR PULLEY.

11:06:01 E:YES, THAT IS CORRECT.  THE PUMP PULLEY ALSO ACTS AS A FAN
         TO COOL THE PUMP.

11:06:26 A:FINE . THANK YOU.

11:08:44 A:ALL RIGHT THE BELT HOUSING COVER IS ON AND TIGHTENED DOWN.

11:08:57 E:FINE, NOW LETS SEE IF IT WORKS.  PLUG IN THE POWER CORD.
         CHECK TO SEE IF BOTH THE MOTOR AND PUMP ARE WORKING AND
         THAT THE PRESSURE IN THE TANK IS RISING.

11:09:56 A:ALL RIGHT.

11:10:06 E:THERE IS NO ON-OFF SWITCH.  IF ANYTHING GOES WRONG, JUST
         UNPLUG THE POWER CORD.

11:10:41 A:THE PULLEY IS TOO LOOSE, OR THE BELT IS TOO LOOSE RATHER.

11:11:00 E:OK. ADJUST THE TIGHTNESS OF THE BELT BY LOOSENING THE
         MOTOR MOUNTING BOLTS AND SLIDING THE MOTOR.

11:11:47 A:YES.

11:13:02 A:ALL RIGHT I'M TIGHTENING THE MOTOR MOUNTING BOLTS NOW.

11:13:15 E:OK. THANK YOU.

11:17:46 E:WHAT'S HAPPENING NOW?

11:17:55 A:I'M TIGHTENING THE LAST MOTOR MOUNT BOLT... THE HARDEST ONE
         TO GET AT.

11:18:12 E:OK. THANK YOU.

11:18:32 E:I ASSUME YOU ARE USING THE SOCKET WRENCH ON TOP AND THE
         BOX END WRENCH UNDER THE PLATFORM.

11:19:04 A:THE TWO BOLTS OF THE MOTOR MOUNT THAT ARE CLOSEST TO THE
         PRESSURE REGISTER ARE NOT EXPOSED ENOUGH TO ALLOW THE SOCKET
         WRENCH TO GO OVER THE TOP.

11:19:43 E:OK.  I WILL MAKE A NOTE OF THAT.
```

FIGURE 9   EXAMPLE OF A DIALOG WITH EXPERIENCED APPRENTICE

We have found these dialogs have a structure closely related to the structure of the task . This structure and its use in understanding natural language are discussed in Ref. 14. Basically, looking at the structured history is like looking at an outline of the task as performed by the particular apprentice. Some parts of the task are explored in more detail than others. The corresponding part of the "outline" has more detailed levels. Two things at the same level in the "outline" do not necessarily have to follow each other in that order. In fact, this is one of the places where the discourse history differs from a plan for carrying out the task. The discourse history keeps track of the order in which utterances were spoken. It is not concerned with whether this order is a necessary one.

B. Interaction Between the Planning and Natural Language Subsystems

We have also begun studying the interaction between the language and planning components of the system. There are two areas where this interaction is crucial.

First, the language system will need to get information from the planning component to determine subtasks and possible problems at a given stage in the task. This is necessary both for constructing the dialog history and for determining likely future utterances. At present, we envision the planning component of the system producing plans in the form of an acylic graph that encodes a partial ordering. This allows Operations that must occur in a certain order to be distinguished from those that can be accomplished in any order. For example, a partial plan to remove the pump is shown in Figure 10. The arrows show the partial ordering. An operation at the head of an

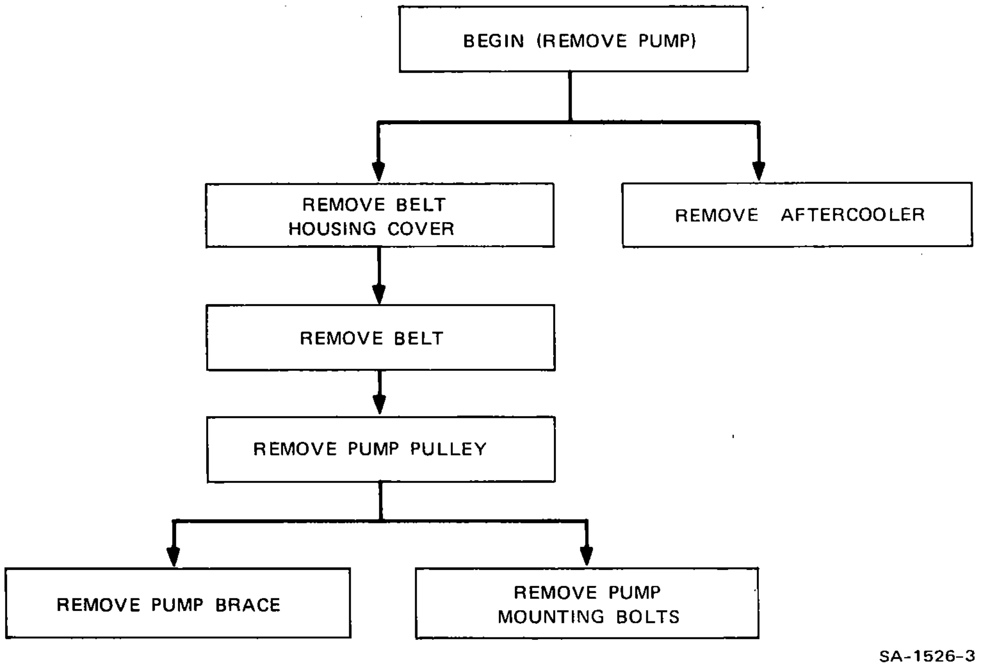


FIGURE 10 PARTIAL PLAN TO REMOVE PUMP

arrow cannot be done until the operation at the tail of that arrow is done. This representation helps both in predicting likely future utterances (i.e., limiting the context) and in constructing the discourse structure. Assume the last utterance expressed the goal represented by some point in the graph ordering. The next utterance can be one of three types: (1) it can give or ask for more information about that goal, (2) it can express completion of that goal, or (3) it can express the goal represented by some other point in the graph. In the example of Figure 10, if the apprentice has just been told to remove the pump pulley, likely next utterances are:

(1) How do I do it? or, Do I have to pull the screws all the way out?

(2) OK, it's off.

(3) Why can't I take the aftercooler off first?

It is clear how this aids the discourse structure. An utterance of type (1) adds a new entry to the structure at a lower level. An utterance of type (2) or (3) causes a new entry at the same level.

Second, a set of general operators and relations must be decided on. Linguistic statements will then be mapped into these operators and relations. This is nesessary for distinguishing different meanings of the same word (e.g., "unscrew" in "unscrew the aftercooler elbow" and in "unscrew the belt housing cover") and for mapping general words into their meaning in the current context (e.g., "put" means different things in "put the nut on the bolt" and "put the nut on the table"). The language system will need these general operators and relations to understand things like "Got it loosened" solves the problem "Bolt is stuck".

IV. PERCEPTION

A. Introduction

Our primary objective is to develop computer based systems that can be rapidly taught to understand (i.e., answer questions about) any class of pictorial scene. This objective encompasses the visual requirements of many applications including, specifically, robots and the computer-based consultant. We are currently pursuing this objective on two complementary fronts, which should begin to merge in the coming year. First, we are continuing work on strategies for locating objects on the basis of distinguishing features. We have sucessfully demonstrated manually developed strategies for finding objects in a room scene /Ref. 15/ and are now trying to emulate the reasoning necessary to program such strategies automatically. Objects will be designated to the computer by circling examples with a cursor in a displayed image. The resulting program will then be run on representative scenes and debugged interactively as errors materialize. Second, we are developing techniques for obtaining reasonably complete explanations of scenes detailing major objects and events and how they relate semantically. An interactive scene interpretation system has been built that attempts to rule out ambiguous local interpretations using global contextual constraints. The system appeals to a tutor for additional knowledge and advice when it gets stuck.

A third effort has begun, concerned specifically with satisfying the vision requirements of a computer consultant set forth in our management plan. The principal accomplishment to date has been the formulation of a thorough research and development plan, outlining the required capabilities and how they will be achieved. Work is now under way in support of graphical communications whereby both man and machine will be able to designate objects to each other by pointing at them. The shape representations and geometrical models developed in this regard should allow our scene analysis systems to be utilized in the computer-consultant domain.

B. Locating Objects by Distinguishing Features

Our research plan for this project was first to experiment with manually programmed strategies for finding objects in an office scene based on their distinguishing features, and then to automate the programming process. In this section, we shall first demonstrate the effectiveness of the distinguishing features approach, by describing some manually developed strategies for finding common objects in a simple room scene. We shall then demonstrate some comparable strategies generated by the computer on the basis of program modules abstracted from successful manual strategies.

1. Generating Distinguishing Feature Strategies Interactively

To facilitate the development of distinguishing feature strategies, we constructed an interactive system, ISIS (Interactive Scene Interpretation System, Ref. 15). This system allows an experimenter to define basic perceptual concepts for the computer using pictorial examples. Strategies such as the telephone finder, though trivial to describe to a person, would take months to translate into code using conventional programming systems. The difficulty stems from the awkwardness of formalizing primitive descriptive concepts (such as black, rectangular, horizontal) as symbolic programs.

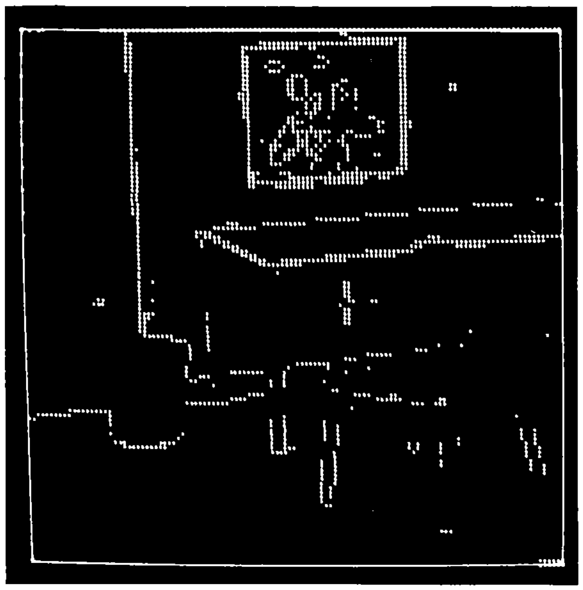
In ISIS, examples are designated graphically by encircling portions of a displayed image with a cursor. Given a pictorial example, a representation can be empirically constructed by trying feature extraction operators in order of increasing complexity until the example is sufficiently distinguished from previously determined representations. Concepts so defined constitute a common vocabulary, shared by man and machine, that can be used symbolically in describing objects and specifying scene analysis procedures.

Briefly, the role of ISIS in strategy development can be explained as follows. To describe concepts in terms of the computer's primitive functions, a human must know (or be able to determine easily) what those primitive functions can distinguish. ISIS provides an experimenter with tools to choose primitive functions empirically. The system first generates a color display of the scene. This display conveys qualitatively which color boundaries are easily discriminated by the computer. The experimenter can circle regions of the displayed image and obtain from the system average or extreme numerical values for local surface attributes such as height, hue,, saturation, and orientation. By applying such operators to examples and counterexamples of pictorial concepts in the image, he can discover which operators sufficiently discriminate an object.

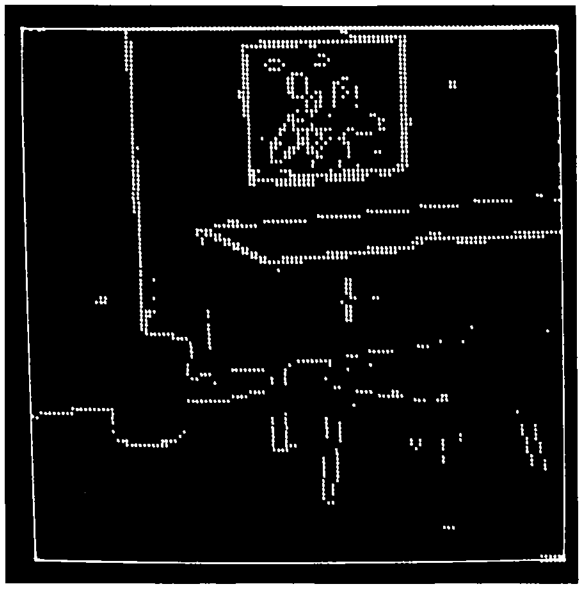
Proposed descriptions can be tested by requesting the system to illuminate all parts of the displayed scene that correspond to the description.

The basic scheme underlying distinguishing feature strategies is to rapidly disqualify obviously irrelevant areas of the scene by sampling for characteristic attributes of the desired object. Additional local properties are then tested to eliminate those samples belonging to other objects that share the acquisition attributes. The surface containing the acquired samples is then bounded and the resulting surface tested for appropriate size and shape. We have interactively developed programs of this sort for finding various objects commonly found in office scenes (e.g., table, door, wall, picture, and floor).

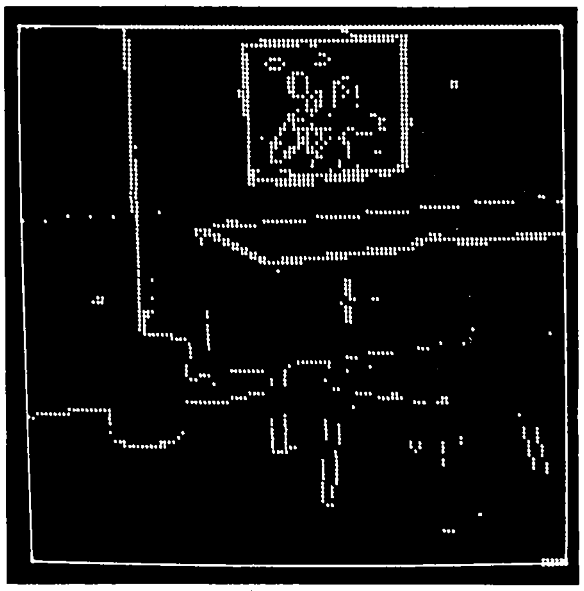
A basic plan for finding a door might consist of acquiring samples at a height likely to contain few things besides a door, and then validating by eliminating those samples which on the basis of remaining local attributes have low likelihood of being part of a door. The door boundary is then obtained by growing a region around the initial samples using attributes such as hue and saturation. One alternative approach, illustrated in Figure 11, would be to first scan downward from the top of the image to a point whose height is unique to DOOR and WALL. The edge of the door is then obtained by scanning horizontally from this point for an abrupt change in saturation. (Doors in SRI offices are deep brown while walls are tan.) If this edge has sufficient vertical extent, the door boundary is inferred by a special vertical rectangle bounding module (VRB) from the known shape and size of our office doors, and the local



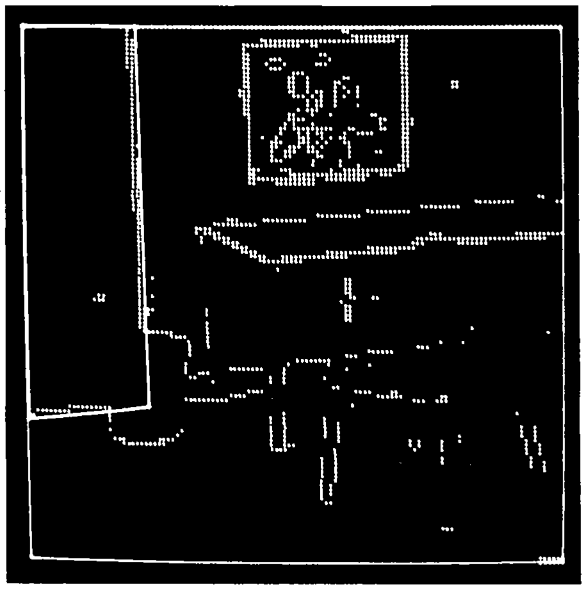
(a) IMAGE IS SAMPLED VERTICALLY DOWN ITS LEFT EDGE UNTIL A HEIGHT ≤ 2.8 FEET IS OBSERVED



(b) IMAGE IS SAMPLED VERTICALLY DOWN RIGHT EDGE UNTIL A HEIGHT ≤ 2.8 FEET IS OBSERVED



(c) IMAGE IS SAMPLED HORIZONTALLY LEFT TO RIGHT ALONG THE LOCUS ESTABLISHED IN STEPS (a) AND (b) UNTIL A SATURATION DISCONTINUITY ≥ 0.2 IS DETECTED



(d) A BOUNDARY IS COMPUTED ON THE BASIS OF LOCATION AND SURFACE ORIENTATION MEASURED AT THE DETECTED EDGE POINT, AND A PRIORI KNOWLEDGE ABOUT THE SIZE, SHAPE, AND HEIGHT OF DOORS

SA-1187-13R

FIGURE 11 OPERATION OF DOOR-FINDING PROGRAM

surface orientation is then measured in the image on the door's side of the discontinuity. Finally, the computed edges are confirmed by testing for evidence of the predicted boundaries in the image.

ISIS makes it very easy to empirically evolve these ad hoc strategies. One can interactively specify arbitrary predicates for scanning filtering, or region-growing, and immediately see what ambiguities arise. Appropriate validations can then be proposed and tested. It is readily seen where a program is going awry, and why, and usually just as easy to quickly attempt something new. A library of useful modules like VRB were developed in the course of experimenting with strategies such as those outlined (see Ref. 15 for a list). More important than the modules, however, are the strategy schemata that we have abstracted for generating such programs automatically.

2. Program Schemata

Distinguishing feature strategies can be schematized into acquisition, validation, and bounding phases. Specific techniques for accomplishing each phase are chosen from available modules on the basis of current information about the pictorial domain.

The most straightforward way of acquiring samples of the object is to filter random picture samples, rejecting any sample that fails to pass a predicate descriptive of the desired object. For example, the sampled scene may be filtered to retain only those samples at a certain height. Or alternatively, the scene can be scanned in order to detect distinguishing boundary discontinuities.

Acquired samples are validated by checking additional local characteristics that distinguish the desired object from others with common acquisition attributes. These additional attributes may be examined sequentially, reducing the number of samples that must be examined by successive tests. The order of testing can be devised to minimize total cost by deferring computationally expensive procedures. At some point, the cost of planning the optimal order will exceed the cost of simultaneously using all remaining attributes to classify samples as belonging to possible objects. Those samples judged likely to belong to the desired object are retained. The validation of boundary samples entails testing the type of boundary (e.g., a color boundary of a certain strength, its length and orientation). A common boundary validation is to use a mask to check for suitable length.

The final bounding phase is accomplished in various ways. If a suitably constrained model is available, expected boundaries can be projected onto the image and validated using extent masks as above (Ref. 15). In the absence of such a model, a region may be grown around validated points; neighbors of validated samples that pass a chosen predicate (often the acquisition predicate) are added to a push-down stack. Neighbors of points on this stack are then examined recursively. All accepted points are collected together into a region. A quick, crude boundary for a set of samples can also be obtained by computing a convex hull. Finally, there are special purpose modules that can produce the best global boundary for various common surfaces (e.g., horizontal or vertical rectangular surfaces) consistent with a set of detected edge points.

3. Generating Strategies Automatically

Our approach to automatic strategy generation is to emulate the reasoning of a human programmer trying to locate objects in a scene. The programmer has a goal (or set of goals) to be achieved (such as finding the table top), and he has available knowledge about the world, a set of useful program modules, models of those modules that indicate their prerequisites, and finally, the basic distinguishing feature strategy schemata.

The generation process is flowcharted in Figure 12. The system first checks to see if any existing plan might be of use; if one is applicable, then it is executed. Otherwise, a check is made for applicable modules. If some are found, then any preconditions are handled by a recursive call on the planner. Thus, if VRB is applicable for finding the object, the planner must next tackle the subgoal of finding edge points expected as input. If the preconditions can be satisfied, the plan is executed; otherwise, (or if there were no relevant modules) acquisition and validation features are selected for the default strategy. After execution, global features are checked to ensure that the desired object has been found, with high confidence. If satisfactory, the system exits. Otherwise, a decision is made to resume execution looking for another object or to select an alternative plan.

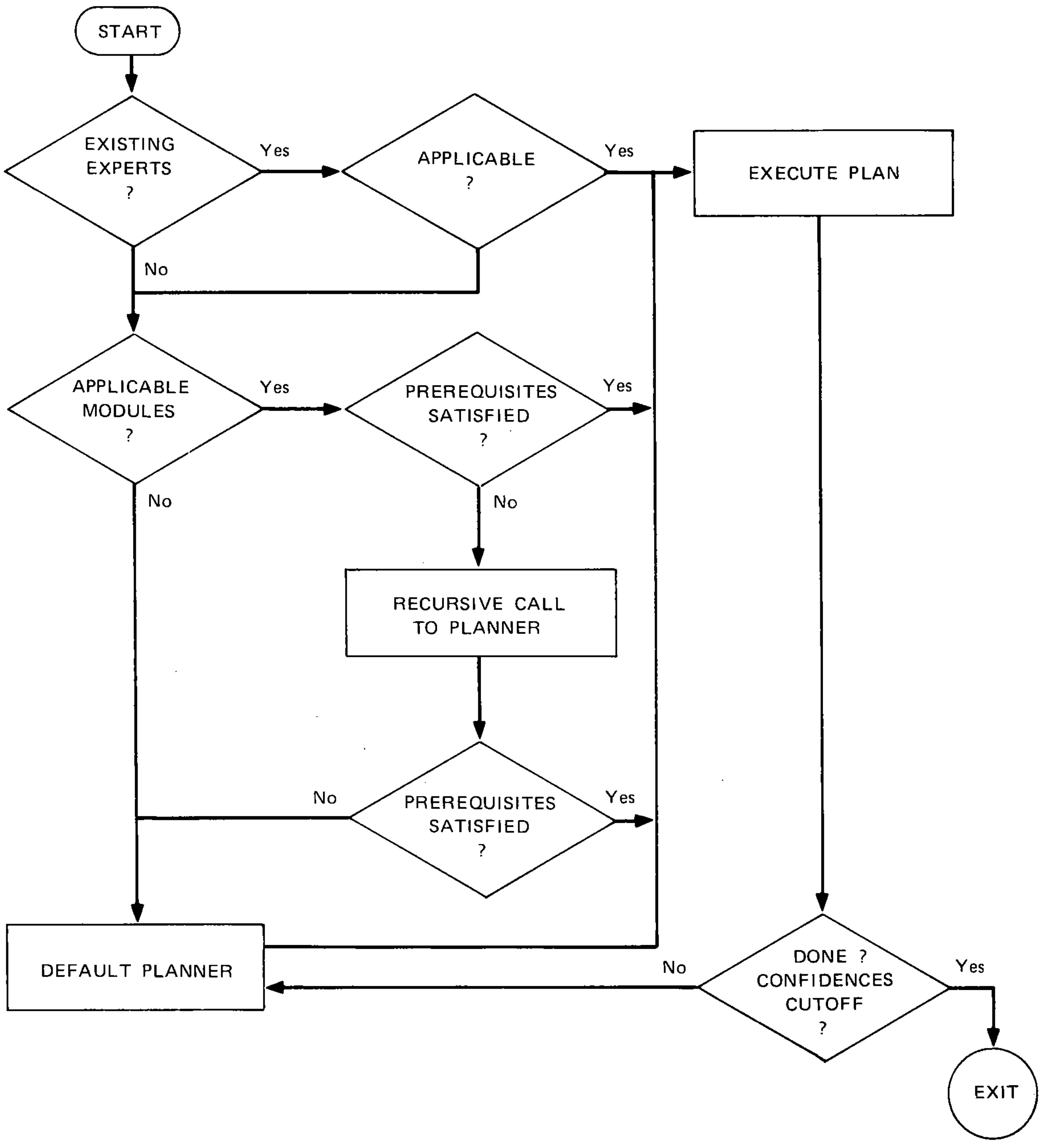


FIGURE 12 PLAN GENERATION PROCESS

The selection of distinguishing and criterial acquisition features for the default scheme involves a mixture of pragmatics with elements of decision theory. Acquisition attributes are chosen on the basis of how well they distinguish the object from other objects, how criterial (i.e. characteristic) they are to the object, and how costly they are to measure. Local attributes (such as hue or local orientation) tend to be determined more easily and reliably than do global ones (such as size or shape). Global attributes, however, are often more criterial than local properties. In some cases, it may be advisable to first locate another object and use that object to localize the desired one. A human programmer can intuitively weigh the available choices. The automatic programmer must resort to quantitative criteria; they are currently provided by routines to compute cost and confidence for some operator or module. Cost is computed (in milliseconds) from the expected effort entailed in applying the operator. Confidence is a measure of the reliability of a hypothesis based on the operator's performance (e.g., the probability that samples selected by the operator belong to a given object). It is expected that these computations will be performed only once in a given context and the results looked up in succeeding analyses.

The modular strategy scheme described above eases the task of generating strategy programs. Each phase has very explicit objectives. During acquisition, for instance, the program is trying to locate samples from either the surface of an object or from its boundary. It is plausible, then, to associate specific error modalities with each objective and to build in appropriate checks and countermeasures . Moreover, the system can include alternative methods of achieving each subgoal. For example, a horizontal plane might be acquired by looking for samples of a given height and orientation, or alternatively, if these data were noisy, by accepting all samples between two height extremes, and then checking for contiguity in other features.

4. Experimental Results

The programs shown in Figures 13 and 14 were generated automatically according to the paradigm of Figure 12. These programs will be used to illustrate important features of the planning process, notably the integration of pragmatic decisions with utility analysis. Note, also, how execution and examination of interim results are interspersed with further planning.

Figure 13 traces the development of a program for finding a table top (TTOP) in scenes such as that shown in Figure 15. Line 1 of Figure 13 shows the initiation of the process, FIND(TTOP). The first subgoal (an executable LISP form) is generated and executed beginning in line 2--(ACQUIRE TTOP). This function first attempts to find a plan for acquiring some samples of TTOP. When no existing plan is found, another check turns up an applicable module, HORIZONTAL. This module is added to the goal list, and expanded by the system in line 7. Line 9 indicates that HORIZONTAL uses HEIGHT and ORIENT (local surface orientation) as pragmatic acquisition features. The expected confidence and cost due to this module appear next. The confidence that any samples passing the height and orientation checks will belong to TTOP is 1.0, since no other known horizontal objects have the same height as the table. The expansion of the module

```
1  FIND:
   Beginning planning to find TTOP
2  *** execution of form:
    (ACQUIRE TTOP)
3      Acquisition planning:
4      No existing plan for TTOP -- attempting to generate one
5      Module selection:
       Applicable modules are:  ((HORIZONTAL TTOP))
6  *** execution of line 2 complete ***
7  *** execution of form:
    (HORIZONTAL TTOP)
8      Horizontal surface acquisition planning:
       Horizontal surface module
9      Horizontal surface acquisition planning:
       Locating TTOP using height and orientation for acquisition
10      Horizontal surface acquisition:
       Expected acquisition confidence is -- 1.0
11      Horizontal surface acquisition:
        Expected acquisition cost is 4700. milliseconds
12  *** execution of line 7 complete ***
13  *** execution of form:
     (SAMPLE-WINDOW SCENE 129)
14      Sample-window:
        Sampling current window at density of .00911
15  *** execution of line 13 complete ***
16  *** execution of form:
     (FILTER-WINDOW (LAMBDA (X)
                           (AND (LIMITP (QUOTE (2.14 2.66))
                                        (HEIGHTS X))
                                (LIMITP (QUOTE (-3.05 3.05))
                                        (ORIENTS X)
17      Filter-window:
        Filtering window
18      Filter-window:
        5 samples accepted
19  *** execution of line 16 complete ***
20  Horizontal surface acquisition:
    There are no ambiguous objects remaining
21  *** execution of form:
     (VALIDATE TTOP)
22      Validate:
        Validated confidence (using  (HUE SAT ORIENT HEIGHT)) is: 1.0
23      Validate:
        No ambiguous objects
24  *** execution of line 21 complete ***
25  *** execution of form:
     (BOUND TTOP)
26      Module selection:
        Applicable modules are:  ((BOUND-HORIZONTAL TTOP))
27  *** execution of line 25 complete ***
28  *** execution of form:
     (BOUND-HORIZONTAL TTOP)
29      Bound-horizontal:
        Growing surface from acquired samples
        using acquisition predicate
30  *** execution of line 28 complete ***
31  *** execution of form:
     (GROW-SAMPLES (LAMBDA (X)
                          (AND (LIMITP (QUOTE (2.14 2.66))
                                       (HEIGHTS X))
                               (LIMITP (QUOTE (-3.05 3.05))
                                       (ORIENTS X)
32      Grow-samples:
        Growing outward from current samples
33  *** execution of line 31 complete ***
DONE
```

FIGURE 13   PROGRAM FOR FINDING TTOP

```
1  FIND:
   Beginning planning to find DOOR
2  *** execution of form:
    (ACQUIRE DOOR)
3      Acquisition planning:
4      No existing plan for DOOR -- attempting to generate one
5      Module selection:
       No applicable modules, defaulting
6  *** execution of line 2 complete ***
7  *** execution of form:
    (DEFAULT DOOR)
8  *** execution of line 7 complete ***
9  *** execution of form:
    (DFLT-ACQUIRE DOOR (HEIGHT (3.19 . 5.28)))
10  *** execution of line 9 complete ***
11  *** execution of form:
     (SAMPLE-WINDOW SCENE 84)
12      Sample-window:
        Sampling current window at density of .00593
13  *** execution of line 11 complete ***
14  *** execution of form:
     [FILTER-WINDOW (LAMBDA (X)
                           (LIMITP (QUOTE (3.19 5.28))
                                   (HEIGHTS X]
15      Filter-window:
        Filtering window
16      Filter-window:
        25 samples accepted
17  *** execution of line 14 complete ***
18  *** execution of form:
     (VALIDATE DOOR)
19      3 samples selected from acquisition samples
20      Validation confidence is 1.0
21  *** execution of line 18 complete ***
22  *** execution of form:
     (BOUND DOOR)
23      Module selection:
        Applicable modules are:  ((BOUND-VERT DOOR))
24  *** execution of line 22 complete ***
25  *** execution of form:
     (BOUND-VERT DOOR)
26      HEIGHT not applicable
27      Growing on basis of  (SAT HUE)
28  *** execution of line 25 complete ***
29  *** execution of form:
     [GROW-SAMPLES (LAMBDA (X)
                          (AND (LIMITP (QUOTE (.317 1.0))
                                       (SATS X))
                               (LIMITP (QUOTE (4.74 120.))
                                       (HUES X]
30      Grow-samples:
        Growing outward from current samples
31  *** execution of line 29 complete ***
DONE
```

FIGURE 14   PROGRAM FOR FINDING DOOR

causes new program steps to be added to the plan. The first of these in line 13 causes the current window (in this case, the scene) to be sampled at an adequate density determined by the table's known size, anticipated range, the viewing angle, and other a priori model information. The sampling produced by this routine is shown in Figure 16. This sampled image is then filtered using a predicate generated by HORIZONTAL (LIMITP is a LISP predicate for checking whether the value of a function is between two limits). The predicate is applied to the samples in Figure 16, and only those passing the tests are retained. The filter passes seven samples, the bounded set shown in Figure 17. This concludes the acquisition phase of the program.

In the validation phase, the samples are checked to ensure (as well as can be determined) that they do belong to the table top. In this case, all acquired samples are retained. The validation confidence is 1.0, and there are no ambiguous objects.

The final operation is to bound accurately the surface encompassing the acquired samples. Again, a specialized module for bounding horizontal surfaces is used. This module generates code to grow the surface, starting from the acquired points by recursively including all neighbor samples satisfying the initial sample acquisition predicate. The resulting bounded surface is shown in Figure 18.

Figure 14 details a program for finding DOORS. As no applicable plans or acquisition modules exist, the system utilizes the default schemata based on known attributes of doors. Default planning requires the selection of applicable detectors. Candidates are chosen

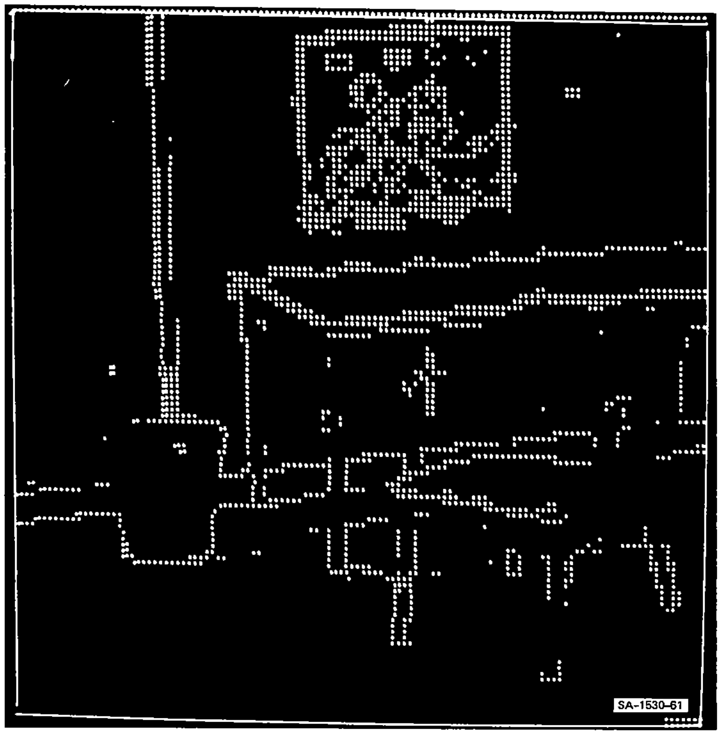


FIGURE 15 GRADIENT PICTURE OF SCENE

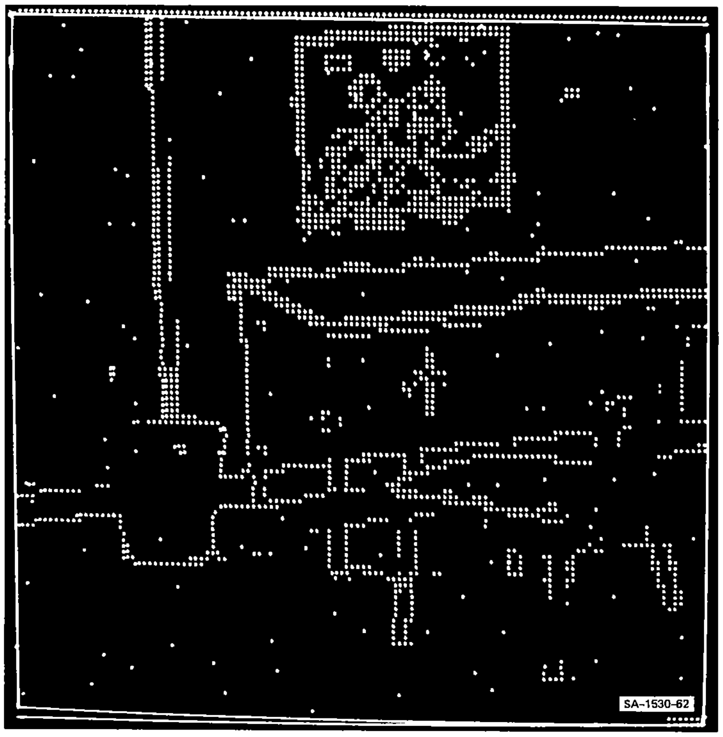


FIGURE 16 OFFICE PICTURE RANDOMLY SAMPLED BY TTOP PROGRAM

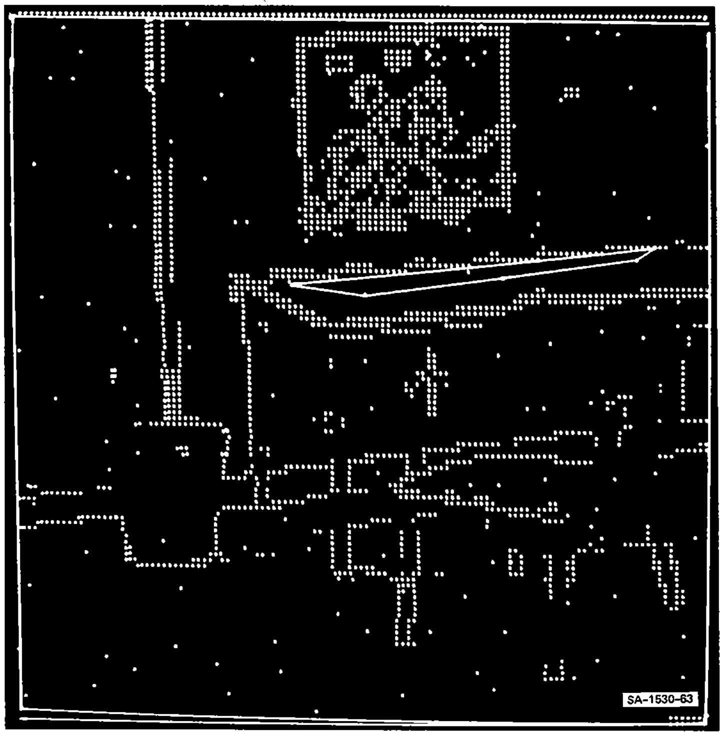


FIGURE 17 OUTLINE OF SAMPLES ACCEPTED BY FILTER

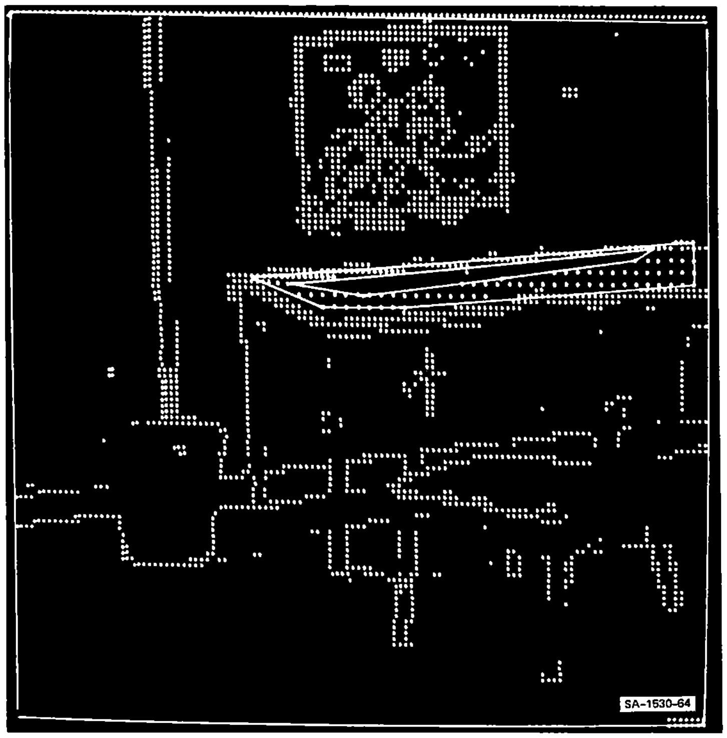


FIGURE 18 OUTLINE EXTENDED TO EDGE OF TTOP BY GROW

in two ways: First, the ranges of attribute values for the object are examined and detectors for peaks in those ranges are selected. This allows the system to choose criterial detectors with no thought to the distinguishability of objects based on those detectors. Additional detectors are then chosen from segments of the attribute ranges which minimize the number of ambiguous objects.

From the resulting set of criterial and distinguishing detectors the system selects optimal subsets for acquisition and validation. In Figure 14, a range of height values was selected for acquisition in order to cheaply obtain a minimum set of ambiguous objects.

The image is actually sampled in line 11; the resulting sampled picture is shown in Figure 19. These raw samples are next filtered with the distinguishing height predicate shown in Line 14, eliminating all but the 25 points, shown bounded in Figure 20.

Validation has been handled as a bottom-up classification problem. Each acquired sample is classified on the basis of all its attribute values as belonging to some object(s) with some confidence. Currently, if the confidence that a sample belongs to the desired object is greater than 0.85, then it is retained; otherwise, it is discarded. In the DOOR example, the six points shown in the upper right-hand corner of Figure 21 were retained out of the original 25 acquired.

Finally, the outline of the door must be determined. The module for bounding vertical surfaces is chosen. This program generates a predicate for the region-growing routine mentioned earlier. HEIGHT

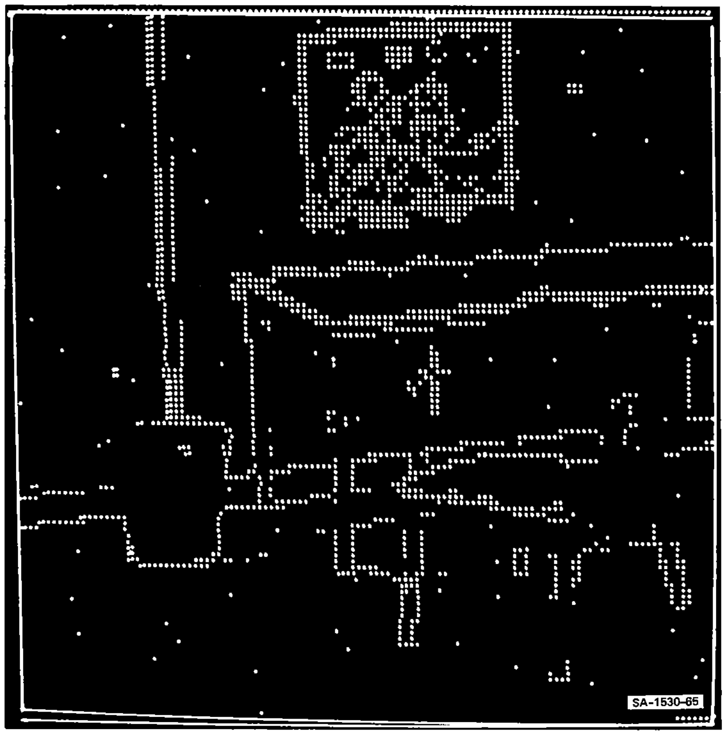


FIGURE 19 OFFICE PICTURE RANDOMLY SAMPLED BY DOOR PROGRAM

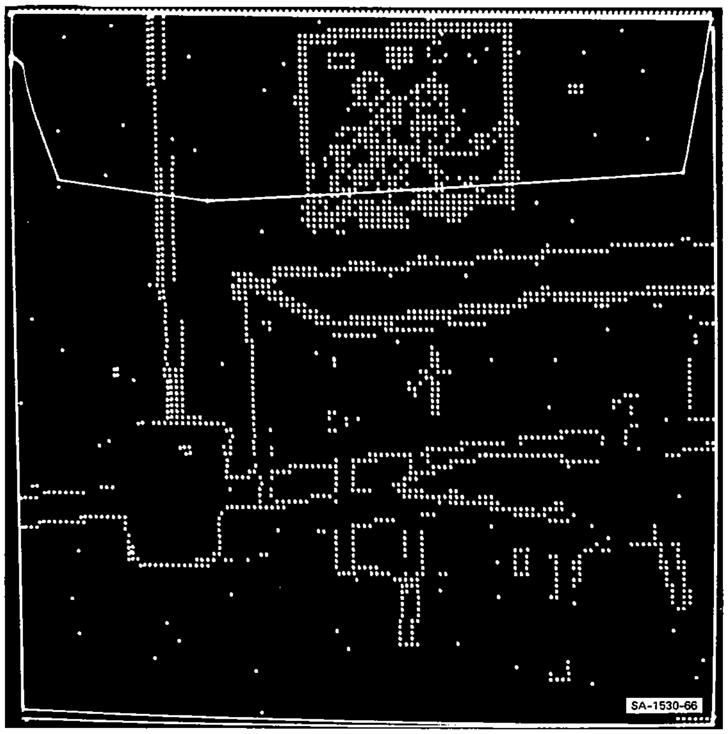


FIGURE 20 OUTLINE OF SAMPLES ACCEPTED BY HEIGHT FILTER

is disgarded as a useful attribute (since the door covers almost the entire extent of heights in the scene) in favor of a combination of hue and saturation. The resulting outline of the door is shown in Figure 22.

5. Discussion

The goal of locating a specific object in a scene avoids the harder, less well-defined problem of describing the scene. The system relies heavily on information from different sensory modalities to describe objects in terms that are semantically related to the object. Thus a horizontal surface is characterized by height and orientation. In a monocular black and white picture, such attributes can only be inferred after identifying the object and all of its supporting structures. With the information already available, we can make use of it to directly isolate the object in the scene. We feel that our distinguishing feature schemata provide an effective framework in which to integrate multisensory data.

The identification and correction of program bugs might be considered the essential strength of the human programmer. It is this strength that must be emulated to provide a robust system capable of generalizing its own plans to overcome unforseen contingencies. The programming system anticipates certain types of recurring bugs that will occur beforehand and inserts checks at appropriate points in a plan. For example, local validation can be viewed as a check for errors of commission during acquisition. Global validation is performed in anticipation of systematic errors arising during acquisition and bounding. Checks for anticipated problem areas

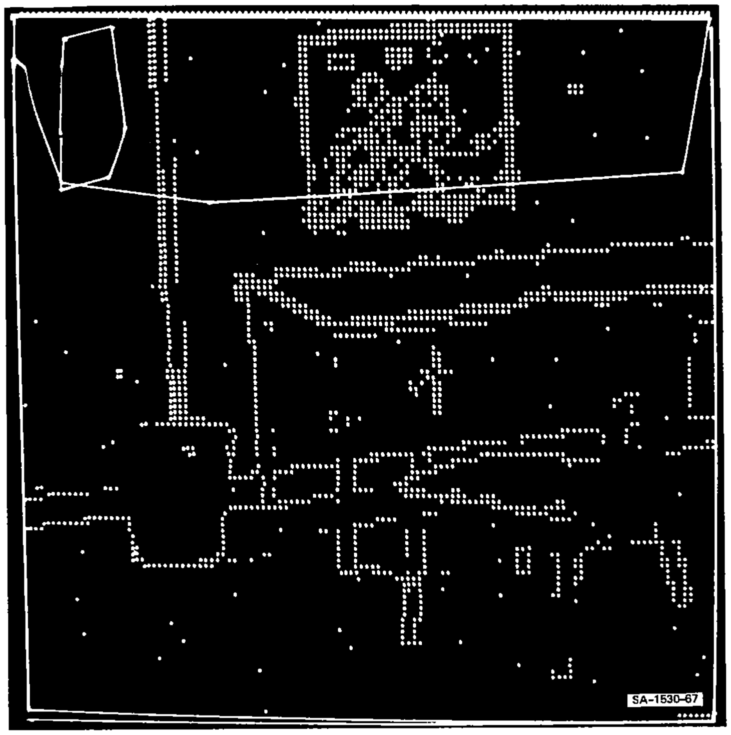


FIGURE 21 SAMPLES RETAINED BY VALIDATION

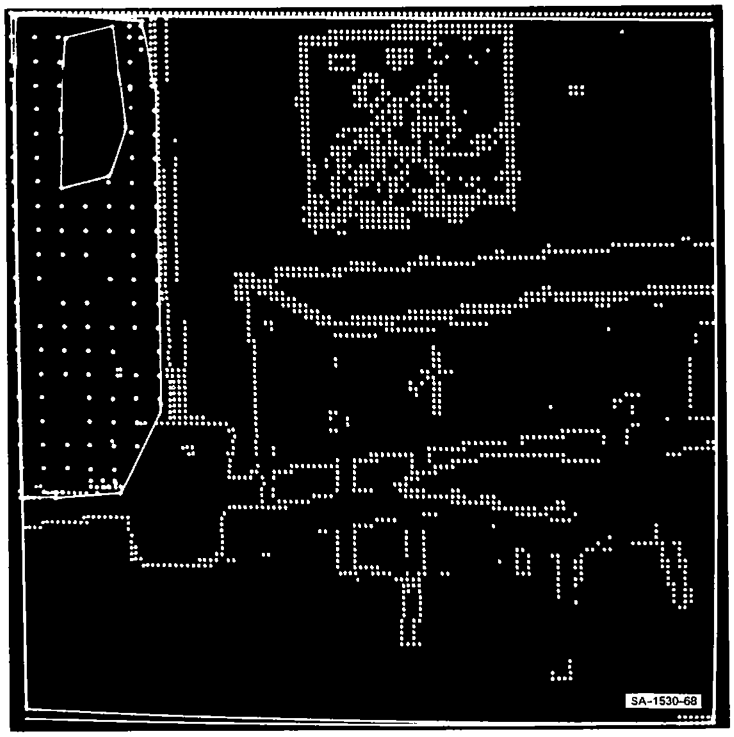


FIGURE 22 DOOR REGION EXTENDED BY GROW

(such as occlusion) can be included by an analysis of their effect on the process of bounding. The basic philosophy followed, though, is not to complicate a strategy unnecessarily by anticipating all possible contingencies. Should a program actually fail, we would then try to patch it by interactively experimenting with additional constraints. The world model would then be updated to preclude a recurrence.

Our emphasis on distinguishing features should not be construed as a denigration of other work on complex representations. Our contention is that although cases will arise that will require subtle tests to differentiate among several objects, for the most part relatively simple descriptions will suffice. Furthermore, it is our intention that the system we are building will provide a conceptual framework for including the best available methods for locating and representinng perceptual information. We have begun by modeling some of the simpler processes with respect to the tasks for which they are useful, their cost, and their expected types of errors. We will extend this modeling to allow us to incorporate any effective procedures and representations, developed by ourselves or others.

C. Reasoning About Scenes*

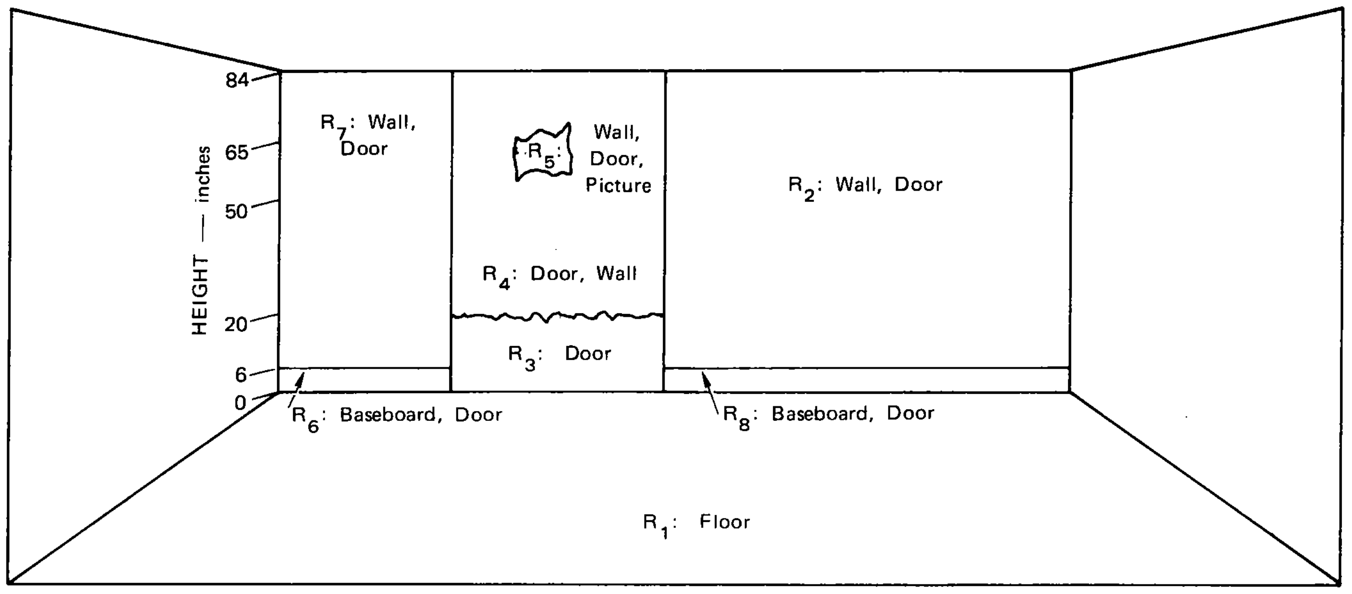
1. Introduction

The previously described object finding techniques, though adequate for simple room scenes, have some obvious limitations as general strategies for perception. These techniques rest on the fact that it is often possible to distinguish amongst a limited number of objects solely on the basis of locally perceptible attributes such as hue, height, and surface orientation. However, in less constrained environments, or with less complete sensory data (i.e., black and white images) it generally is not possible to assign unique interpretations to parts of a scene taken out of context. Different objects may have similar appearances, while objects belonging to the same functional class can have strikingly different appearances (e.g., coffee cups). Ambiguous local interpretation must then be ruled out by using contextual constraints to achieve a meaningful, globally consistent interpretation of the whole scene.

Let us illustrate with a trivial example the style of reasoning entailed in scene interpretation. Suppose that a segmentation procedure initially partitions a room scene into regions as shown in Figure 23. The labels in the figure indicate the possible local interpretations of those regions, based on their individual attributes and the descriptive semantics given in Table 5(a). For

---

*The work reported in this section was partially supported by ONR under Contract No. N00014-71-C-0294.



NOTE: Correct interpretations are listed as the first possibility.

SA-1530-36

FIGURE 23 POSSIBLE REGION INTERPRETATIONS OF A SIMPLE ROOM SCENE

Table 5

PARTIAL LIST OF SEMANTICS FOR DOMAIN OF FIGURE 23

(a) Descriptive Semantics

1.	Floor:	height = 0 in.
		orientation = horizontal

2.	Baseboard:	0 in. < height < 6 in. orientation vertical
3.	Wall:	6 in. < height < 84 in. orientation vertical
4.	Door:	0 in. < height < 74 in. orientation vertical
5.	Picture:	30 in. < height < 70 in. orientation vertical

(b) Relational Semantics

x. (NOT (ABOVE WALL DOOR))

y. (NOT (ADJACENT PICTURE DOOR))

z. (HOMOGENEOUS DOOR)

example. R3 must be part of a door, since doors are the only vertical surfaced objects that extend from floor level (which excludes wall and picture) to a height greater than six inches (which excludes baseboard). To further narrow the interpretation possibilities, it is necessary to invoke relational semantics like those in Table 5(b). A typical line of reasoning might proceed as follows:

1. R4 cannot be a wall by rule X, since it is above R3, door.

2. R5 then cannot be a picture by rule 1, since it is adjacent to R4, door.

3. R5 also cannot be a wall by rule 4, since it is above part of R4, door.

4. R2 cannot be a door by rule Z (homogeneity) since it is adjacent to but significantly more saturated than R4, door.

5. R8 cannot be a door as that would violate rules X and Z.

6. By similar reasoning to that used in steps 4 and 5 above, neither R6 nor R7 can be a door.

The above example is an attempt to explain observed sensory data in terms of semantic knowledge about the depicted domain. The explanation represents a mapping of pictorial entities (i.e., regions and lines) onto world concepts (i.e., objects) such that the locally perceived attributes of each region are consistent with those of a corresponding world concept and moreover, that global constraints among the concepts are preserved in the image.

An explanation of this sort need not be limited to descriptive semantics at a pictorial level. Explanations can extend through many levels of knowledge ranging from functional constraints on form and material composition to knowledge about optical phenomena such as occlusion and shadowing. For example in a machinery context, flywheels are recognized not by a visual attribute of a particular instance (e.g., number of spokes) but rather on the basis of characteristics associated with all flywheels: they are round with mass concentrated toward the rim. A class of machines such as compressors will share functional constraints on form and interconnection that ultimately could be used to identify parts on a particular unit not previously encountered. A scene-understanding system that can reason at this level is clearly a formidable long-range goal that draws on many areas of artificial intelligence research. Like an infant, we are concentrating initially on explanations involving surface level descriptions.

The scene understanding problem as formulated above focuses attention on a number of key issues for AI research.

1. What knowledge is necessary, sufficient, useful, and so on for understanding various classes of scenes?

2. How might this knowledge be represented in a computer?

3. How should this knowledge be acquired?

4. How can this knowledge be used effectively for scene interpretation?

These questions are readily answered for trivial problems such as the introductory example of Figure 23, but pose very formidable design problems for systems dealing with real scenes. In the remainder of this section, we will discuss some answers developed in the context of room scenes. The approach seems general enough to apply in arbitrary domains (e.g., our workstation) given appropriate semantics and representations. We begin by summarizing the important features and liabilities that characterized past attempts at scene understanding. Against this background we will describe in detail the design of a proposed scene understanding paradigm intended to overcome limitations of previous systems. The paradigm is then illustrated by simulating the analysis of a representative scene. This Gedanken analysis provides a feeling for the type of knowledge necessary to understand simple room scenes and also an idea of the required depth of reasoning.

2. Background

The objective in scene interpretation is to partition a scene into regions and assign the most likely interpretations consistent with global semantic constraints. Formally stated, the objective is to maximize the product of probabilities that (region i has interpretation j) over all partitions of the scene into regions and all assignments of interpretations to regions. Unfortunately, the search space explodes combinatorially with scene complexity and universe of possible interpretations, rendering brute force search unfeasible.

A number of investigators have proposed doing scene interpretation by tree search wherein interpretations are assigned to regions in order of maximum likelihood, subject to semantic consistency with interpretations previously selected for other regions. [Refs. 16, 17, 18, and 19] Exhausting all possible interpretations for an unassigned region forces reconsideration of an earlier assignment. With a few exceptions these proposals all shared a number of unrealistic assumptions imposed to make the search feasible:

1) A correct partioning of the scene into regions was presumed.

2) Interpretations were drawn from a very small universe, usually less than 10.

3) Only pairwise semantic constraints on legally adjacent region interpretations were utilized.

The first assumption is probably the most unrealistic, since the only hope of obtaining reliable partitioning is to integrate the segmentation and interpretation processes. Yakimovsky has made a good start in this direction. His program takes a color picture as input and first partitions it into uniform elementary regions. Next, two adjacent regions with weakest common boundaries are merged. Merging iterates in this fashion until a manageable number of regions are obtained (i.e., 75-100). It is assumed that, with this many regions, few if any incorrect merges will have been made. Semantics now enter the arena. A probability is assigned to all possible interpretations of each region based on prototype attributes learned from examples. Independently for each boundary between regions, possible boundary interpretations and associated probabilities are ascribed. These two probabilities are then used to compute the likelihood that adjacent regions both have the same interpretation. The merging process resumes joining, at each step, the pair of regions most likely to have the same interpretation. Merging continues so long as a measure of total likelihood over the complete picture interpretation increases.

Although Yakimovsky's system performs impressively in analyzing real road scenes, his formulation has several weaknesses. Merging and interpretation are too integrated; merging should not be the only means for adjusting interpretation likelihoods. As a result of this entanglement, the system can only utilize contraints on pictorial (i.e., two dimensional) adjacency. There is, for example, no obvious way to utilize the fact that sky is still above road, even when trees intervene. The system also cannot deal adequately with different instances of the same type of object: two regions interpreted as car will probably be merged if they are adjacent, even when they correspond to different cars. The system is heavily rooted in probabilities. While this permits consideration of arbitrarily unlikely situations, it precludes semantically valid interpretations in previously unseen relations; a vision system ought to be able to infer that X may occlude Y, even if it has never seen it happen. Moreover, it is by no means clear that the probabilities can be meaningfully evaluated, especially in temporally varying scenes. Finally, the present control structure does not permit the merging process to backtrack. The assumption that always performing the locally best operation will lead to the globally best result is very weak, particularly since the semantic merging criteria is a consensus over all joint interpretations of an adjacent pair of regions. It remains to be seen how well Yakimovsky's system will do in scenes with more than a trivial number of interpretations.

Another inherent fault of all of the above systems is their common dependence on conventional search algorithms. A tree search will inevitably perform a large amount of redundant work by throwing away hard won deductions whenever it backtracks. In one of Duda's examples, for instance, a particular region is deduced not to be an electrical outlet on at least five separate branches of the interpretation tree. In each context, outlet was excluded as a possible interpretation because the region in question was adjacent to a region that had already been postulated to be the floor. In his important thesis, Waltz (Ref. 20) argued that such redundant reasoning could be sharply reduced by eliminating as many globally inconsistent interpretation possibilities as one could before embarking on a search.

Specifically, Waltz advocated the use of cheap pairwise compatability tests to eliminate interpretation alternatives (such as electric outlet) that were clearly inconsistent with previously constrained interpretations (such as the floor). It had been widely feared that increasing the universe of possible interpretations, and correspondingly the number of constraints, would completely bog down the search for a good scene interpretation. Waltz showed that, with his technique, more semantic information could be constructively utilized.

3. An Interpretation System

In this section we introduce a new paradigm for scene interpretation that combines important principles from the various systems referred to above. Our emphasis is first to integrate segmentation and interpretation on a truly semantic basis, and second to utilize semantic knowledge efficiently in arriving at a consistent (and in some sense best) interpretation of the whole scene. The system is outlined schematically in Figure 24. We will first present an overview of its operation, and then elaborate on the design of the various components.

a. Overview

Briefly, the system is intended to work as follows. A preliminary description is obtained by partitioning a scene into homogeneous regions and assigning to each region all interpretations permitted by its local pictorial attributes (e.g., it may be door or wall on the basis of color, size, and surface orientation). A likelihood is associated with each interpretation. The deductive stage next reevaluates the likelihood of each local interpretation based on pairwise consistency with the possible interpretations of contextually related regions. If the likelihood of an interpretation is altered, then any previously considered interpretations for which this current interpretation provided contextual support must be reexamined.

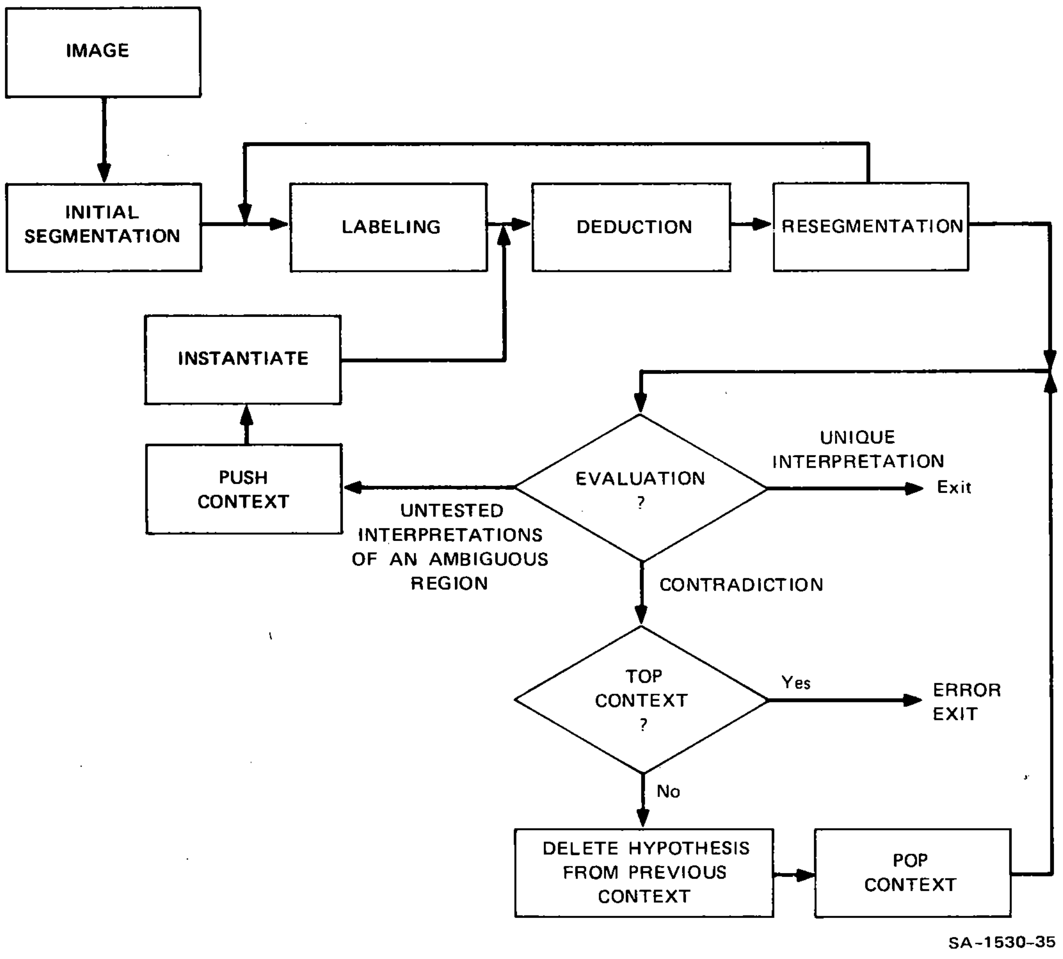


FIGURE 24 OVERVIEW OF SCENE INTERPRETATION SYSTEM

Any interpretation that is shown to be inconsistent with all possible interpretations of another region can be eliminated immediately. Since eliminating an interpretation may remove the only contextual support for an interpretation of some other region, the effect of a deletion can be to propagate another deletion. This chain of deduction, similar to Waltz's "filtering" technique [Ref. 20], is pursued as far as possible toward the objective of obtaining a unique, globally consistent interpretation for each region.

After all pairwise inconsistencies have been removed, additional semantic considerations are used to indicate possible segmentation errors and plausible corrections. For example, eliminating all interpretations for a region indicates that some region must be split to allow new interpretations. A uniquely identified region that is physically too small for the assigned interpretation indicates that in the absence of a reason (such as occlusion) that region must be merged with an adjacent region admitting the same interpretation. Regions that have been merged or split must be relabeled, triggering a further round of deduction.

Should any regions survive deduction and resegmentation with multiple interpretations, the analysis proceeds by assuming possible interpretations for one of the ambiguous regions and exploring the consequences of each choice. For each assumed interpretation, deduction and resegmentation are pursued to one of three conclusions:

(1) All interpretations of some region are deleted; the current hypothesis is thus inconsistent with the rest of the scene and can be abandoned.

(2) All regions are left with unique interpretations; a consistent interpretation of the whole scene has thus been found.

(3) At least one region is still left with multiple interpretations; the analysis again branches to explore the possible interpretations of a remaining ambiguous region.

The instantiation process described above is conceptually a tree search. The goal of this search is to determine the set of instantiations for the ambiguous regions that provides the best global interpretation of the scene. Traditional search heuristics can be employed in selecting which region to instantiate and the order in which its various interpretations are explored, so as to find the "best" global interpretation of the scene with the least amount of searching.

The above system differs from previously proposed scene interpretation schemes in two essential ways. First, pairwise contextual constraints are used to eliminate incompatible interpretations prior to any search. As a result the search algorithm avoids stumbling on the same inconsistencies repeatedly, as in a backtrack search. When an interpretation is rejected, this may remove the contextual support of other interpretations, requiring their deletion. The consequences of a deletion are propagated to achieve an exponential reduction in search space.

Secondly, segmentation and interpretation are integrated by using semantic considerations to indicate segmentation errors and plausible corrections. Inconsistent interpretations are eliminated when regions are merged and additional interpretations are admitted when a region is split. The decision to merge or split regions is considered as an alternative to instantiation when deduction terminates without a unique global scene interpretation.

The above system is being implemented as a network of demons representing regions, interpretations, and constraints. The demons are interconnected through a global data base so that the deletion or reevaluation of any interpretation will automatically cause all contextually related interpretations to be queued for reexamination. Conversely, an interpretation will only be reevaluated when the likelihood of a contextually related interpretation is questioned.

This implementation appears to be an efficient way to realize several important features of heterarchical organization. All information is accessible to all parts of the system for use in evaluating any interpretation. New elements of knowledge and new processes can be added or removed independently, without affecting the rest of the system. There is also no central strategy; control propagates throughout the network following the currently most promising line of deduction (as when pursuing the implications of a deletion as far as possible). Strategy modules could of course be coupled into the network to pass control to appropriate specialist routines.

b. Initial Segmentation

The success of the above understanding paradigm hinges on the ability to partition the scene into reasonable regions, corresponding to distinctly interpretable surfaces. This preprocessing is an expediency necessitated by time and space constraints of extant processors which preclude a detailed semantic analysis at the picture cell level.

A number of desiderata were established for a segmentation algorithm. First, the segmentation should attempt to minimize the number of regions that erroneously join distinct surfaces, as such regions cannot be reliably interpreted in the subsequent labeling phase. Moreover, this bias tends to localize any segmentation errors that do occur so that, hopefully, they can be detected and corrected on semantic grounds later in the analysis. Second, segmentation should not be based on arbitrary discontinuity thresholds. Third, segmentation decisions should capitalize on the availability of multi-sensory data.

The following algorithm was designed with these considerations in mind:

First, partition the entire picture into 15x15 cells, each being 8 picture elements (pixels) square. The total amount of discontinuity within each cell is measured by applying a simple gradient operator over the cell and counting discontinuities in brightness, hue, saturation, and particularly, range. A high level of discontinuity within a region suggests that it should be split. Divide the cell with the greatest total discontinuity into 2x2 subcells. Total the discontinuity remaining in each subcell totaled and iterate until the scene has been partitioned into about 350 cells. At this point it is assumed that most of the necessary splitting has been done, leaving the uniform areas relatively unscathed. The process next enters a merging phase in which the pair of adjacent cells with the weakest boundary are joined (i.e., those regions for which the amount of discontinuity along their common boundary is least). The merge results in a new region, and the process again iterates until the desired number of regions, say 150, is obtained.

The above algorithm was implemented and found unsatisfactory, primarily because the arbitrary quartering of cells in the splitting phase produced poor spatial quantization of edges. Specifically, it tended to fractionalize the discontinuity content of strong edges among the subcells so that the total content in each one became insignificant. As a result, substantially more than 350 regions are needed to insure few false mergers. A possible improvement would be to split cells along the best line (or simple curve) that could be fit to the discontinuity points within it. However, cells containing several edges would still present difficulties. It was consequently decided to abandon the idea of starting with an arbitrary cellular partition and, instead, to modify an available region growing algorithm to fulfill the desiderata.

The Brice and Fennema (B/F) algorithm (Ref. 21) is essentially a two pass process; first, all contiguous pixels of identical brightness are agglomerated into regions. Second, adjacent regions with weak common boundaries are merged. (Weak boundaries are those for which the ratio of [cummulative discontinuity strength/length] is below a threshold.) To use this algorithm it is necessary to establish a basis for comparing adjacent pixels in a multisensory image. Our initial approach will ignore range and compare the average intensities computed over the three color separations; i.e., $I(X,Y) = [R(Y) + G(X,Y) + B(X,Y)]/3$. The resulting averages will be quantized into 32 levels.

Thresholds will be eliminated from the second pass by iteratively joining the adjacent regions with the globally weakest common boundary until the desired minimum number of regions (i.e., 150) is obtained. A tree sort is used to update the list of relative boundary strengths efficiently.

The modified B/F algorithm is being coded in FORTRAN. The resulting region structure is thereafter processed symbolically in LISP, where it is represented in terms of atomic boundary segments (i.e., segments of contiguous boundary between two and only two regions). The regions and boundary segments are also characterized by appropriate cummulative properties (i.e., area, average brightness, length). Although the modified algorithm has not yet been tested, a similar scheme has been demonstrated successfully in outdoor scenes by Yakimovsky /Ref. 19/.

A number of improvements are planned for the basic segmentation algorithm. First, the arbitrary quantization of color data will be refined to classify picture elements into semantically meaningful categories (i.e., the characteristic colors of prominent objects, such as walls, doors, tables). In effect, segmentation will be preceded by a feature extraction phase. Second, easily distinguished regions such as floor (height = 0) will be extracted by special purpose modules before applying the general segmentation procedure. Third, regions containing large amounts of range discontinuity /Ref. 22/ will be partitioned along the best line through that region, as in the initially proposed algorithm.

c. Labeling

The objective in this first stage of interpretation is to rapidly label each region with all possible interpretations admitted by its attributes. Labels will be assigned liberally as it appears easier to contradict errors of commission than to subsequently accommodate ommissions.

In our room scene domain, there are typically a fairly small number of interpretations for each region, say 3 to 5 out of a possible 20. Hue is not a particularly useful basis of discrimination, since rooms are often color coordinated. For example, in a typical SRI office (shown later in Figure 26) the floor, door, wall, table, and chair are all colored various shades of brown; in this case saturation, though in general not as reliable a measure as hue, is about as useful. Size can rule out possibilities if a region is too big, but not if it is too small since the object may be occluded. Height turns out to be extremely valuable: objects have characteristic heights and the height extremes of a corresponding region must lie within that range; e.g., anything over 7 feet can only be a wall, but anything below 6 inches cannot be a wall since walls stop at the baseboard. Figure 25 diagrams the height ranges of objects in the room domain. Note that height uniquely constrains the interpretation of all horizontal surfaces (i.e., floor, chairseat, tabletop) providing useful nodal points in the analysis.

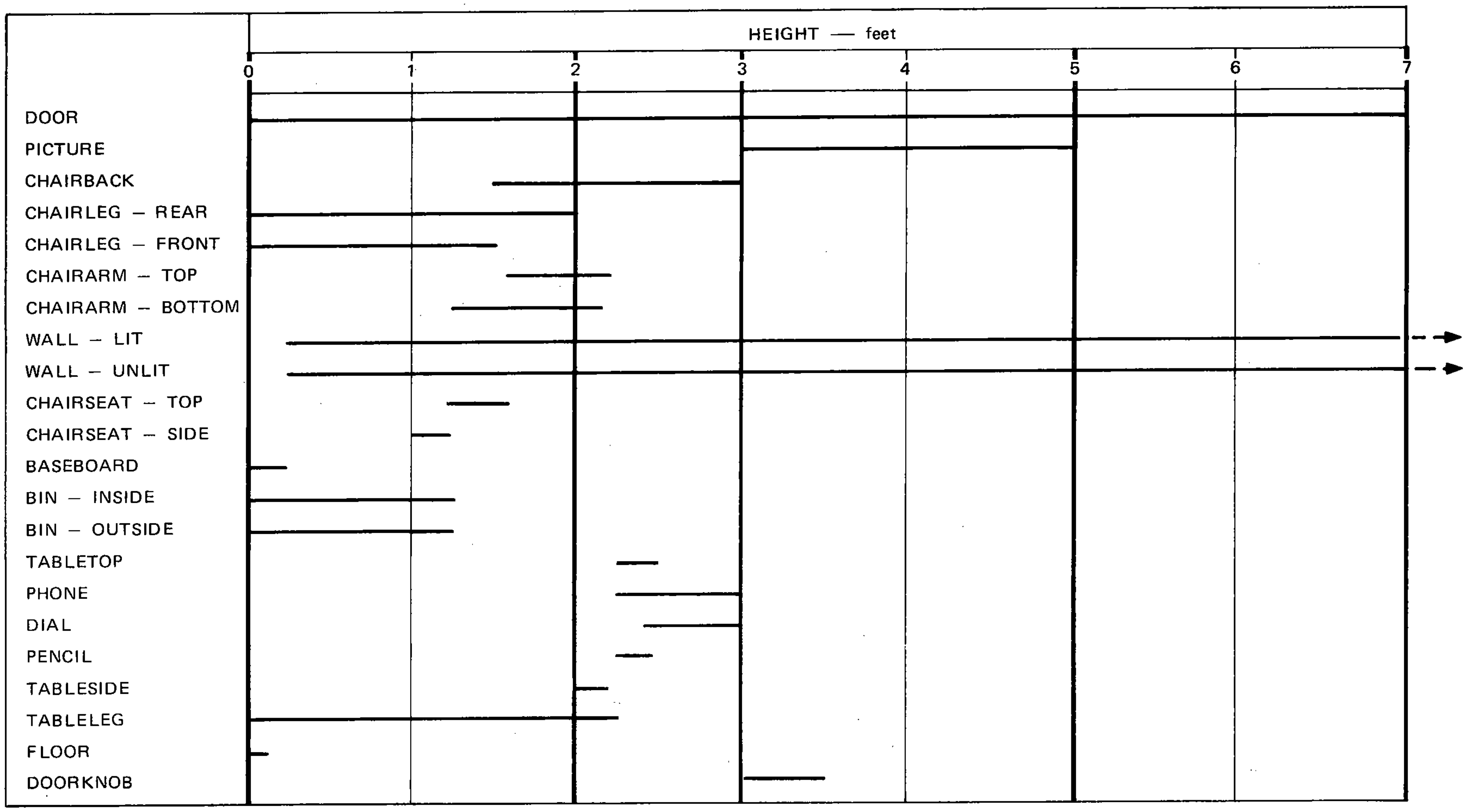


FIGURE 25 HEIGHT RANGES OF OBJECTS IN THE ROOM DOMAIN

In outdoor scenes, by contrast, hue and texture provide the most important local clues to identity. Range-based considerations such as absolute height and surface orientation are less meaningful as most everything is simply "far."

Our current approach to labeling is a two pass process, reminiscent of conventional pattern classification. Each region is first passed through a discrimination tree that disqualifies all interpretations for which a range of attribute values observed over the region exceeds the range allowed in prototype examples. Surviving interpretations are then ranked in likelihood using a recursive Bayes procedure (See Appendix B.) (This procedure computes P(I/A1 A2...An), the probability of an interpretation, given attribute values A1...An, based on empirical estimates of P(Ai /i) the probability of those attribute values for given interpretations, and P(i), the a priori likelihood that a region will take interpretation I.) The discrimination tree and likelihood data are both automatically generated from pictorial examples of interpretations designated in training scenes [Refs. 15 and 19].

A future possibility is to replace the classification procedure with a set of labeling procedures, one for each of the 20 or so principal interpretations in any given domain. Each procedure when applied to a region would attempt to disqualify its respective interpretation on the basis of distinguishing features. Labeling would then consist of applying the set of procedures to each region.

d. Deduction

The local interpretations generated during the labeling phase are interrelated by a large number of global semantic constraints. The objective now is to eliminate interpretations on semantic grounds in order to simplify the subsequent search for an optimal set of consistent interpretations.

The idea of filtering inconsistences before search was used by Fikes [Ref. 23] to simplify a variety of constraint satisfaction problems (e.g., cryptarithmetic, 15 puzzle) and, more recently, by Waltz [Ref. 20] to help interpret line drawings.

In dealing with large numbers of interpretations and constaints, a systematic method for discovering inconsistencies is essential. Our quest for efficiency led to a constraint satisfaction method based on cooperating independent processes coupled through a global data base. Briefly, the method may be described as follows.

The data base consists of variables representing constrained entities and constraints. Associated with each variable is a procedure for computing a value in terms of the current values of other variacles. Each variable also has a list of related variables whose procedures utilize the present variable as input. When the value of a variable is changed, its related variables are activated by adding their procedures to a stack of jobs to be run. Thus, if running a process changes the value of its associated variable, additional processes may be activated. Execution terminates when the job stack is empty.

An implementation of this scheme which is applicable to a variety of constraint satisfaction problems is outlined in Appendix C.

For scene interpretation, the variables are used to represent regions (i.e., boundary descriptions), region interpretations, and constraints on spatial relationships. The variables are interconnected so that the deletion or reevaluation of any interpretation will automatically cause all contextually related interpretations to be queued for reexamination. Conversely, an interpretation will only be reevaluated when the likelihood of a contextually related interpretation is questioned.

The process associated with each interpretation variable computes current likelihood as a function (e.g., the average) of its local likelihood assigned during labeling and a global confidence derived from the current likelihood values of semantically related region interpretations. In both speech [Refs. 24 and 25] and vision, it has proved difficult to develop functions that can balance diverse and often conflicting evidence to arrive at fair likelihood estimates. Our initial experiments have relied on simple functions such as the (weighted) average or minimum of global and local likelihoods. Each of these functions has inadequacies. A single inappropriate value for a must or must-not attribute should force likelihood to zero, as in the minimum. On the other hand, several strong contextual indications should counteract a single weak feature, as in the average. We are currently trying to empirically determine a composite function combining advantages of the minimum and average. The alternative to a unified scoring function is a set of ad hoc procedures that compute likelihoods based on empirical considerations peculiar to each interpretation. Two problems with this approach are, first, the significant programming effort required to perfect the procedures and, second, the difficulty of normalizing scores in order to compare competing hypothesis on an equal basis.

Processes are constructed in a generic format, as shown in Table 6, by instantiating prototype semantic constraints on an interpretation to region interpretations in the current scene. Global confidence of an interpretation is expressed as a conjunction of all constraints on that interpretation. Each constraint is represented, within the conjunction, by a disjunction of all possible ways of satisfying the constraint in the scene. A constraint on a region is satisfied by finding one or more other regions in a specified spatial relationship with the constrained region, admitting the required interpretation. The construction process is illustrated in Figure 26.

A process is represented in the data base by decomposing it hierarchically into elementary S-expressions, each canonically represented by a variable, as indicated in Figure 27. Super-expressions are placed on the related variable lists of subexpressions, and the lowest level subexpressions become related variables of any region and interpretation variables they utilize. This representation increases efficiency by allowing reevaluation to terminate at the lowest level subexpression whose value is unchanged by a triggering event. Moreover, redundant processing is avoided in cases where procedures share common subexpressions.

Constraint satisfaction is initiated by establishing a data base variable for each region interpretation found during labeling and

Table 6

GENERIC FORM OF A PROCESS FOR COMPUTING THE LIKELIHOOD OF INTERPRETATION I_A OF REGION R_A

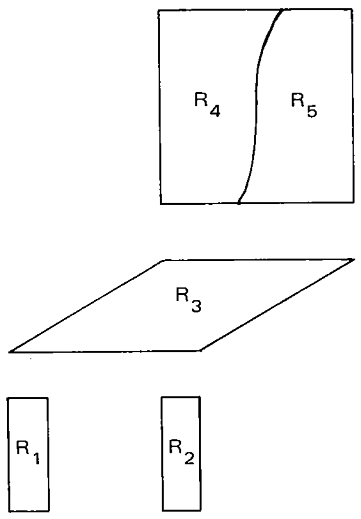
```
(I_A R_A)_L = (AVERAGE (I_A R_A)_LL
                   (AND*                                  --conjunction of all (n)
                                                            constraints on I_A
                        (OR*  (AND* (C_A1 R_A R_11)_L (I_1 R_11)_L)--disjunction
                                                                  of all (ℓ)
                                     .                    possible region inter-
                                     .                    pretations that could
                                     .                    support constraint C_A1

                              (AND* (C_A1 R_A R_1ℓ)_L (I_1 R_1ℓ)_L))
                                     .
                                     .
                                     .
                        (OR*  (AND* (C_An R_A R_n1)_L (I_n R_n1)_L)--disjunction
                                                                  of all (m)
                                     .                    possible supporting
                                     .                    interpretations for
                                     .                    constraint C_An

                              (AND* (C_An R_A R_nm)_L (I_n R_nm)_L))))
```

Notes:

$(I_i\ R_j)$	Interpretation i of region j
$(C_{Ai}\ I_A\ I_i)$	Prototype constraint on interpretation I_A; requires that I_A be in relation C_{Ai} with interpretation I_i. This prototype instantiated to the particular region interpretation $(I_A\ R_A)$ specifies that there exists a Region R with interpretation $(I_i\ R)$ such that $(C_{Ai}\ R_A\ R)$, $0 \le (C_{Ai}\ R_A\ R) \le 1$.
$(X)_L$	Current likelihood of X
$(X)_{LL}$	Local likelihood of X (i.e., likelihood assigned during labeling).
AND*	Fuzzy AND: returns minimum argument
OR*	Fuzzy OR: returns maximum argument



POSSIBLE REGION INTERPRETATIONS

(CHAIRBACK R_4) (CHAIRBACK R_5)

(WALL R_4) (WALL R_5)

(CHAIRSEAT R_3)

(CHAIRLEG R_2) (CHAIRLEG R_1)

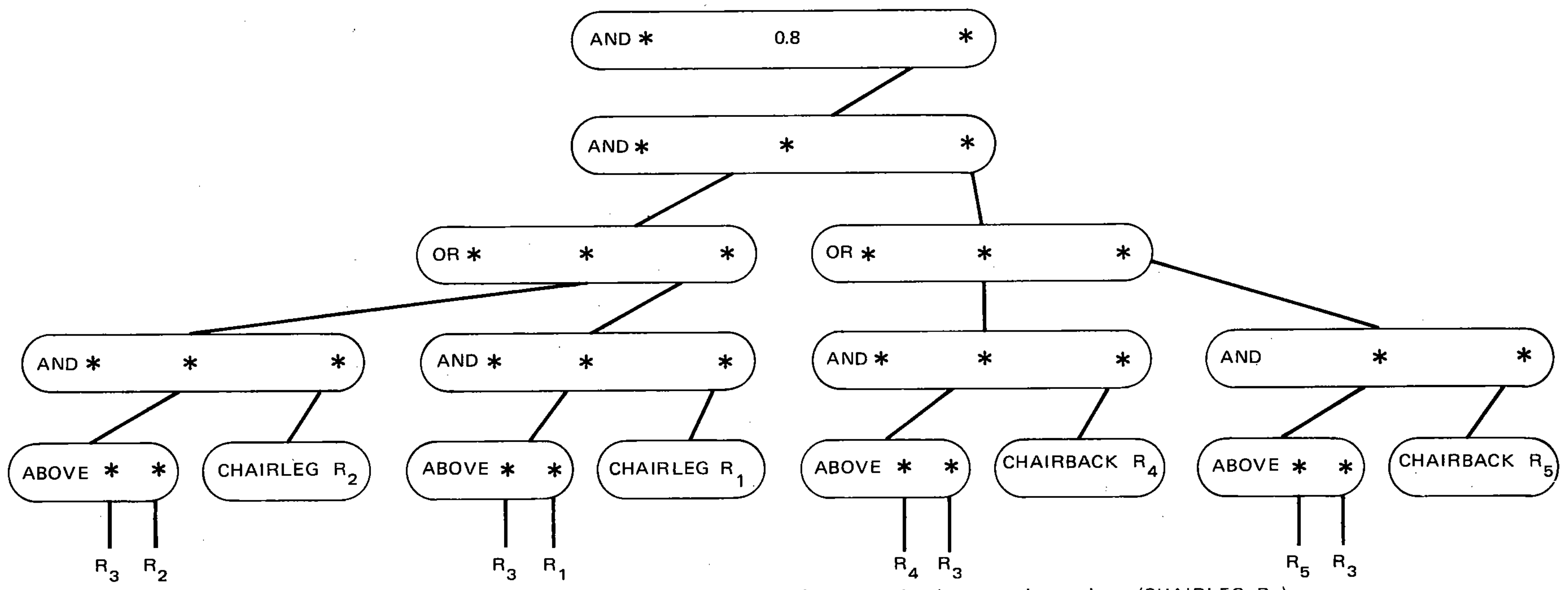
(TABLELEG R_2) (TABLELEG R_1)

CONSTRAINTS ON CHAIRSEAT
(ABOVE CHAIRBACK CHAIRSEAT)
(ABOVE CHAIRSEAT CHAIRLEG)

(CHAIRSEAT R_3) = [AVERAGE 0.8
(AND*
  (OR* (AND* (ABOVE R_3 R_2) (CHAIRLEG R_2))
       (AND* (ABOVE R_3 R_1) (CHAIRLEG R_1))
  (OR* (AND* (ABOVE R_4 R_3) (CHAIRBACK R_4))
       (AND* (ABOVE R_5 R_3) (CHAIRBACK R_5))))]

SA-1530-59

FIGURE 26 CONSTRUCTION OF PROCESS FOR COMPUTING THE LIKELIHOOD OF (CHAIRSEAT R_3)



NOTE: The likelihood of (CHAIRSEAT R_3) is only recomputed when the likelihood of a supporting interpretation such as (CHAIRLEG R_2) changes or when a spatial relationship such as (ABOVE R_3 R_2) is altered by resegmentation.

SA-1530-60

FIGURE 27 HIERARCHICAL DECOMPOSITION OF PROCESS REPRESENTING (CHAIRSEAT R_3)

preloading the job stack with corresponding setup procedures for each. Values of these variables are initialized to the local likelihood of the interpretation.

The setup procedure performs two basic functions:

(1) It constructs a function for evaluating global likelihood of the interpretations as per Table 6 and connects it into the data base as per Figure 27. Specifically the relative lists of variables used in the function are up-dated.

(2) The evaluation function is executed and the result assigned as the new likelihood value of the interpretation variable.

As a side effect of (2), all related variables that had previously been evaluated using the original likelihood of the current interpretation are added to the front of the job stack to be reevaluated. The implications of these reevluations are thus explored before running additional setup jobs. Execution finally terminates when the job stack has emptied, leaving globally consistent likelihoods for all interpretations.

The related variables concept provides an effective channel for exchanging criticism. Altering the likelihood of an interpretation as a result of applying specialized semantic knowledge amounts to criticizing the reasoning of any processes that had previously based decisions on the original likelihood value. Actuating related variables, in effect, forces those processes to reconsider in light of current knowledge.

e. Resegmentation

Interpretaton to this point has been based on a conservative initial partitioning, bound to contain errors. A refined segmentation is now sought by using semantics to indicate when and how regions should be joined or split. Resegmentation is intimately related with interpretation, since altered regions must be relabeled, leading,perhaps, to further deductions. The following ideas represent first thoughts toward a semantic basis of segmentation.

Merger Semantics

Merging may be indicated when a region is too small for one of its interpretations. Two regions can be merged only if they share a common interpretation. Thus, barring excuses such as occlusion, the fact that no adjacent region admits the oversized interpretation as a possibility is sufficient to exclude that interpretation as a possibility of the small region.

A specific merger is forced when the undersized interpretation is the sole remaining possibility for the region and there is only one adjacent region admitting that interpretation. After the merger is performed, the combined region assumes the unique common interpretation.

More generally, either the decision to merge or the region to merge with is optional: the region may have other possible interpretations, there may be several merger candidates, the interpretation may allow but not require a larger region. In these cases, mergers can be postulated in the manner that interpretations are hypothesized, by creating a subcontext and pursuing the consequences to see if contradictions arise. Possible interpretations for the merged region are obtained by intersecting the interpretation possibilities of the component regions and excluding any for which the new region is too large.

Many other semantic considerations besides size can indicate merger possibilities. Applying Occam's razor, one can argue that two adjacent regions admitting the same interpretations should always be merged unless there is a persuasive reason not to. Nonadjacent regions with the same interpretation can also be merged if it can be shown that they correspond to the same physical surface. For example, the structure of a door requires that two coplaner regions, one above the other, each interpreted as door, must be the same door. On a more global scale, knowledge that rooms have only one door or picture implies that all regions so labeled are instances of the same physical object. Isolated regions of the same object may also be linked through their dependence on a common region interpretation for satisfying a constraint. Thus, two regions interpreted as chairseat on the basis of a particular chairback interpretation probably belong to the same chair.

Partitioning Semantics

While merging regions can be considered as a refinement of a conservative initial segmentation, splitting regions is more appropriately thought of as error correction. An error is indicated whenever all possible interpretations of some region are ruled out. It is assumed that the semantics will allow a consistent interpretation for any correctly segmented scene and moreover, that any portion of a proper region will be correctly interpreted. Problems arise when a region erroneously spans two physical surfaces; the combination of attributes, or simply the large size can eliminate the correct interpretation of either surface from that region's set of possibilities. Such regions must be split along the surface boundary to admit these additional interpretations.

Determining what region to split and how to split it is substantially harder than determining what and how to merge. Unfortunately, it does not follow that the region finally deprived of interpretations is the region that needs to be split. The valid interpretation could, in fact, have been excluded by an unsatisfied constraint, because some other region failed to provide a needed support interpretation. The process of pinning down precisely which region is at fault can thus involve considerable backtracking.

Once the faulty region has been determined, there are virtually an unlimited number of ways in which it can be split. Some clues may be available from looking harder (i.e., with a lower threshold) for data discontinuities, and from boundary continuity with adjacent regions. The region may be split in two according to global procedures such as agglomerative clustering [Ref. 26] or a Hough transform of internal discontinuities [Ref. 27]. A more semantic approach might attempt to split the region so as to obtain interpretations needed for global consistency. These many alternatives require that region splitting almost always be treated as a tentative hypothesis.

f. Instantiation

After all inconsistent interpretations have been eliminated from the global context, scene interpretation reverts to a search for the set of compatable region interpretations with highest combined likelihood. Many optimal and heuristic search strategies have been studied. The following proposed strategy is an outgrowth of Duda's heuristic search procedure (Ref. 16), which utilizes a more global evaluation criterion and more global constraints to improve the order in which nodes are examined:

(1) Select the best (i.e., most likely) ambiguous interpretation in the current context and hypothesize it.

(2) Generate branches corresponding to acceptance and rejection of that hypothesis, setting up a new context for each.

(3) In the respective contexts, reevaluate all interpretations whose likelihood is directly affected by acceptance or rejection of the hypothesized interpretation; pursue the consequences of all reevaluations as far as possible short of further hypothesizing.

(4) Evaluate the global likelihood of each context, say by summing the likelihoods of the best interpretation for each region. Any context in which all possible interpretations of some region are deleted (or receive a very low likelihood) is assigned a zero global likelihood.

(5) If all regions in either context are assigned unique interpretations, terminate and return that context as the best scene interpretation. If all regions in both contexts are uniquely interpreted, return the context with the highest global likelihood.

(6) Select the globally best context and go back to (1) above.

The above procedure is distinguished from conventional breadth first search algorithms by the use of pairwise constraint satisfaction in step (3). As in the initial global context, interpretations that are inconsistent with each hypothesized context are eliminated before any further search. Any context in which all interpretations of some region are eliminated is assigned a zero global likelihood and thus effectively excluded from further exploration. Even when interpretations are not eliminated, the application of pairwise constraints improves estimates of interpretation likelihoods, thereby improving the order of search and minimizing backtracking.

The following factors seem relevant as criteria for selecting the best hypothesis within a context:

(a) Select the ambiguous interpretation of highest absolute likelihood.

(b) Select the best interpretation of the region for which the likelihood ratio of best to next best interpretation is maximal.

(c) Select the best interpretation of the ambiguous region with fewest alternatives.

(d) Select the interpretation most semantically interrelated with other interpretations.

We have also considered an entropy type measure that encompasses factors (a), (b), and (c). The most effective criteria will probably be an empirically determined combination of factors like the above.

Some form of backtracking mechanism is needed to implement the breadth first search. The procedure, as described, can be implemented most simply by modifying the existing job stack to run as a priority queue (ordered by global likelihood) and by providing a means for preserving states of the data base. The QLISP context mechanism wherein values of data base expressions are stored by context on a property list seems appropriate.

A more sophisticated implementation of our basic search procedure would allow context evaluation in step 3 to be suspended as soon as the cumulative reductions in likelihood reduced the context's desireability below that of the previously best context. Control would then jump from context to context, always pursuing the globally best. This refinement would require primarily that the stack of jobs concerned with the reevaluation of a context at the time of suspension be stored with the job representing that context on the top level priority queue.

4. Scenario

Before attempting to implement the above scene understanding system we thought it advisable to test the design philosophy by undertaking a detailed manual analysis of a simple but representative room scene, which is shown in Figure 28. In the process we hoped to obtain some preliminary answers to the critical questions posed in the introduction: what types of knowledge, what representations, what depth of reasoning is needed to interpret an image of a room.

a. Description of Scenario

The scene was first partitioned into about 50 regions, as shown in Figure 29. This segmentation presumed a conservative merging algorithm that might occasionally split a single surface into two regions but would almost never falsely join two surfaces into a single region. (Erroneous merging is considered undesireable because the resulting combination of local attributes could preclude correct interpretation.) Each region in the partitioned scene was assigned all possible interpretations admitted by local attributes. The incorrect (i.e., globally inconsistent) interpretations were then eliminated using empirically determined semantic constraints.

b. Discussion of Results

Most regions were assigned from three to five interpretations out of a universe of 20; these interpretations are shown in Table 7. Horizontal surfaces (i.e., floor, chairseat, tabletop) are uniquely

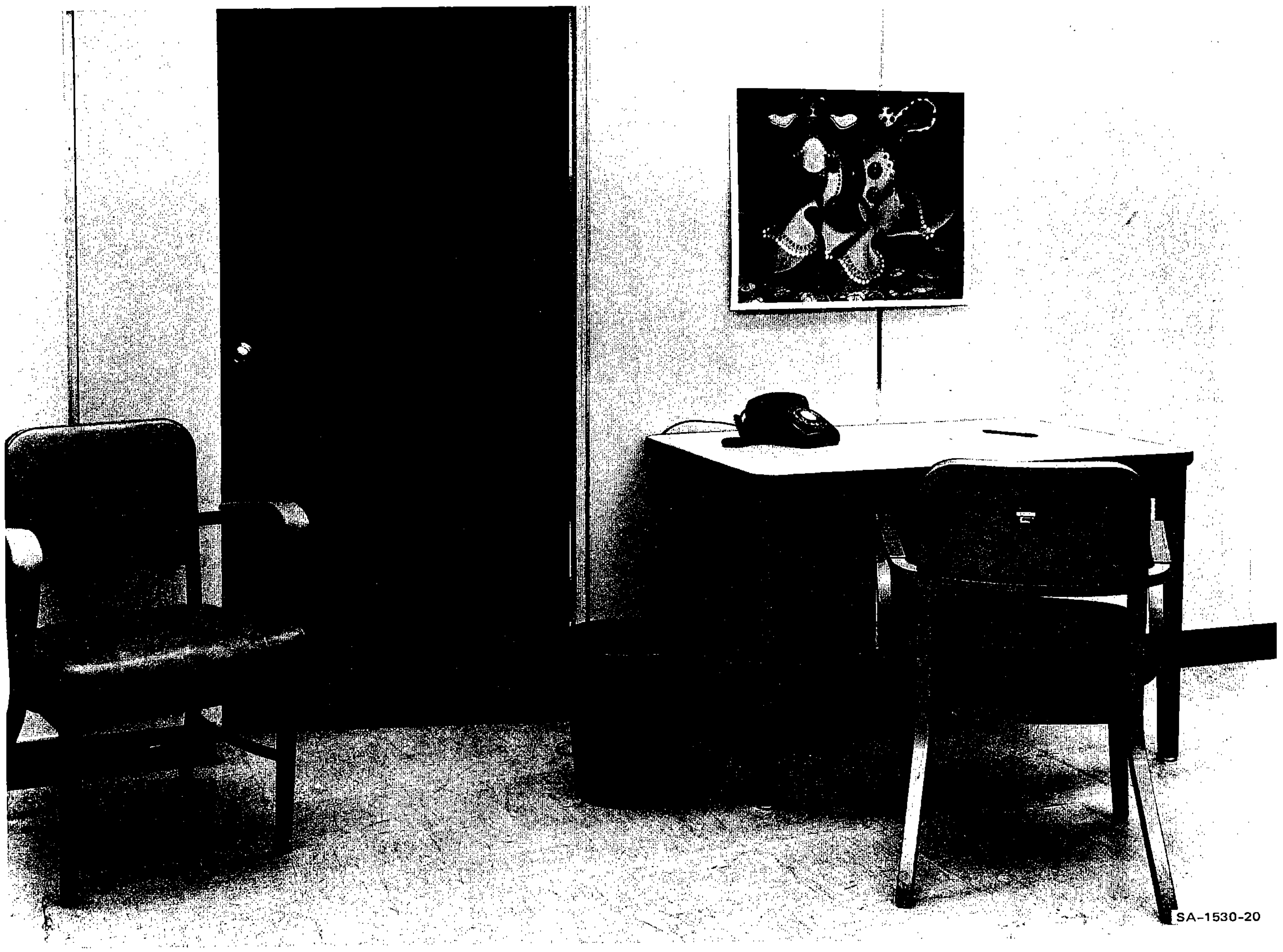


FIGURE 28 A SIMPLIFIED OFFICE ENVIRONMENT REPRESENTING OUR EXPERIMENTAL DOMAIN

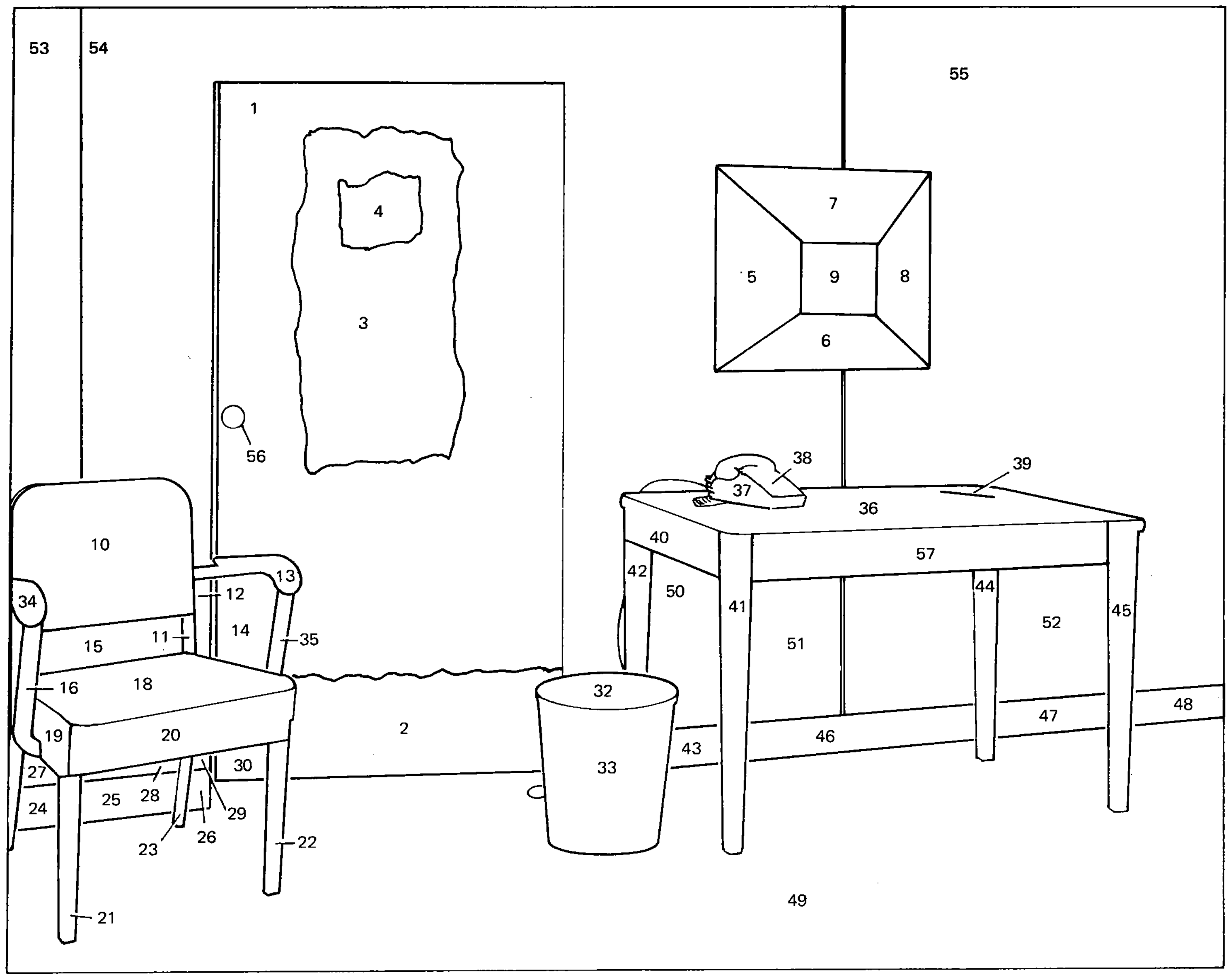


FIGURE 29 PARTITIONED ROOM SCENE FOR SCENARIO

Table 7

REGION INTERPRETATIONS FOR SCENARIO

Region Number	Correct Interpretation	Other Locally Possible Interpretations
1	DOOR	WALL-LIT, WALL-UNLIT
2	DOOR	WALL (legs and bin, excluded by horizontal extent)
3	DOOR	PICTURE, WALL-LIT, WALL-UNLIT, DOOR
4	DOOR	PICTURE, WALL-LIT, WALL-UNLIT, DOOR
5	PICTURE	PICTURE, WALL-LIT, WALL-UNLIT, DOOR
6	PICTURE	PICTURE (color blue excludes WALL, DOOR)
7	PICTURE	PICTURE, WALL-LIT, WALL-UNLIT, DOOR
8	PICTURE	PICTURE, WALL-LIT, WALL-UNLIT, DOOR
9	PICTURE	PICTURE, WALL-LIT, WALL-UNLIT, DOOR
10	CHAIRBACK	DOOR, WALL-LIT, WALL-UNLIT (unambiguously CHAIRBACK for green chairs)
11	CHAIRLEG-BACK	TABLELEG, BIN-IN, BIN-OUT, DOOR (WALL excluded since height = 0)
12	WALL	DOOR, CHAIRLEG, CHAIRARM-BOTTOM, TABLELEG, WALL-UNLIT
13	CHAIRARM-TOP	
14	DOOR	(too low for CHAIRBACK, too wide for arms, legs)
15	WALL-UNLIT	DOOR (too wide for arms, legs)
16	CHAIRARM-BOTTOM	DOOR, TABLELEG, CHAIRLEG, WALL-LIT
17	CHAIRLEG	DOOR, WALL-LIT, WALL-UNLIT, BIN-IN, BIN-OUT, TABLELEG
18	CHAIRSEAT-TOP	
19	CHAIRSEAT-SIDE	DOOR
20	CHAIRSEAT-SIDE	DOOR
21	CHAIRLEG-FRONT	TABLELEG, DOOR, BIN-IN, BIN-OUT, CHAIRLEG-BACK
22	CHAIRLEG-FRONT	TABLELEG, DOOR, BIN-IN, BIN-OUT, CHAIRLEG-BACK
23	CHAIRLEG	TABLELEG, DOOR, BIN-IN, BIN-OUT, CHAIRLEG-BACK
24	BASEBOARD-UNLIT	DOOR, BIN-IN, BIN-OUT, BASEBOARD-UNLIT
25	BASEBOARD-UNLIT	
26	BASEBOARD-UNLIT	DOOR, BIN-IN, BIN-OUT, CHAIRLEG, TABLELEG
27	WALL-UNLIT	DOOR (too wide for legs), BIN-IN, BIN-OUT
28	WALL-UNLIT	DOOR, WALL-LIT, BIN-IN, BIN-OUT
29	WALL-LIT	DOOR, CHAIRLEG, WALL-UNLIT, TABLELEG

Table 7 (Concluded)

Region Number	Correct Interpretation	Other Locally Possible Interpretations
30	DOOR	BASEBOARD-LIT, BASEBOARD-UNLIT, BIN-IN, BIN-OUT
31	BORDER	
32	BIN-IN	DOOR (nonplanar), WALL-LIT, WALL-UNLIT, BIN-OUT
33	BIN-OUT	DOOR, BIN-IN
34	CHAIRARM-TOP	
35	CHAIRARM-BOTTOM	DOOR, TABLELEG, CHAIRLEG, WALL-LIT
36	TABLETOP	
37	PHONE	DOOR, CHAIRBACK (could eliminate both by testing for nonplanarity but expensive)
38	DIAL	(orientation and height are unique)
39	PENCIL	TELEPHONE, TABLETOP, DOOR, WALL (too small for reliable interpretation)
40	TABLESIDE	DOOR (too wide for CHAIRBACK), WALL-LIT, WALL-UNLIT
41	TABLELEG	DOOR, CHAIRLEG-BACK
42	TABLELEG	DOOR, CHAIRLEG-BACK, WALL
43	TABLELEG	DOOR, CHAIRLEG-BACK
44	TABLELEG	DOOR, CHAIRLEG-BACK, CHAIRLEG-FRONT
45	BASEBOARD-UNLIT	DOOR, BIN-IN, BIN-OUT, BASEBOARD-UNLIT
46	BASEBOARD-UNLIT	DOOR, BASEBOARD-LIT
47	BASEBOARD-UNLIT	DOOR, BIN-IN, BIN-OUT, BASEBOARD-LIT
48	BASEBOARD-LIT	DOOR, BIN-IN, BIN-OUT, BASEBOARD-UNLIT
49	FLOOR	
50	WALL-UNLIT	DOOR, WALL-LIT
51	WALL-UNLIT	DOOR, WALL-LIT
52	WALL-UNLIT	DOOR, WALL-LIT
53	WALL	
54	WALL	
55	WALL	
56	DOOR KNOB	DOOR, PICTURE
57	TABLESIDE	WALL (too wide for DOOR)

distinguished by height and orientation and thus are particularly valuable in constraining semantically related interpretations of other regions.

Table 6 is a representative list of semantic constraints, largely ad hoc and certainly not unique. A sufficient set was deduced in a few hours of Gedanken reasoning, and alternatives are easily found. The generality of a particular set of constraints is best determined in the context of a functioning scene interpretation system.

The most frequently invoked constraints entailed vertical ordering (e.g., above), adjacency, coplanarity, and homogeneity (of hue, brightness, surface orientation, and so on). Each of these constraints can be represented by relatively simple computational procedures. For example, vertical ordering relationships between two regions can be defined in terms of the relative vertex coordinates of their bounding rectangles. Homogeneity requires that no adjacent region with the same interpretation have a significantly different value for the constrained attribute.

Constraints can be usefully categorized as vertical or horizontal. Vertical constraints deal with interpretations of one region. For example "this region can't be part of a door because it is above a region known to be part of a wall." Horizontal constraints, on the other hand, deal with regions sharing an interpretation (or class of interpretations). For example, "form a global consensus of wall location based on all regions with Wall as a possible interpretation. Then, rule out Wall as a possible interpretation from all regions that are not coplanar with this consensus" or "if several proximate

Table 8

SEMANTICS

1.	(ABOVE TABLETOP TABLELEG)	
2.	(ABOVE CHAIRSEAT CHAIRLEG)	
3.	(ABOVE CHAIRBACK CHAIRSEAT)	
4.	(V-COPLANAR WALL) (V-COPLANAR DOOR)	
5.	(NOT (COPLANAR CHAIRBACK WALL))..............	(V-COPLANAR X) → a region pictorially above or below a region uniquely labeled X and not coplanar with that region cannot itself be X.
6.	(NOT (ABOVE WALL DOOR))	
7.	(NOT (COPLANAR CHAIRBACK DOOR))	
8.	(ADJACENT DOORKNOB DOOR)	
9.	(OR (ADJACENT PICTURE WALL) (ADJACENT PICTURE PICTURE)...............	an interpretation can be legally adjacent to itself due to conservative merging.
10.	(SURROUND PICTURE WALL)......................	(SURROUND A B) → A adjacent on all sides to B.
11.	(HOMOGENEOUS WALL)..........................	(HOMOGENEOUS X) = A region that is adjacent to a region uniquely labeled X and with significantly different brightness, hue, or saturation cannot itself be labeled X.
12.	HOMOGENEOUS DOOR	
13.	(NOT (ABOVE DOOR BASEBOARD))	
14.	(ON TELEPHONE TABLE)	
15.	(ON PENCIL TABLE)	
16.	(NOT (COPLANAR BIN-OUT WALL))	
17.	(NOT (ADJACENT BIN-IN FLOOR))	
18.	(ABOVE CHAIRBACK CHAIRLEG-REAR)	
19.	(ADJACENT CHAIRARM-FRONT CHAIRARM-TOP)	
20.	(INSTANCES DOOR 1)..........................	(INSTANCES A n) → n instances of object A can be expected in any one scene.
21.	(PROXIMAL CHAIRSEAT CHAIRBACK CHAIRLEG)......	(PROXIMAL A B ... Z) → Objects A, B, ... Z will be found in a common vicinity of the scene.
22.	(PROXIMAL TABLETOP TABLESIDE TABLELEG)	

regions have been identified as part of a chair, activate an expert chair procedure to identify uniquely other regions corresponding to the remaining parts." Horizontal constraints thus simplify many regions simultaneously. Within the above categories, constraints can be identified as absolute or preferential, depending on whether they positively eliminate an interpretation from further consideration or merely reduce its likelihood.

Constraints can also be classified on the basis of how they are used in the reasoning process. Some constraints were primarily used for disqualifying interpretations. For instance (ABOVE CHAIRBACK CHAIRSEAT) was literally interpreted as "to establish whether a region could be a chair seat, check whether any region above it could be a chairback."

Other constraints were most useful for propagating the consequences of an interpretation once it had been established. For instance, after determining that a given region was floor, the constraint (NOT (Adjacent FLOOR WALL)) was invoked to delete wall as a possible interpretation of all adjacent regions. These two categories of usage are related respectively to consequent and antecedent reasoning.

Most negated constraints are in the antecedent category because they are not effective until some region has been uniquely assigned a constrained interpretation. (Negated constraints can however be used in consequent reasoning by attaching demons to regions having a constrained interpretation as a possibility. These demons would activate the contraint when and if all other interpretations of the region are deleted.)

Many horizontal constraints are also effective only when an antecedent condition has been satisfied. For instance, once a region has been uniquely identified as a door, knowledge that an office has only one door can be invoked to eliminate that interpretation from all noncoplanar regions.

Another use of antecedent reasoning was to hypothesize each alternative interpretation of a region in turn, amd propagate all applicable antecedent constraints. All hypotheses that led to contradictions were eliminated.

Deep chains of deduction were not required to interpret the scenario. Most of the reasoning was similar to that used in the introductory example to disambiguate wall and door. For instance, the picture on the wall (regions 5-9 in Figure 29) was analyzed by deducing first that regions 5, 7, and 8 could not be wall because of hue and saturation discontinuities with regions 54 and 55 known to be wall. Moreover, these regions could not be door because of their proximity to region 6 known to be picture. Region 9 was then constrained to be part of the picture to which the surrounding regions belonged. (Note the value of picture frames which clearly delineate picture-wall boundaries in this kind of analysis.) Legs belonging to tables and chairs were differentiated on the basis of respective proximities with regions 18 (chairseat) and 36 (tabletop).

One of the more interesting deductions was that used in eliminating waste basket as a possible interpretation of small baseboard regions (i.e., 24, 26, 45, 47, and 48). It was first reasoned that a region such as 26 could not be the outside of a wastebasket because it was coplanar with region 54 known to be a wall. (Only the inside of a basket up against a wall would be indistinguishable from the plane of that wall.) On the other hand, region 26 could not be the inside of the basket (viewed from above) because it was adjacent below to the floor. (Only the outside of a basket can share a bottom border with the floor.) Thus, while region 26 did not violate any global constraints for the interpretation "waste basket against a wall," since it could be neither the inside nor outside of such a basket that interpretation was disallowed. The above reasoning confirms an observation made by Waltz /Ref. 20/ in the blocks world, that more specific local interpretations (e.g., distinguishing between inside and outside) lead to more constrained global interpretations.

One of the most encouraging observations to result from the scenario is that multisensory room scenes are highly constrained and should prove much easier to analyze by constraint satisfaction methods than the abstract blocks world. By comparison, the number of interpretations initially possible for each region (3-5) is far less than the average number of local vertex interpretations encountered by Waltz (>50). Moreover, the contextual constraints in real world scenes are infinitely richer than the extremely local restrictions that can be applied to randomly arranged polyhedra. (Waltz relied solely on the requirement that interpretations assigned to adjacent vertices be consistent with respect to the type of edge joining them.)

5. Current Status and Plans

The components discussed above are curently being integrated into a complete scene understanding system. The basic deduction subsystem that evaluates the global consistency of possible interpretations has been programmed and tested on symbolic data. This subsystem is currently being interfaced with ISIS [Ref. 15] so that experiments can proceed on real pictorial data. A labeling program and procedural representations of the most common constraints are under development. Manually partitioned images will be utilized for initial debugging, while the segmentation algorithm is being perfected. When the core system is able to handle simple scenes, modules for semantic resegmentation and instantiation will be added.

In the near term, the scene interpretation system will be used to determine empirically what semantic knowledge is necessary to understand various pictorial domains. We are especially interested in learning how much general knowledge carries over between domains. Our system allows the contribution to understanding of each piece of knowledge to be clearly evaluated.

We envisage a system that will accumulate knowledge about room scenes incrementally, by appealing to a human tutor when stuck (i.e., when all known interpretations of some region are eliminated, or ambiguities cannot be resolved). The tutor will be able to advise the system by providing additional region interpretations, new semantic constraints, refinements to existing constraints, and so forth. This paradigm of knowledge acquisition makes more sense than attempting to anticipate, a priori, a body of semantic knowledge that is necessary and/or sufficient for interpretating a given domain.

In related work, supported by NASA, we are building a facility for interactive versus automatic scene analysis. The user of this system will not only provide new knowledge when the system gets stuck, but will take an active role in directing the analysis. For example, the user could provide a crude verbal description of the scene, thereby indicating relevant interpretations and constraints. He could even directly designate the desired interpretation for a particular region. The consequences of such an assignment might then propagate to influence the global interpretation. The system's reasoning process will be interruptable at any point, to query the reasons for discarding an interpretation, and to introduce or eliminate constraints, interpretations, region boundaries, and so forth. Assimilating arbitrary new knowledge into an ongoing analysis is a complicated control problem; the system will have to leave notes in the data base explaining the reasoning behind every action so that decisions can be undone should subsequent events counteract those reasons.

In the longer term, the understanding paradigm will be augmented with additional analysis techniques to achieve more flexibility in the use of knowledge. A uniform constraint satisfaction procedure requiring exhaustive segmentation and enumeration of possibilities seems inappropriate in very complex scenes, especially when a complete scene description is not necessary. In such cases, it would be preferable to sample the image for uniquely identifiable surfaces and then initiate independent processes to confirm each of these interpretations. These processes would in turn spawn further processes to seek supporting interpretations and so forth. Region boundaries would be determined top down after recognition has been completed. The related variable concept could coordinate the analysis by communicating criticism and detecting contradictions. Conflicting interpretations proposed by independent processes might be resolved by invoking higher level semantic considerations or by initiating a process to discriminate on the basis of distinguishing features. A complementary refinement, when specific information goals exist, is to restrict the detailed semantic analysis to relevant parts of the scene localized by a rapid distinguishing features search.

Another unsatisfying restriction of the present implementation is that all knowledge must be represented at a "surface" level. That is, a constraint that chairs are near tables would have to be represented as (NEAR CHAIPSEAT TABLETOP). As pointed out earlier, scene interpretation should utilize semantic constraints at many levels, including relations between parts of an object, relations between objects, global domain constraints (e.g., constraints associated with indoor scenes), and functional constraints on form and structure. It is not feasible, as Charniak [Ref. 28] discovered, to have all semantic knowledge simultaneously available. Relevant constraints must be activated as the context emerges during interpretation. Multiple levels of knowledge could be accommodated in the present constraint satisfier by using the related variable mechanism to activate specialist routines when sufficient evidence had accumulated to indicate their relevance. Different levels of knowledge are more easily accommodated in the proposed independent process model since the use of knowledge is explicit in a procedural representation.

D. Vision Support Tasks

1. Introduction

The principal objectives in this first year were to determine the visual information requirements of a computer consultant in order to formulate an appropriate research plan. The requirements were determined by studying protocols between a human consultant and an apprentice. Table 9 contains annotated excerpts from the experimental dialogs indicating a need for vision. Vision is also required when designating parts and tools by pointing at them.

In analyzing these protocols, we conclude that basic perceptual strategies developed for robots can also be utilized by a computer consultant. Both must be able to select from a large repertoire of specialized techniques those best suited for obtaining a particular piece of information. The consultant, however, has the big advantage of including in its repertoire the perceptual skills of the apprentice. Many difficult-to-automate perceptual determinations, such as whether a belt is worn, are easy for humans to judge at a glance. In such cases, the consultant's best perceptual strategy is simply to ask the apprentice.

One difference between compressor scenes and room scenes is that tools and compressor parts are distinguished primarily by structured shape descriptions. A critical research objective is thus to develop adequate representations of curved three-dimensional objects. We do not necessarily envisage a single comprehensive formalism for

Table 9

PROTOCOL EXCERPTS ENTAILING VISION

A = Apprentice E = Expert

1. "Show me the belt please." Make sure A had the right belt.

2. "Show me the 1/2-inch combination wrench please." "Show me the tool you are using." Check to see if A was using the correct tool.

3. "Show me what you are doing." Verify hypothesis that A was trying to fit wrong wheel.

4. "Point at the screws you are going to remove." Verify correct intent.

5. "Show me what you are doing please." A was taking a long time. E wanted to see if A had air line disconnected yet or had gone astray.

6. "Show me the after cooler elbow." A was having trouble connecting the after cooler to the elbow. E wanted to check whether the after cooler elbow and the pump housing to which it was attached had been correctly assembled.

7. "Let me see it." Check alignment of grooves in pulley and flywheel.

8. "Show me what you are doing." A had gone ahead and E had to update his assembly model.

9. "Show me the pressure gauge." Check for correct operation.

describing shape but will continue throughout the course of this project to develop representations for more and more complex structures. Desirable characteristics of representation are the following: They should be capable of representing an arbitrary degree of detail without making the retrieval of the coarser features unwieldly; they should be capable of relating to models of surface characteristics, including reflectivity, color, and texture; they should be capable of characterizing relationships among parts in an assembly; above all, they should enable the system to produce (understand) descriptions easily understood (produced) by humans.

Initially we will concentrate on representing a fixed set of specific parts. These representations will be "model-based", in the sense that they refer to the precise geometric details of specific prototype parts. As the research proceeds further, we hope to develop functional semantics that can be used to recognize parts generically. Such representations would give the consultant general expertise about a class of equipment.

The algorithms by which the appropriate representations are abstracted from visual and range data are an integral part of the research on representation. In the early stages of the program, recognition strategies for compressor parts will be hand-coded into the system, based on "distinguishing features," a set of tests that may economically distinguish among parts in a limited context. Later, we hope to be able to interactively "teach" the system which features or characteristics of an object may be reliably used for discrimination.

Specific progress has been made in two areas of research: spatial representation of compressor parts, and the low-level processing of range data. These are described more fully below:

2. Representation of Compressor Parts

Work has begun on specification of the data structure and algorithms needed to point to and/or identify parts of an assembled compressor. The emphasis in this task is on spatial modeling of an assembly, rather than on extracting information from an image.

The compressor model will contain the location, extent, and shape of each part. Based on this model, the system should be capable of: (1) identifying any part of the compressor designated by manually pointing the laser and (2) pointing the laser at any named part. Identfication entails finding where the laser ray intersects the model, while pointing involves aiming the laser at the centroid of the visible portion of the desired model part. Several algorithms are under study for outlining the visible portion of an object. The initial specification and subsequent modification of the model of the compressor will be done with interactive graphics, possibly utilizing range images of actual objects.

Initially, the extent and shape of parts will be represented by plane-faced polyhedra. Later descriptions will be more flexible and feature-oriented.

3. Acquisition, Display, Transformation, and Display of Range Data

The computer consultant will depend heavily on a laser range finder for obtaining three-dimensional shape and orientation of parts and assemblies. Three-dimensional data have been obtained from the laser range finder in usable form. A mirror scans the laser beam in a raster pattern, under computer control. Phase measurements on the reflected light are digitized, converted to range, and stored on a disk file. (The range finder system is described in Section V of this report.)

The data may be displayed in perspective, under interactive user control. By varying the perspective in small increments, "motion stereo" may be simulated to give the illusion of depth. A split-screen viewer is being built to enable stereo presentation.

Calibration of the range finder entails two distinct phases: determination of the laser's orientation, deflection coefficients, and scale factors; and determination of the correspondence between TV images and range images of the same scene.

Calibration of the laser's own parameters is currently done manually using a ruler. Since none of these parameters are stable, calibration must be performed each time range measurements are to be made. Automatic and interactive calibration methods are under investigation.

The correspondence between TV and range images is computed by maximizing the correlation between discontinuities in the two images. The two images are first brought into rough correspondence interactively, using the perspective display program. A hill-climbing algorithm then maximizes the correlation.

V THE SCANNING-LASER RANGEFINDER

A. Introduction

In a previous report (Ref. 1) a basic description was given of a range scanner capable of generating a "picture" analogous to a television picture, but in which each picture point was associated with an analog value denoting range rather than light intensity.

A simplified block diagram of the range scanner is given in Figure 30 and a more detailed diagram in Figure 31. The range scanner consists of three functional components:

+ A transmitter that amplitude modulates a light beam with a 9-MHz sine wave

+ A mirror scanner that sweeps the modulated beam over the field of interest

+ A receiver that picks up the light from whatever object intercepts the scanning beam. After amplification the output from the receiver is demodulated by a phase demodulator.

The demodulator effectively measures the length of time required for a light beam to make the round trip from range scanner to light

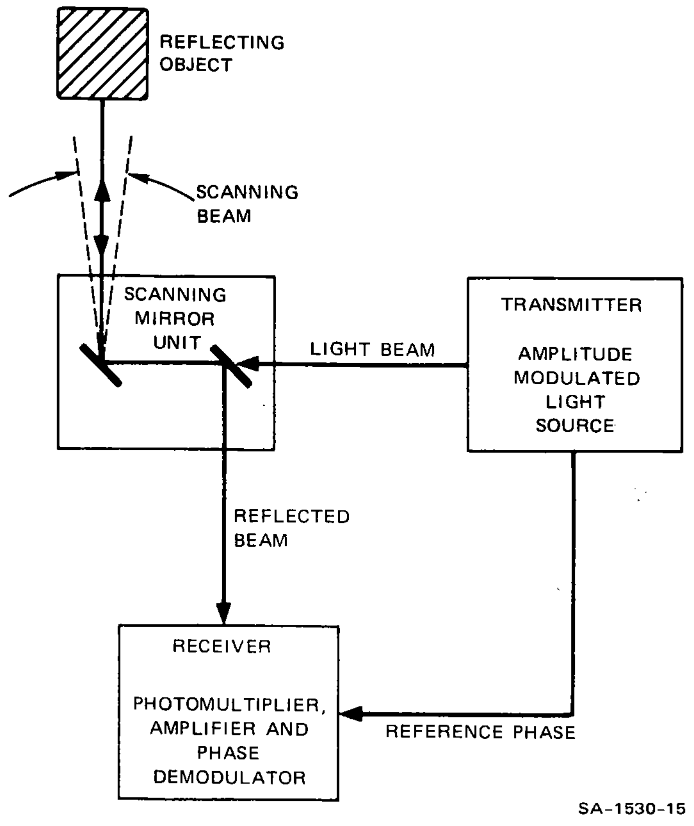


FIGURE 30 SCANNING RANGE FINDER—SIMPLIFIED

scattering object and back. The range value, of course, is a constant multiple of this time.

The merits of the phase modulator are worth emphasizing. In this device the output from the receiver is compared with the phase of the modulating signal feeding the transmitter, and a dc level is derived that is a function of this phase difference only. Signals of other frequencies average to zero. Since the phase difference varies linearly with the distance to the light scattering object, the dc level is a measure of the range of the object. The method has the virtue of being a one-frequency method not depending on the amplitude

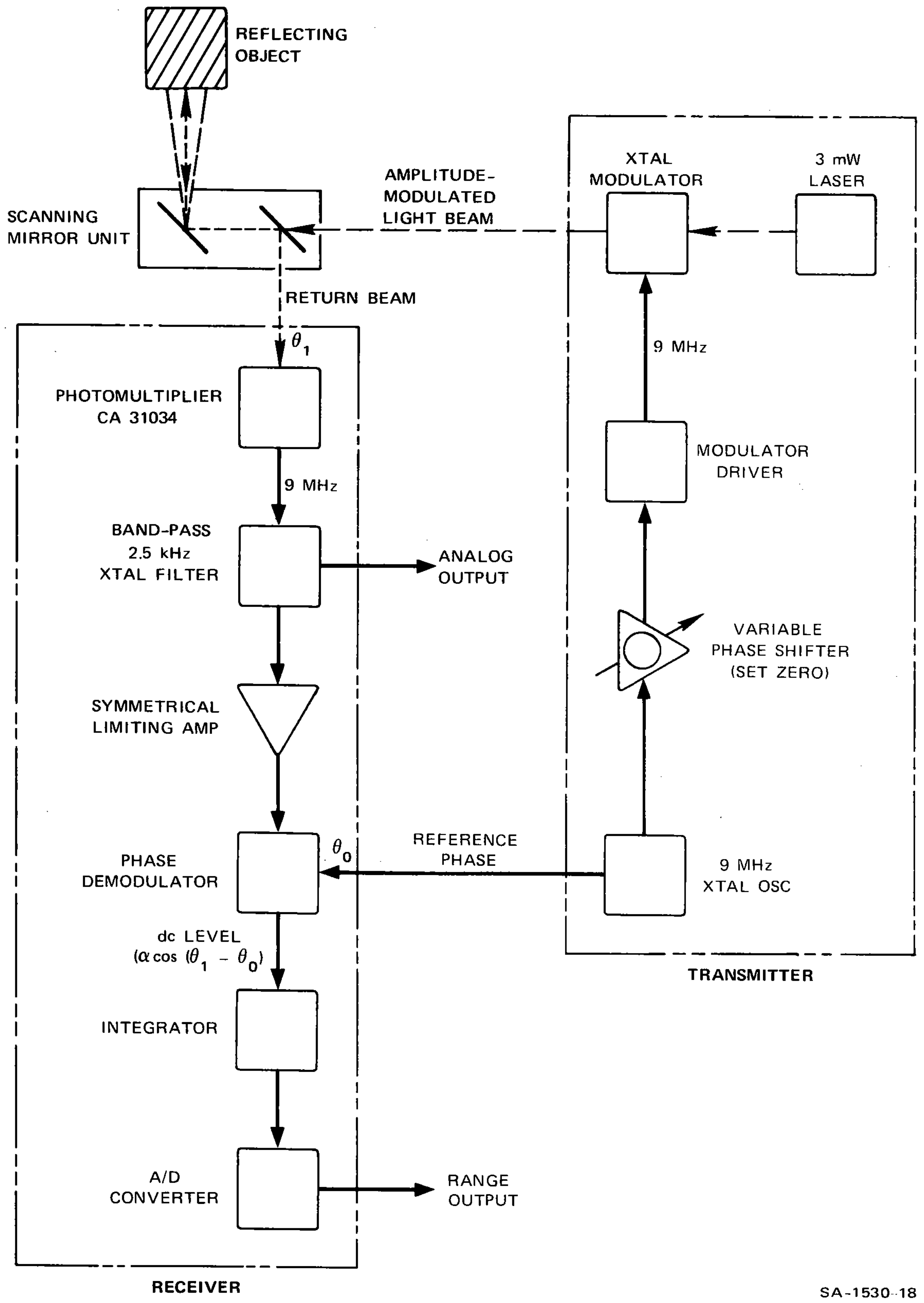


FIGURE 31 SCANNING RANGE FINDER—DETAILED

of the received signal; it is therefore possible to trade range discrimination against signal integration time. The sensitivity calculations presented later indicate that, at a range of 10 feet, it may be possible to make 1,000 measurements per second with a range discrimination of 0.25 inch.

Considering the transmitter section, the process begins with the 9-MHz crystal oscillator that drives the modulator via an adjustable phase shifter. The phase shifter provides an adjustment of approximately 60 degrees so that one may adjust for a precise zero or, alternatively, offset the minimum range and use increased sensitivity in the read-out. The zero may also be displaced by fixed amounts and with great precision by short lengths of coaxial cable inserted between the various units. The modulator provides amplitude modulation of the light source, which may be either a 9050-a.u. gallium arsenide diode or a 6328-a.u. helium-neon laser. The latter is preferable since it can provide a smaller spot size (better lateral resolution). Although gas lasers typically show 1 percent amplitude noise modulation, this does not interfere, except to a minor extent, with the precision of the phase measurement. The modulated light beam is then deflected by a scanning mirror system (in two orthogonal directions) so that it scans the field of view that is of interest.

The receiver section makes use of the same scanning unit so that its line of sight coincides with that of the laser beam. The return signal is put through an interference filter to eliminate the ambient illumination and passes to a high-grade photomultiplier, an RCA CA 31034, where it is converted to electrical form. The use of a premium quality photomultiplier is warranted since it is here that the signal-to-noise ratio of the system is determined. The primary requirements are for an efficient photocathode and low dark current. It is desirable that the signal level from the CA 31034 should be at least 1 millivolt. This signal is then put through a 2.5-kHz wide crystal filter (form factor 2:1) at the first opportunity in order to minimize effects due to electrical interference, in particular before any saturation can occur.* The filter is followed by a symmetrical limiter to remove amplitude variations due to inverse square law, orientation of the surface, reflection coefficient, laser noise, and so on. Note that the limiter must limit the top and bottom halves of the sine wave symmetrically since asymmetry usually introduces a shift in phase.

Perhaps the most stringent part of the design occurs in the phase demodulator. What is required is a clean multiplier. We wish to multiply together the reference signal, $e_1 \cdot \sin\ \omega t$; and the reflected signal that has passed down the receiver chain, $e_2 \cdot \sin(\omega t + \emptyset)$. We do not have a great deal of control over the various phase shifts that occur down the receiver chain, but we assume they can be nulled out by the phase shifter mentioned previously. We are interested in developing a dc level that varies linearly with the variable part of $\emptyset$; i.e., the part due to the variable path length between the deflecting mirrors and the reflecting object. The output, M, from the multiplier is given by:

---

*The motivation behind this use of a narrow-band filter is entirely valid, but temperature-induced phase shifts in the crystal filter necessitated its removal.

$$
\begin{aligned}
M &= k \cdot e_1 \cdot e_2 \cdot \sin \omega t \cdot \sin(\omega t + \emptyset) \\
&= k \cdot e_1 \cdot e_2 \left\{ \sin \omega t \cdot (\sin \omega t \cdot \cos \emptyset + \cos \omega t \cdot \sin \emptyset) \right\} \\
&= k \cdot e_1 \cdot e_2 \left[\cos \emptyset \left\{ \frac{1 - \cos 2\omega t}{2} \right\} + \sin \emptyset \left\{ \frac{\sin 2\omega t}{2} \right\} \right] \\
&= k \cdot e_1 \cdot e_2 \left[\frac{\cos \emptyset}{2} + \text{terms in } 2\,\omega t \right]
\end{aligned}
$$

where k is an arbitrary constant. The terms in $2\omega t$ are easily removed by a low pass filter leaving an output depending only on cos $\emptyset$; We shall be working over the linear region either side of 90 degrees, as shown in Figure 32. Note that the output varies with e_1 and e_2, therefore, these must be kept constant to at least the precision of the range measurement.

Alternatively, the two waveforms applied to the multiplier may be converted to square waves, in which case the output varies directly as $\emptyset$ rather than cos $\emptyset$. However, the limiter problem is implicit in whatever phase measuring device we choose since a constant output reading must be provided for a receiver input level varying from 1 millivolt to 1 volt.

B. Sensitivity and Signal-to-Noise Ratio

1. Received-Signal Level

The initial design assumptions were as follows:

Scanning transmitter power (collimated beam) = 10^{-3} watt

Maximum range = 10 feet

Minimum reflectivity = 1 percent

The reflected power was assumed to be scattered uniformly over a hemisphere.

Receiver capture area = 0.25 square inch.

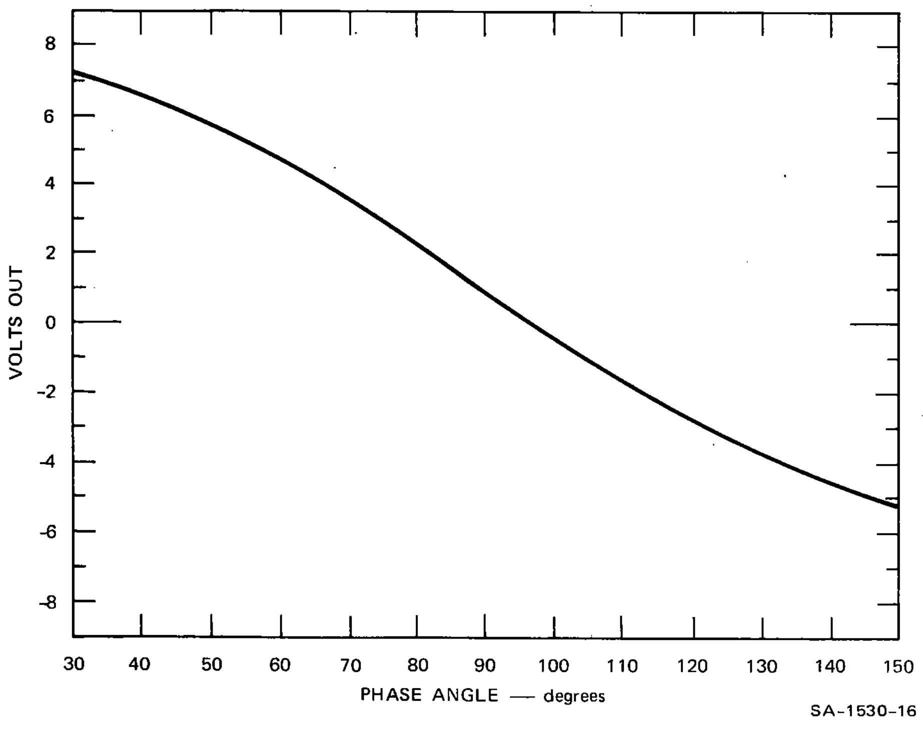


FIGURE 32 ANALOG OUTPUT FUNCTION

The minimum received power, P, applied to the photomultiplier is

$$P = 10^{-3} \times 10^{-2} \times \frac{0{\cdot}25}{2\pi(120)^2} = 2{\cdot}5 \times 10^{-11} \text{ watt}$$

The sensitivity of the RCA CA 31034 photomultiplier is given as 4×10^4 amps per watt at 6328 a.u. For an input of $2 \cdot 5 \times 10^{-11}$ watt, the output current, I, is given by

$$I = (2{\cdot}5 \times 10^{-11}) \times (4 \times 10^4)$$
$$= 10^{-6} \text{ amp}$$

Assuming a load impedance of 4,000 ohms and a transmission of 25% by the interference filter,

Minimum output level = $4{,}000 \times \frac{1}{4} \times 10^{-6}$ volts

= 1 millivolt

The maximum signal to be anticipated is about one volt, since the working range variation of from 2 1/2 feet to 10 feet results in a 16-fold increase of signal, and the transition from a "black" object to a "white" object gives another factor of 60. The working range of the limiter must therefore be at least 60 db.

2. Phase Noise

The constraints on receiver phase noise can be evaluated as follows; The vector OA in Figure 33 represents the return-signal voltage out of the head amplifier. AB is the random noise contribution from all sources--head amplifier noise, jitter due to power supply variations, etc.--so that the instantaneous value of the output voltage is given by OB. OR is the reference phase and the output of the system is the phase angle between OR and OB. Thus, the angle AOB represents phase noise, the fluctuation in the output measurement corresponding to the noise vector AB.

It is desired that the equipment yield an accuracy of 0.1 inch over the range 2-1/2 feet to 10 feet, or 1 part in 1,000. Since we have previously assumed in the design that will vary from +50 degrees to -50 degrees, the phase noise level must be approximately equal to 0.1 degree, or 1/600 radian. Thus, the required minimum signal-to-noise

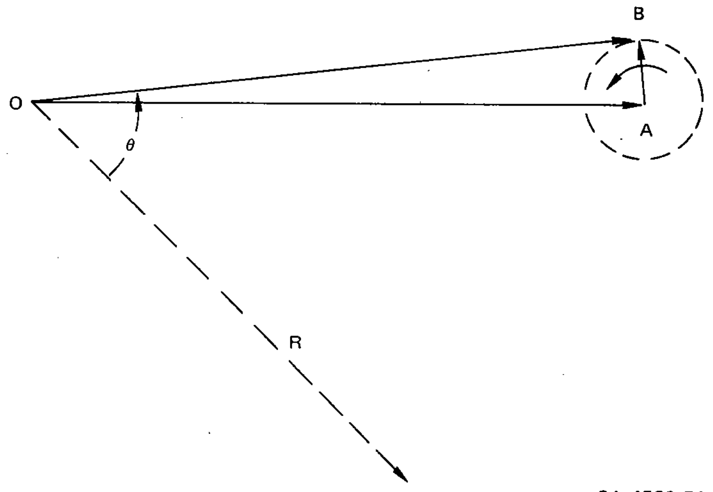


FIGURE 33 EVALUATION OF PHASE NOISE DUE TO RECEIVER CHANNEL

ratio is the ratio of the vector OA to OB, or 600:1. If the minimum received signal OA is 1 millivolt, then the noise level of the system must be 1 microvolt. For 1,000 measurements per second and a receiver bandwidth of 2.5 KHz, a receiver noise level of 1 microvolt at 9 MHz is a practical value.

3. Quantum Noise

The minimum received power has been calculated at $2 \cdot 5 \times 10^{-11}$ watt; and combined with 25% transmission by the interference filter and 10% efficiency at the photocathode, the equivalent power level referred to the photocathode of the photomultiplier is

$$P = (2{\cdot}5 \times 10^{-11})(0{\cdot}25)(10^{-1}) \approx 6 \cdot 10^{-13} \text{ watt}$$

The energy, E, associated with each photon is given by $\frac{hc}{\lambda}$

$$E = \frac{(6{\cdot}6 \times 10^{-34})(3 \times 10^{10})}{6328 \times 10^{-8}} \approx 3{\cdot}13 \times 10^{-19} \text{ watt seconds}$$

The worst case rate of electron generation at the photocathode is

$$\frac{6 \times 10^{-13}}{3{\cdot}13 \times 10^{-19}} \approx 2 \times 10^{6} \text{ per second}$$

If we make a measurement every millisecond, each sample will in the worst case, on the average, include only 2000 electrons, with a random variation of $\sqrt{N}$ = 45 electrons. The phase noise θ = 45/2000, which equals 1/45 radian, or about 1-1/4 degree. The phase noise due to photoelectron emission is made worse by the

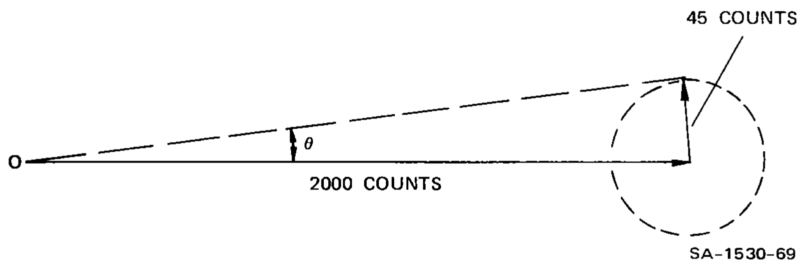


FIGURE 34 EVALUATION OF PHASE NOISE DUE TO FINITE NUMBER OF QUANTA IN THE MEASURING SAMPLE

multiplier noise at the first dynode and also by the dark current noise, amounting to a factor 1.5, giving a measuring error approximately equal to 2% of the working range -50 degrees to + 50 degrees. (We have only 2,000 photo-electrons with which to generate an electrical signal representing 9,000 cycles of 9 MHz.) For a gray object of reflectivity 17% at 5 feet, the error will be reduced by a factor of $\sqrt{17 \times 4}$ times, which is approximately equal to 8.

4. Noise Due to Ambient Illumination

We will assume that the stray light has the spectral quality of sunlight and the receiver "sees" an area of one square foot at ten feet.

Direct sunlight at 6328 a.u., amounts to 0.18 watts/square metre/a.u., which equals 0.32 watts/square foot in a 20 a.u. pass band.

If the ratio of room light intensity to direct sunlight is 200:1, then stray light from the area seen by the receiver is equivalent to radiation of power level = 1.6 milliwatts in the 20 a.u. pass band of the receiver.

We will assume that the stray light patch has a reflectivity of 20% (middle gray). In the worst case of a 1% reflecting object, the power received from the ambient illumination will be (1.6/1.0)20 $\approx$ 30 times greater than the laser return power from the scanned object. It will contribute $(30)^{\frac{1}{2}}$ or 5.5 times as much ambient noise as the inherent noise in the photon count for the signal, so that the random variation for a 1 millisecond sampling period becomes 7% rather than 1-1/4%. This dependence of the signal to noise ratio upon the ambient illumination level is in fact observed, and in a typical case, the noise level on the signal from a wall at 10 feet fell by a factor of four when the room was darkened.

Evidently, it is worth expending considerable effort to improve this situation; in fact a substantial improvement has been obtained by a redesign of the optical layout so the area that the receiver "sees" has been reduced from 1 square foot, corresponding to a maximum off-axis angle of 4 degrees, to 1/64 square foot and 1/2 degree off-axis angle as shown in Figure 35. A lens was fitted behind the interference filter and an image of the scanning spot (at 10 feet) formed on a diaphragm containing a 1-mm pinhole ahead of the

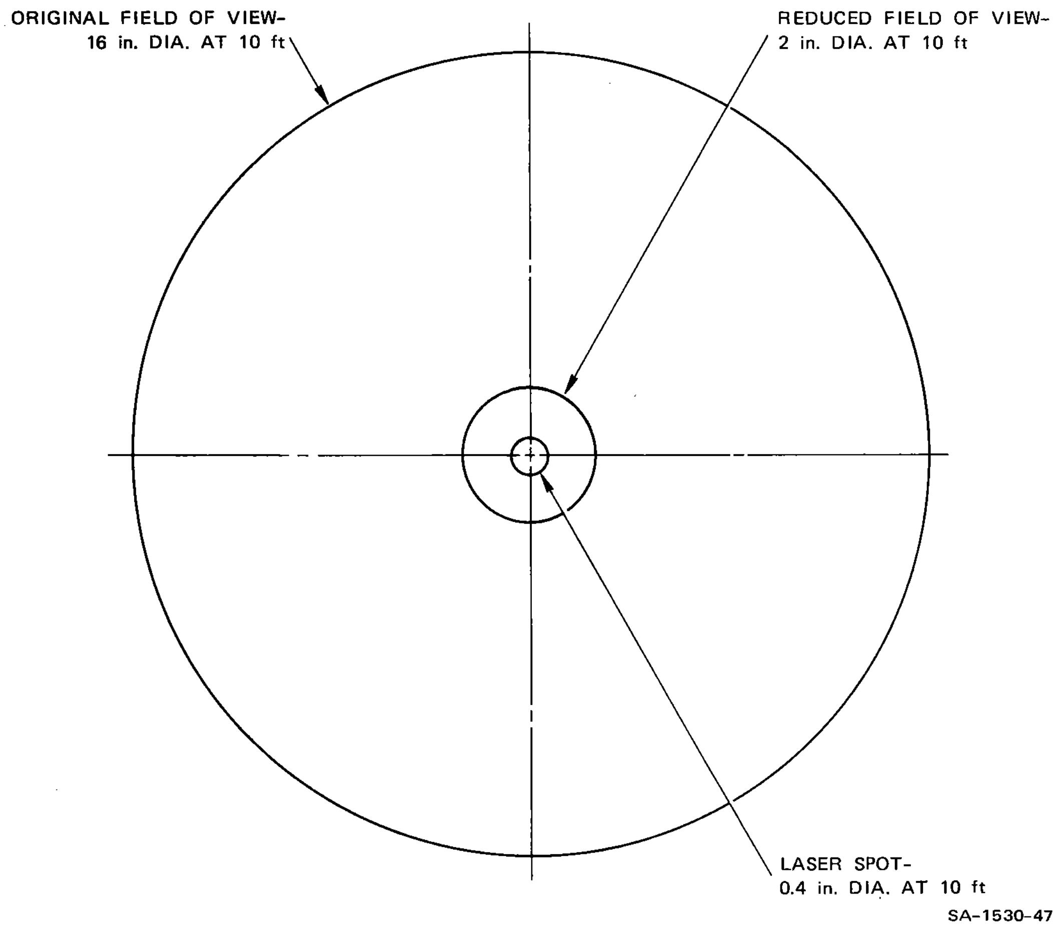


FIGURE 35 REDUCTION IN THE FIELD OF VIEW OF THE RECEIVER CHANNEL TO MINIMIZE NOISE DUE TO AMBIENT ILLUMINATION

photomultiplier, as shown in Figure 36. The noise created by the ambient illumination has been reduced to a level approximating that due to the quantum noise contribution of the return signal from the laser itself.

To summarize the result of the above calculations in terms of practical equipment design, a 1-milliwatt laser may be expected to provide a 0.25 inch accuracy at 5 feet on a 20% reflecting object in

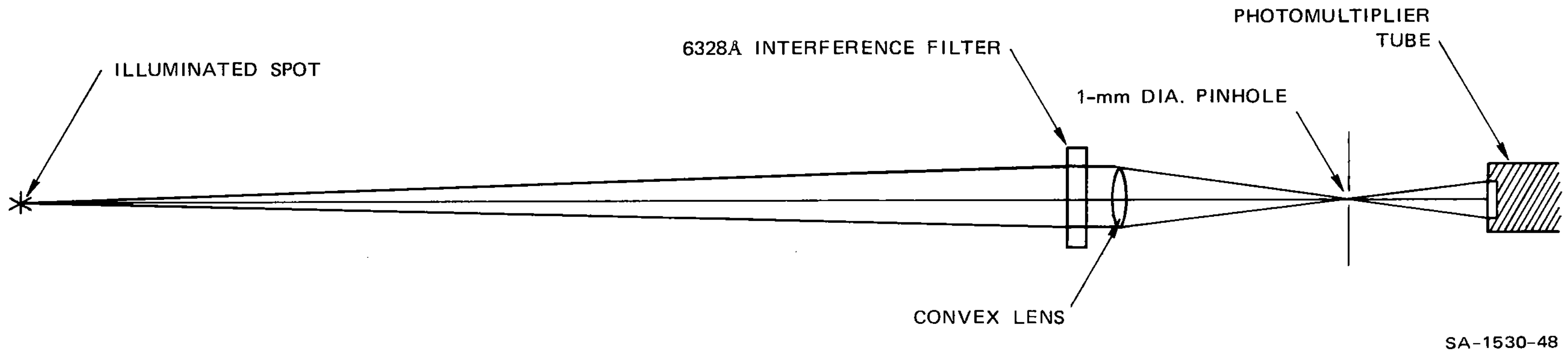


FIGURE 36 OPTICAL CONFIGURATION USED TO REDUCE AREA "SEEN" BY RECEIVER CHANNEL

1 millisecond, subject to the numerous other parameters that have been given realistic values, and assuming that the effect of ambient illumination has been reduced to a negligible level. This result has to do only with counting photons, and can be improved upon only by increasing the laser power. The power, P, required to provide a 0.25- inch accuracy at 10 feet on 4% reflecting object would be

$$\left(\frac{10}{5}\right)^2 \cdot \frac{20}{4} = 20 \text{ milliwatts}$$

assuming the same 1-millisecond sampling interval. It is impractical to use a longer sampling interval since at 120 X 120 resolution the total picture time amounts to $(120)^2 \times 10^{-3} = 14.4$ seconds. which is already in excess of what would be desirable.

C. Hardware Design

1. Transmitter Section

The general layout of the transmitter section is shown in Figure 37. So far all of the experimental work has been carried out using a Hughes 3-mW helium-neon laser; by the time the modulated beam leaves the scanning mirror the actual radiated power is close to the 1-mW assumed in the previous calculations. In view of the computations given in the section above, it is evident that a considerable increase in transmitter power is mandatory if the desired data rate and accuracy are to be achieved. When this is done, additional shielding and care in operation will be necessary to guard against possible damage from radiation splatter to the eyesight of personnel.

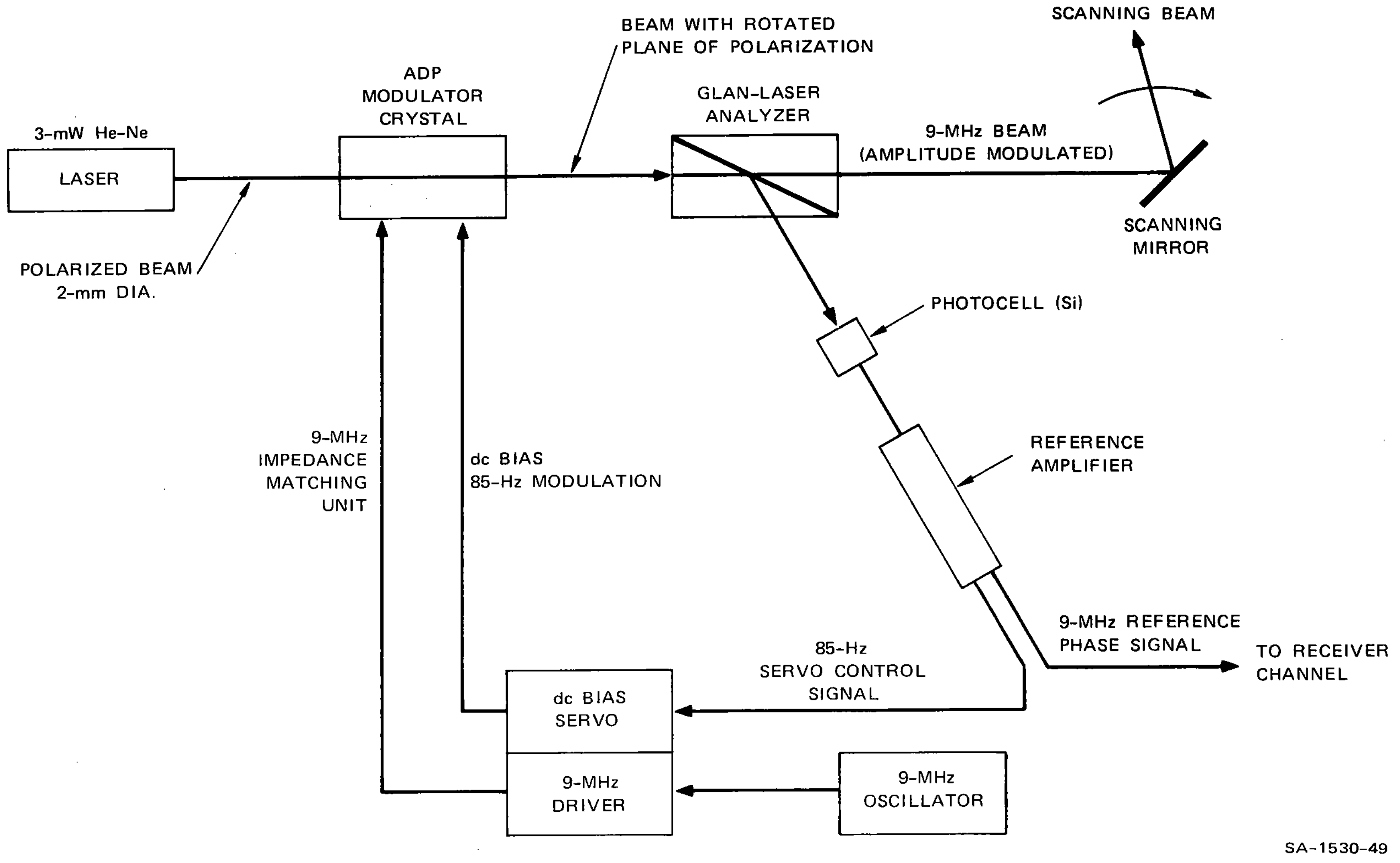
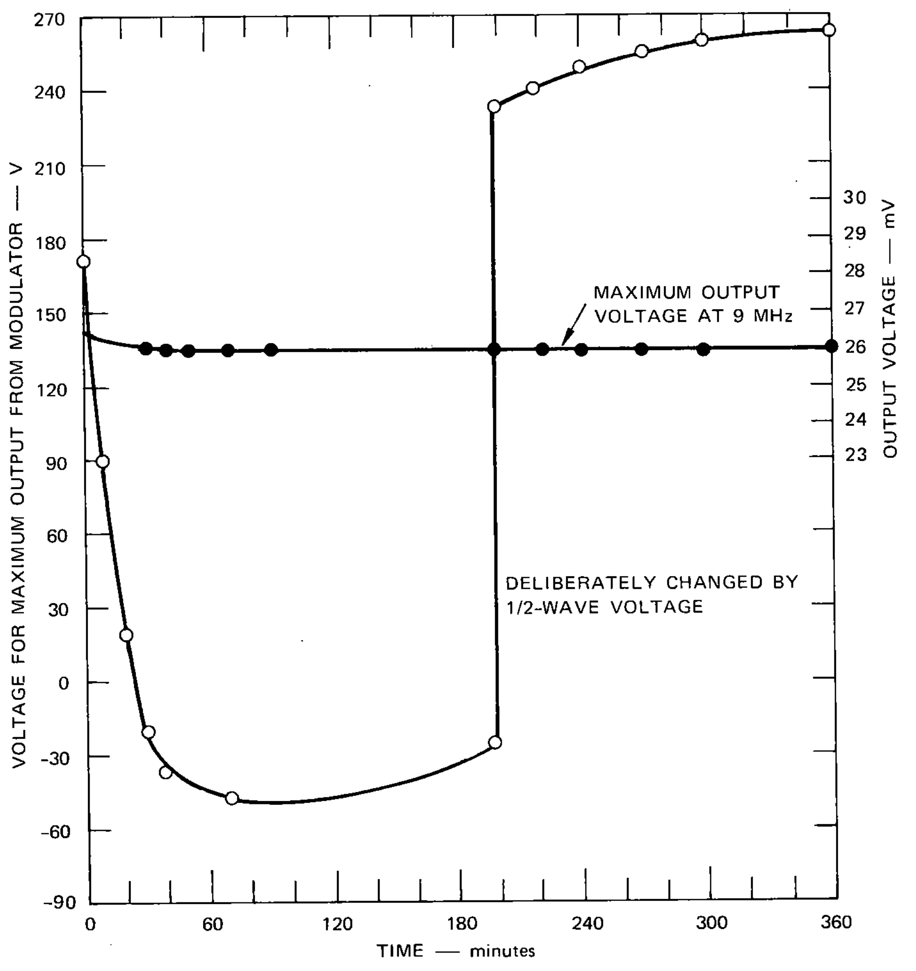


FIGURE 37 TRANSMITTER SECTION OF LASER RANGEFINDER

The 6328 a.u. beam from the helium-neon laser is plane polarized, and the plane of polarization is rotated to an extent depending on the bias level of the ADP crystal, the half-wave voltage of which is 250 volts. At 9 mhz the required drive power is approximately 0.75 watt, and the circulating power through the crystal is approximately 50 volt-amps.It is necessary to pay careful attention to the radio-frequency shielding associated with the modulator drive hardware since the receiver channel, which is physically adjacent, is tuned to the same frequency and the phase measurement is vulnerable to the addition of coherent interference signals. One consequence of this appreciable circulating power is heating of the crystal and the development of minor temperature gradients within the crystal. These give rise to a progressive rotation of the plane of polarization amounting to 180 degrees over a period of an hour or more. This rotation results in a reduction of the level of modulation imposed on the laser beam, and can be cancelled by the superimposition of a DC bias level across the crystal. This effect is shown in Figure 38, in which the output from the receiver channel was "tuned" for maximum output by varying the DC bias level on the crystal, leaving the rf drive constant. Even though the required bias varied over a range of 200 volts, the maximum output signal changed hardly at all.

Obviously this effect is of major significance in determining the performance of an instrument that will be switched on and allowed to run indefinitely. Rather than seeking to eliminate the temperature gradients, assuming these to be the cause, or to find a modulator crystal of a different chemical composition that did not show the effect, it was decided to add a servo loop to provide the necessary



SA-1530-50

FIGURE 38 VARIATION WITH TIME OF OPTIMUM BIAS ON ADP MODULATOR CRYSTAL

optimizing bias to the crystal. The circuit arrangement to do this is shown in Figure 39.

The circuit works as follows. The ADP modulator crystal is driven simultaneously by three signals: (1) the 9-MHz drive, which is approximately 200 volts peak-to-peak; (2) a DC level out of the servo circuit, which may vary over the range 50-350 volts, and (3) an 85 hz signal, approximately 10 volts peak-to-peak, that is superimposed on

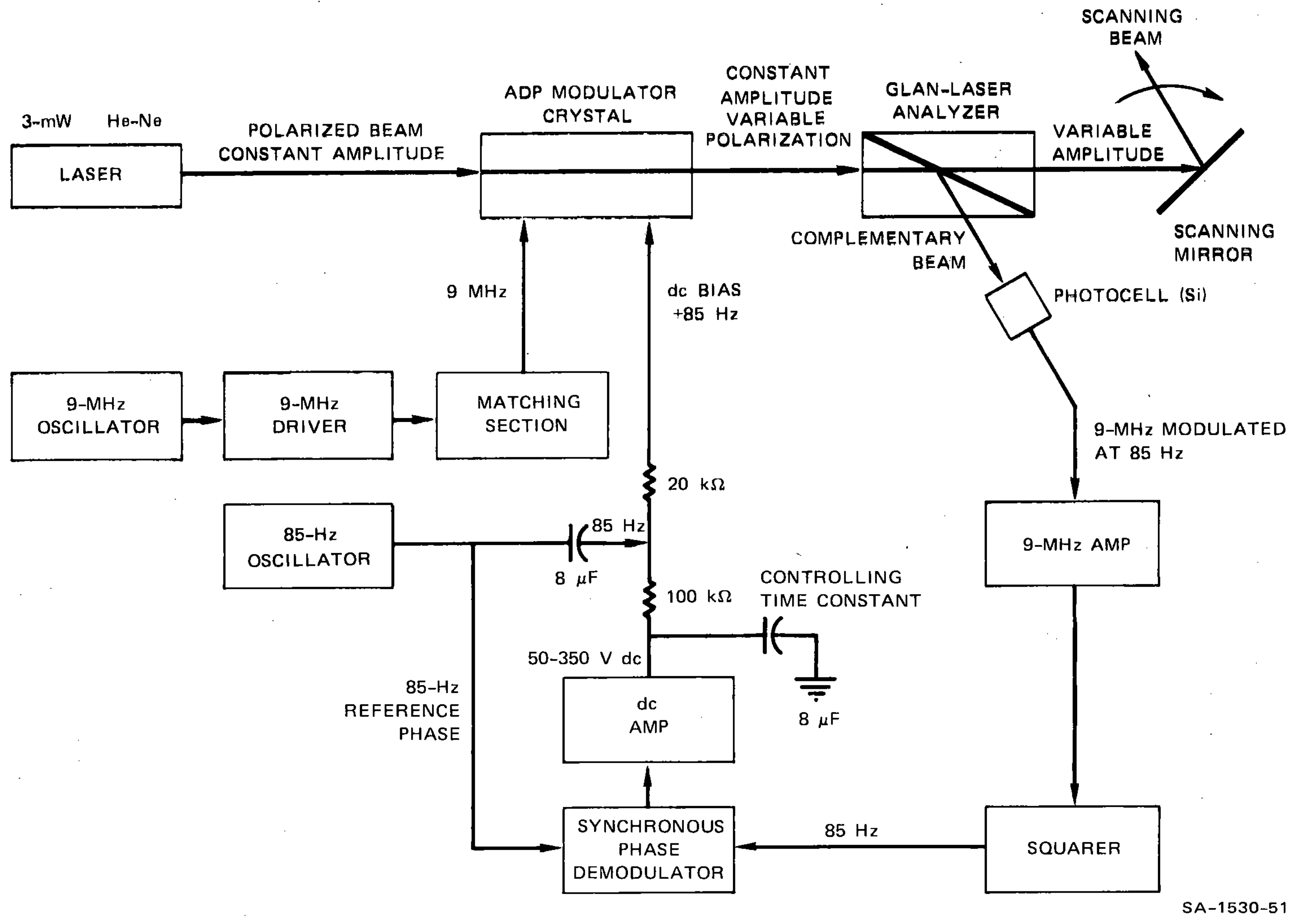


FIGURE 39 LASER MODULATOR SERVO LOOP

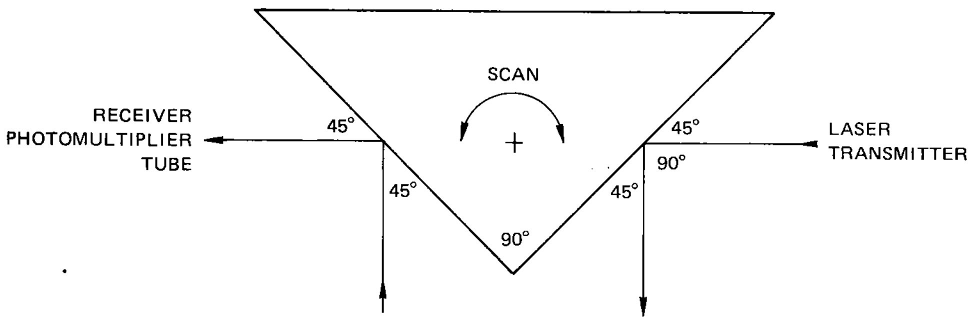
the DC bias voltage. When the bias circuit is providing the optimum control voltage, the 9-MHz modulation on the beam carries a superimposed 85-Hz amplitude modulation, less than 4 percent in magnitude. This 85-Hz modulation is recovered from the laser beam of the reference channel and applied to a synchronous phase demodulator together with an 85-Hz signal taken directly from the controlling oscillator. The resulting DC output level is amplified and used to control the bias level on the modulator. With the correct loop gain and time constants, the settling time to provide maximum 9-MHz modulation on the beam is approximately 0.5 second. The 4 percent 85-Hz amplitude modulation on the laser beam is not detrimental in our application, since it is removed in the limiter of the receiver channel.

The modulator crystal used in the equipment is a Lasermetrics EOM-3078A mounted on a lasermetrics MV-120D base (adjustable in pitch and yaw).

2. Scanning Mirror

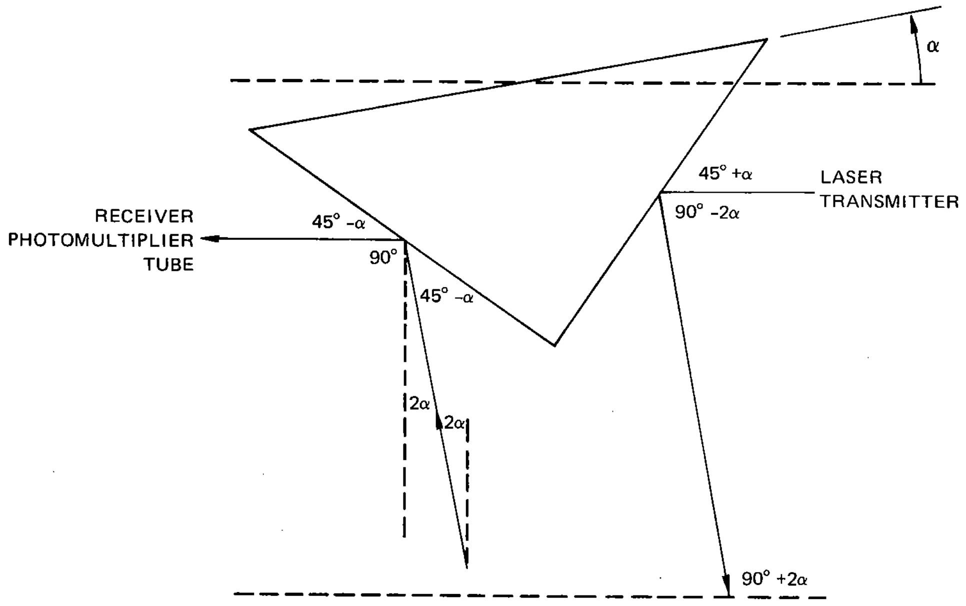
The initial work was done with the two mirror scanning unit made by General Scanning. This unit consists of two galvanometer scanning units spaced approximately 1 inch apart, and does an excellent job of scanning the transmitter beam; however, for our application it has a limitation that renders it unusable. The overriding consideration is to ensure that no scattered light from the mirrors and other surfaces associated with the transmitter channel can leak into the photomultiplier of the receiver channel. Such leakage gives rise to a spurious signal of constant phase that is added vectorially to the return signal from the target and is essentially indistinguishable from it. The phase of the resultant is grossly in error. Therefore the receiver channel must not be able to look back through the system and see the transmitter scanning mirror except via the subject being scanned. It is strictly inadmissible to use the same mirror for transmitting and receiving. Since the transmitter power is approximately 10^{-3} watt and the received power in the worst case is down to 10^{-11} watt, the backscatter from the mirror will inevitably swamp the return signal.

Of the configurations for the scanning mirror assembly that were examined only one was found to be satisfactory as a practical device; it is shown in Figure 40. The design starts from the premise that the transmitter scanning mirror cannot be the same as the receiver scanning mirror, but at the same time any parallax error must be extremely small. In scanning it is essential to use the slow forward sweep rapid-return-sweep type of scan, rather than attempt to use both forward and return sweeps for signal information. This is because nonlinearity effects are so critical that registration of edge information becomes impractical. By using only one sweep, the same nonlinearity occurs in register on every line. We desire a sweep time of 1/10 second or less, and a return time of 1/50 second or better, so that the moment of inertia of the mirror scanning system must be kept small; that means that the mirrors must also be small and light. The mirrors used in the experimental tests have been aluminized microscope cover glasses, and these were glued to an aluminum supporting frame. The parallax between the transmitting and receiving beams is approximately 1/2 inch.



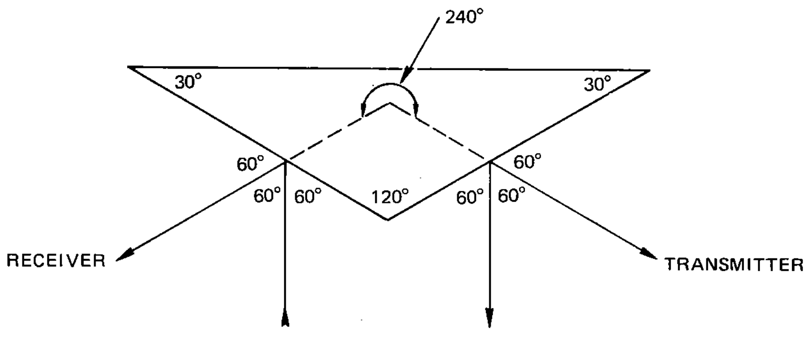
(a) **LASER AND PHOTOMULTIPLIER IN LINE**

SA-1530-52a



(b) **MIRROR ROTATED THROUGH ANGLE** α

SA-1530-52b



(c) **MIRROR WITH 120° APEX ANGLE**

SA-1530-52c

FIGURE 40 SCANNING MIRROR CONFIGURATION

The primary virtue of this configuration is that the transmitter hardware axis (between scanning mirror and laser) and the receiving channel axis (between scanning mirror and photomultiplier) are fixed and, in our experimental equipment, collinear. Figure 40(b) shows the result of rotating the mirror assembly through an angle alpha; if the values of the various angles are checked, it will be seen that the return beam emerges parallel with the laser beam. The interference filter in the receiver channel has a width of only 20 a.u. in wavelength, and the center frequency decreases with increasing path length; thus, at 4 degrees off axis there is an enormous reduction in transmission, since the center wavelength has moved to 6343 a.u. from 6328 a.u., or 15 a.u., as compared with a filter half-width of 10 a.u.. This effect can be important at near distances, but at a range of 30 inches, an off-axis error of 0.5 inch amounts to less than 1 degree of parallax, and so is not a serious detriment.

While the in-line configuration shown in Figure 40(a) is convenient and will probably be retained in further work on the laser scanner, it is in fact an example of the more general case, in which the angle between the transmitting and receiving beams is twice the apex angle between the two mirrors. The 120 degrees included angle case is shown in Figure 40(c). It is attractive since it offers a more efficient utilization of the area of the receiver mirror, and yet still retains freedom from obstruction.

Scanning in the vertical direction is accomplished by rotating the mirror assembly about the transmitter-receiver common axis. A dc position servo gives an accuracy of approximately 1 part in 500, with the control voltage being directly proportional to the rotation theta, as defined by a linear potentiometer and suitable gearing.

The completed scanning mirror hardware is shown in Figure 41. Note that the photomultiplier is housed within the hollow rotating shaft that provides vertical deflection. This configuration is recommended.

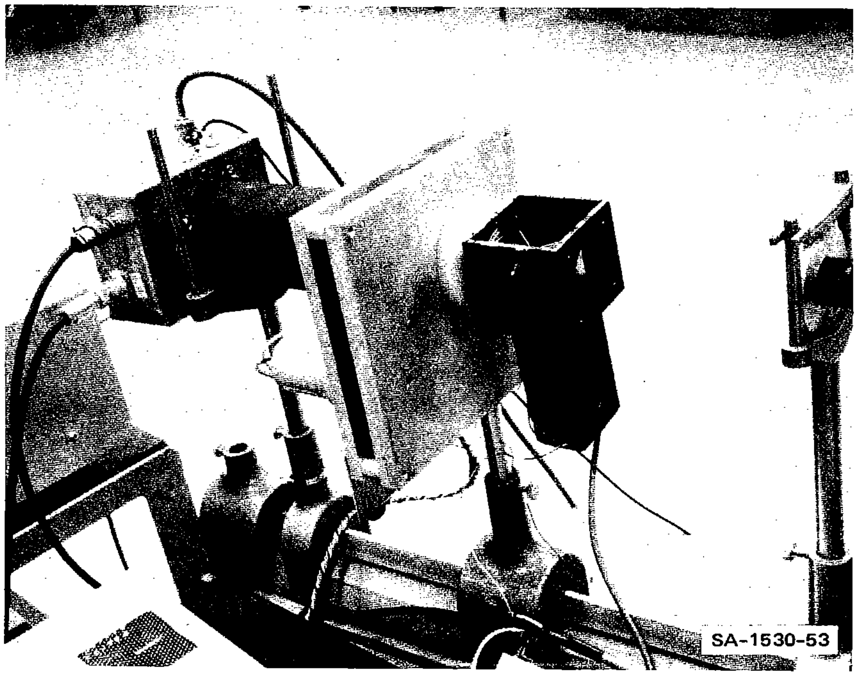


FIGURE 41 MIRROR SCANNER HARDWARE

3. Receiver Section

The receiver section of the scanning rangefinder is essentially a phase meter. It measures the phase of the output from the photomultiplier relative to the output from the photocell that receives the reflected beam coming out of the Glan-Laser prism. The prism is a zero-loss device, and hence the reflected beam modulation must be complementary to that of the direct beam. Since the input beam is of constant amplitude and the direct beam is amplitude modulated at 9 mhz, the reflected beam also carries modulation at 9 mhz, phase-shifted by 180 degrees.

The specifications of the characteristics of the phase meter follow from the calculations given previously:

Operating frequency:	9.0 MHz
Input level:	1 mv to 1 volt (or more)
Bandwidth:	2.5 kHz
Receiver noise level:	less than 1 microvolt
9.0 mhz Reference signal level:	approximately 1 volt, with 4 percent modulation at 85 Hz
Phase dynamic range:	$-50° < \theta < 50°$
Desired accuracy:	0.25 degree to 0.1 degree.

The crucial problem in the receiver is to achieve 60 dB or more of limiting without at the same time incurring a phase shift greater than desired. The effect of such phase shifts may best be appreciated by considering a practical example. Suppose the laser beam is scanning across a typical multirange voltmeter, with a black plastic case and a white dial behind the pointer. The phase offset introduced in scanning over the dial makes it appear to be moved out from the black case by an inch or more--a phase shift of 1 degree is approximately equal to a range difference of 1 inch. Zero phase shift implies exactly equal limiting on both halves of the 9- mhz waveform, and this exact equality must be maintained over the whole dynamic excursion from a 2-millivolt input to 1-volt input. Such symmetry in turn implies identical devices with a physically symmetrical layout of leads and geometry. The hardware that has been constructed using integrated circuits creates phase shifts of approximately 1 degree, which is worse than the desired specification by a factor of from 4 to 10. It is obvious that the design of a limiter to satisfy our requirements is a nontrivial problem.

However, certain pieces of laboratory test gear -- on which a great deal of development time have been lavished -- are capable of measuring phase at 9 MHz. Two such units have been tested. The Hewlett-Packard 8405A Vector Voltmeter is a sampling device measuring both amplitude and phase. Its specification states that the phase measurement is preserved with an accuracy of plus and minus 2 degrees for a signal level change from 3 mv to 1 volt. The bandwidth is given as 1 khz, and the search and lock time as 10 milliseconds; however, the observed transient settling time as the laser beam scanned over the edge of a box at 5 feet onto a rear wall at 10 feet was approximately 30 milliseconds. A low pass RC filter was used to clean up the reconstructed signal at the rear output terminal, and a satisfactory picture was taken in at 120 by 120 resolution. However, at 30 milliseconds per picture point, the time for a complete picture is 430 seconds, or 7 minutes, which is impractical. There is no way to speed up the process since it is tied back to the sampling time of the interval circuits, which control the settling time of the phase measurement at the output.

The second piece of laboratory phase-measuring equipment that has been tested is the Hewlett-Packard 8407A Network Analyzer combined with the 8412A Phase-Magnitude Display and the 8601A Sweeper. All of these pieces are necessary to provide an operating system to satisfy our requirements. The 9 mhz input signal is converted to 278 khz for measurement; the bandwidth is 10 khz for maximum information display, or 100 Hz to filter displayed noise. The phase error vs. limiting ratio performance curve is given in Figure 42. This is a very elaborate and highly developed combination of equipment; it probably represents the closest approximation to state-of-the-art that is commercially available. For a dynamic range of 40 dB, which is typical of many scenes, its performance approximates to our design specification. Unfortunately, the cost is very high, and the physical size and weight render it impractical for anything except laboratory testing.

D. Experimental Results

The practical experiments have been carried out using 1 milliwatt of laser scanning power, and as a consequence our measuring rate and signal-to-noise ratio have been photon limited. The calculations given in Section II are in good agreement with the experimental observations, and seem to provide a satisfactory basis for the design of equipment that is more than an experimental test bed. Increased laser power is obviously essential, but here is an understandable reluctance to have more than 1 milliwatt of laser power splattering around in a laboratory shared with many other people. An increase to 30 milliwatts or more would seem to be appropriate but, since we are counting photons, a substantially equivalent result may be obtained by stretching the time scale by a factor of 30. A typical experimental scene is shown in Figure 43. It shows an open cardboard box, roughly 10 inches on a side, and having the flap raised; the box

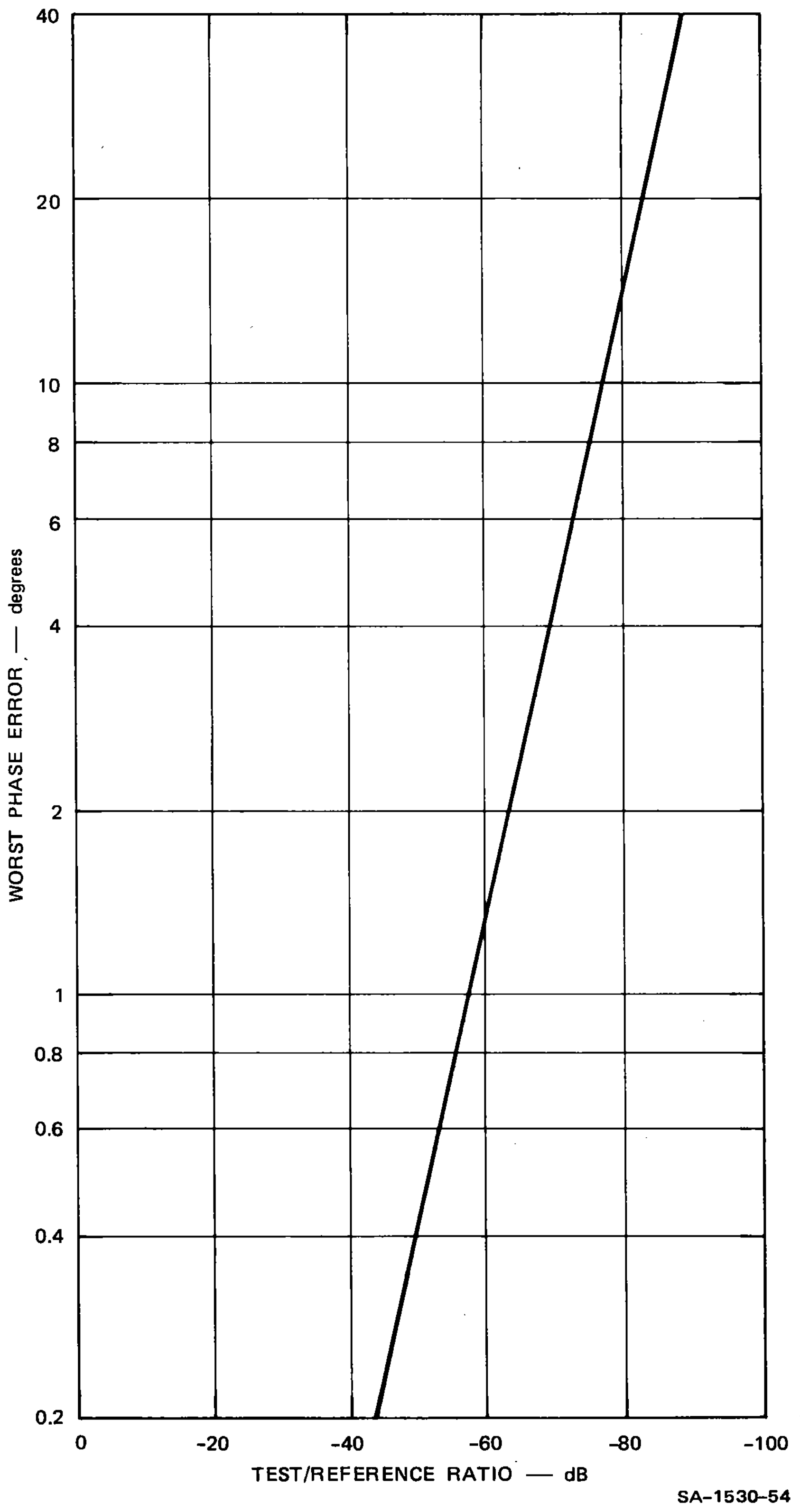


FIGURE 42 PHASE ERROR CHARACTERISTIC OF HEWLETT-PACKARD 8407A

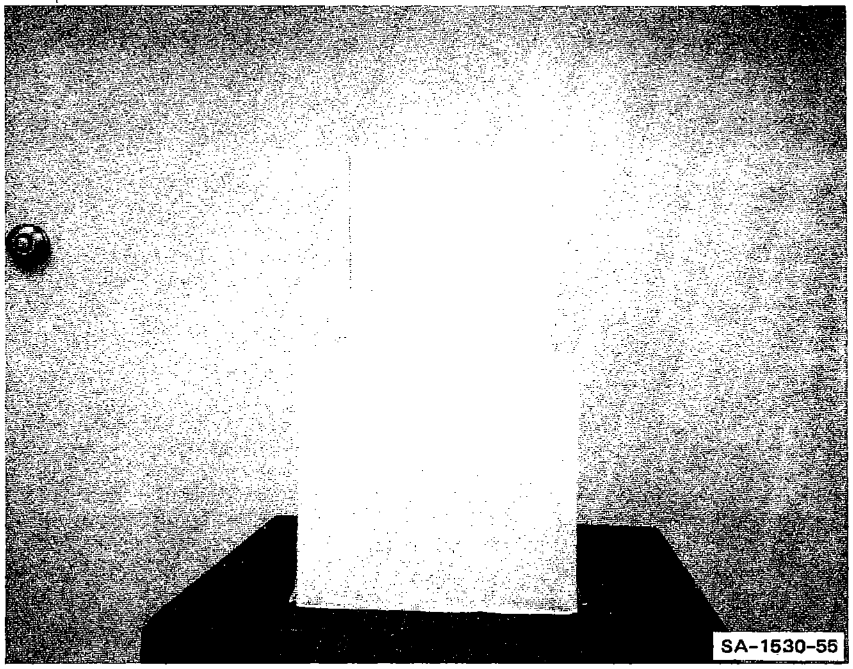


FIGURE 43 PHOTOGRAPH OF BOX AND REAR-WALL (TEST PICTURE)

is approximately 6 feet from the range scanner, and behind the box there is a flat wall approximately 4 feet farther away. The data was read into the A-D converter, with 30-millisecond integration time per point, at 120 by 120 resolution. The range signal was given a dc offset, and the analog gain was increased so that the dynamic range of the A-D converter was effectively utilized. Part of the printout covering the outlined area of Figure 43 is shown in Figure 44. The overall scanning angle is aproximately 30 degrees, so that on the rear wall 10 feet away the picture points are approximately 0.6 inch apart. The scanning spot is somewhat less than 0.6 inch in diameter at 10 feet, being proportionately smaller closer in. The box is 15 inches deep, corresponding to 45 range units, so that a resolution incEMENT OF RANGE IS EQUAL TO 0.33 inch. Across the middle of the box the variation is from 21 to 23 units or plus and

```
133-134-133-134-133-134-133-133-134-133-134-134-133-131-134-135-133-133-132-132-
131-132-133-133-132-134-132-132-131-132-133-133-133-132-132-131-132-131-130-135-
135-134-131-131-132-133-132-132-132-131-133-131-133-131-131-132-133-131-132-135-
115-116-116-115-116-117-113-114-111-112-113-113-112-109-108-115-111-125-126-131-
-17 -20 -24 -25 -26 -27 -26 -29 -26 -27 -27 -29 -28 -26 -26 -29 -27 -35-123-127-
-25 -27 -26 -27 -26 -27 -26 -28 -26 -27 -27 -28 -26 -29 -27 -27 -28 -29-113-126-
-26 -25 -26 -27 -26 -27 -28 -28 -27 -27 -27 -28 -27 -27 -27 -26 -28 -34-118-127-
-25 -28 -27 -27 -27 -26 -26 -27 -27 -28 -27 -28 -27 -28 -27 -28 -27 -28-115-124-
-25 -27 -25 -26 -25 -26 -25 -26 -26 -26 -26 -28 -26 -27 -27 -27 -27 -32-116-127-
-25 -28 -25 -25 -26 -25 -27 -27 -26 -28 -26 -28 -27 -27 -28 -27 -27 -28-115-124-
-25 -26 -24 -26 -26 -27 -25 -26 -26 -25 -25 -26 -26 -27 -28 -26 -25 -32-120-125-
-27 -28 -26 -25 -26 -27 -27 -25 -26 -25 -26 -26 -26 -27 -25 -26 -27 -27-112-124-
-26 -25 -26 -26 -27 -25 -27 -25 -27 -26 -27 -25 -27 -25 -25 -26 -28 -30-119-124-
-25 -25 -25 -25 -25 -26 -27 -27 -26 -26 -26 -26 -25 -26 -26 -26 -25 -27-113-122-
-26 -25 -26 -25 -26 -26 -26 -26 -25 -27 -26 -26 -25 -26 -25 -24 -26 -30-117-123-
-25 -25 -25 -24 -23 -25 -27 -26 -25 -25 -25 -26 -26 -24 -25 -24 -24 -27-112-123-
-25 -24 -24 -24 -25 -25 -24 -25 -25 -24 -27 -26 -23 -26 -26 -25 -26 -29-114-123-
-25 -26 -25 -24 -26 -25 -25 -24 -25 -24 -26 -25 -25 -26 -25 -25 -25 -26-103-121-
-25 -25 -25 -23 -25 -23 -25 -25 -25 -26 -24 -24 -26 -24 -24 -25 -24 -27-118-123-
-25 -24 -24 -24 -25 -24 -24 -24 -25 -25 -23 -25 -25 -24 -26 -25 -24 -26-107-120-
-26 -25 -26 -25 -24 -23 -26 -25 -24 -24 -24 -25 -24 -24 -26 -26 -25 -27-111-123-
-24 -25 -25 -24 -24 -24 -24 -25 -24 -24 -24 -23 -24 -27 -24 -23 -24 -24-112-120-
-24 -24 -25 -25 -25 -23 -25 -25 -25 -24 -23 -25 -23 -24 -25 -25 -25 -25-116-124-
-24 -25 -23 -25 -25 -25 -23 -25 -26 -25 -25 -24 -25 -24 -26 -24 -24 -26-109-120-
-24 -23 -23 -24 -25 -25 -23 -23 -23 -24 -24 -24 -23 -25 -24 -24 -24 -27-114-122-
-24 -25 -24 -24 -24 -25 -22 -23 -24 -23 -24 -23 -24 -23 -25 -26 -24 -26-103-121-
-23 -24 -23 -22 -24 -23 -23 -23 -24 -25 -24 -23 -24 -23 -23 -24 -24 -27-112-123-
-24 -22 -24 -24 -23 -23 -25 -24 -24 -24 -24 -24 -23 -24 -23 -23 -24 -26-114-120-
-23 -22 -23 -24 -24 -23 -24 -23 -24 -24 -23 -23 -24 -24 -22 -24 -23 -28-115-121-
-23 -24 -23 -25 -24 -24 -24 -24 -24 -24 -23 -24 -24 -24 -23 -23 -22 -25-109-121-
-24 -23 -23 -22 -24 -22 -23 -22 -23 -23 -23 -23 -25 -24 -23 -23 -24 -27-112-120-
-22 -23 -23 -22 -22 -22 -22 -23 -25 -24 -23 -23 -24 -21 -23 -26 -23 -25-113-119-
-23 -23 -23 -23 -25 -24 -23 -23 -23 -22 -23 -23 -23 -25 -25 -24 -24 -26-114-121-
-23 -23 -22 -22 -24 -23 -23 -23 -23 -23 -22 -23 -23 -23 -22 -23 -24 -25-107-121-
-23 -23 -24 -23 -22 -23 -22 -22 -23 -22 -22 -21 -23 -22 -23 -22 -23 -26-114-120-
-22 -23 -22 -22 -22 -23 -24 -23 -21 -23 -22 -21 -24 -22 -23 -23 -24 -26-105-118-
-23 -25 -23 -23 -24 -22 -22 -23 -22 -23 -22 -23 -23 -21 -24 -22 -24 -24-112-121-
-22 -22 -23 -24 -22 -23 -23 -22 -23 -22 -23 -23 -23 -24 -22 -23 -23 -24-107-118-
-22 -22 -23 -23 -21 -22 -21 -22 -23 -22 -22 -21 -22 -22 -22 -22 -22 -26-113-120-
-22 -22 -24 -22 -22 -22 -21 -22 -23 -22 -22 -21 -23 -23 -22 -23 -24 -25-106-119-
-22 -21 -22 -22 -22 -23 -22 -23 -23 -23 -22 -23 -23 -22 -22 -23 -23 -30-114-120-
-21 -21 -22 -22 -22 -24 -23 -21 -22 -22 -22 -22 -23 -22 -23 -21 -23 -23-110-117-
-22 -21 -22 -22 -21 -22 -21 -23 -23 -22 -22 -23 -22 -21 -22 -22 -23 -24-110-122-
-22 -22 -21 -22 -23 -23 -21 -23 -21 -22 -23 -23 -23 -22 -22 -22 -22 -21-106-122-
-23 -23 -21 -22 -23 -21 -21 -22 -24 -22 -20 -22 -23 -21 -23 -23 -23 -23-110-118-
-21 -21 -22 -22 -21 -22 -22 -23 -21 -21 -21 -23 -22 -22 -22 -22 -22 -24-113-117-
-21 -22 -22 -21 -22 -22 -22 -23 -23 -21 -21 -21 -21 -22 -23 -22 -20 -25-113-118-
-21 -22 -22 -22 -21 -20 -22 -23 -22 -23 -22 -23 -22 -23 -22 -22 -23 -24-109-117-
-22 -21 -21 -21 -21 -19 -21 -21 -20 -21 -22 -20 -21 -20 -21 -21 -21 -23-112-119-
-19 -19 -18 -19 -18 -20 -17 -20 -18 -19 -19 -20 -19 -22 -21 -21 -20 -21-100-105-
 24  22  23  25  28  25  23  26  23  14   3   8   8 -14 -16 -13 -10  -7 -58 -75
 21  21  21  20  21  22  20  21  22  20  20  19  19  20  22  21  21  20  22  21
```

FIGURE 44 PART OF 120 x 120 LASER RANGEFINDER PRINTOUT FOR TOP RIGHT QUADRANT OF TEST PICTURE (FIGURE 43)

minus 0.33 inch, and across the first clear line above the top of the box from -130 to -135 or plus and minus 0.8 inch.

This picture has been used to test a program that generates a display of the range data with rotated axes. Thus Figure 45 shows a side view with the back wall, the vertical back flap of the box, and the vertical front face of the box edge on. Figure 45 would seem to be a convincing demonstration that the scanning rangefinder is measuring range in a manner consistent with the real world. The display is based on only 750 data points of the 14,400 measured.

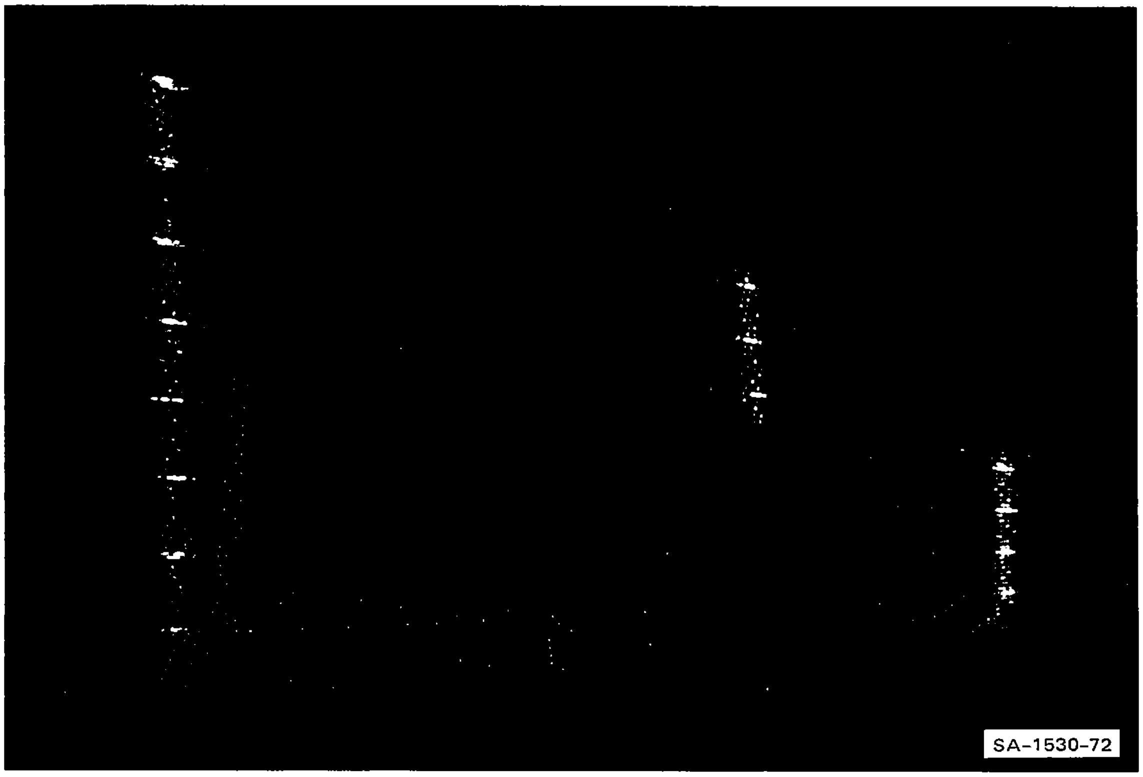


FIGURE 45 RANGE DATA DISPLAYED AS A SIDE VIEW OF THE WALL, REAR FLAP AND FRONT FACE OF THE BOX (COMPARE FIGURE 43 AND 44)

E. Conclusions

Figure 45 demonstrates that useful results are possible from the present experimental hardware, which seems to function in agreement with the theory presented. Both in theory and practice, performance in terms of data rate is limited by the number of photons received in the time interval allocated to each picture point. A high power laser will be necessary if it is desired to scan dark objects at a distance of 10 feet with 1-millisecond integration time per picture point.

VI. THE ARTIFICIAL INTELLIGENCE CENTER COMPUTER SYSTEM

A. Introduction

The past eighteen months has been a building and shoring operation for the TENEX related computer system software and hardware. The present status of the system is shown in Figure 46. Much effort has gone into understanding and using the variety of conveniences available to the research minded programmer who finds himself using the Virtual Machine offered by the BBN TENEX Hardware/Software.

The major software efforts can be broken into 4 areas: (1) PDP-10 TENEX System Software Maintenance/Customizing, (2) PDP-15 Operating System Complete Rework, (3) User level support software, and (4) Decision to use minicomputers to drive devices.

The major hardware efforts were performed in 3 areas: (1) Transfer of ARPANET Interface from PDP-15 to PDP-10, (2) Addition to disk storage capacity, and (3) Replacement/Increase of PDP-10 memory.

B. Software

1. PDP-10 Operation System

Much effort has gone into gaining a working understanding of the detailed operations of the TENEX operating system. The stimulus for

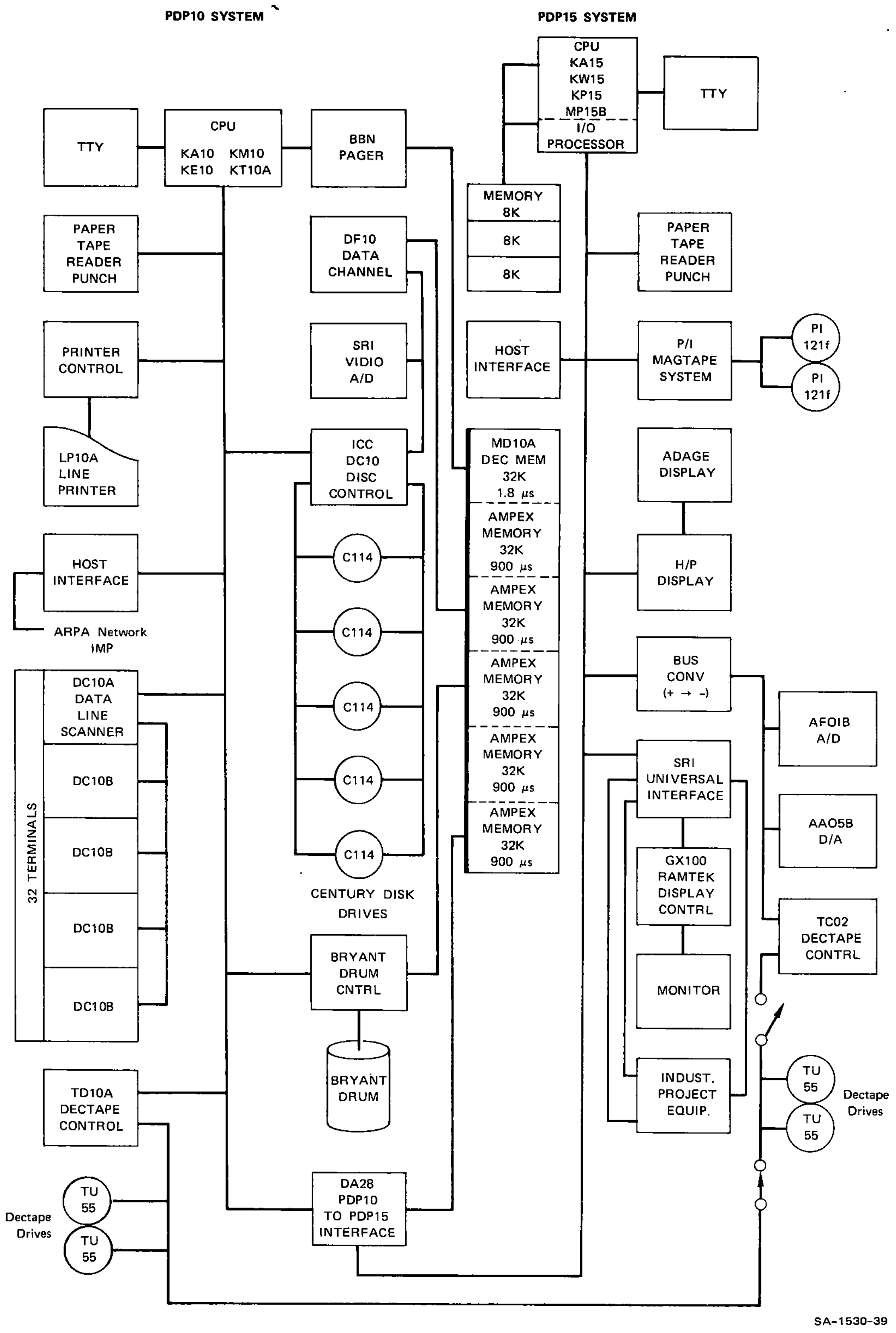


FIGURE 46 SRI AI CENTER COMPUTER CONFIGURATION

this arises from (1) the need to alter the operating system code to make additions to accomodate special devices, (2) unreliable hardware that has caused numerous file system crashes -- which are very difficult to unravel, and (3) the inherent inability of TENEX to support more than 6-8 active LISP programmers simultaneously.

We have accomplished the following specific tasks. (1) we have brought up the latest TENEX release (1.31). (2) We have modified drastically the priorities of the various hardware devices for interrupt levels in order to prevent data loss. (3) We have implemented the new (standard) IMP interface code. This was mainly an exercise in judicious removal of PDP-15 related software. (4) We have redesigned the 10/15 comunication protocols to gain robustness. (5) We have begun implementation of new higher level language processors for working with PDP-11 computers (mainly BCPL).

2. PDP-15 Operating System Rework

The PDP-15 computer has been used to control a variety of devices: displays, Shakey the Robot, tape drives, analog to digital converters, digital to analog converters, TV cameras, ARPANET IMP interface, range finder, laser pointer, and others. It was assumed that the use of a "peripheral computer" to drive the assortment of devices that arise in endeavours such as ours was a good idea from the standpoint of decoupling the operation of the timesharing computer (the PDP-10) from the devices its programs sought to control. The basic idea is, of course, valid. What we have found, though, is that the cost of writing software to operate a multitude of vastly dissimilar devices is immense. One starts out with a simple design for handling devices on an "as needed" basis, and after a few generations of devices have been added, the software that remains is incredibly complex. One can, of course, spend the time to try to understand all the interrelationships of the various devices and their possible conflicts with one another. Due to there being no other choice, we have done this with the PDP-15 computer. The removal of the IMP device has seen a vast improvement in the behavior of the ARPANET related software that so many users have daily need for. The removal of Shakey the Robot has caused the complexity of the remaining code to be decreased. In the past period we have had to modify the PDP-15 code drastically as follows: (1) we have modified the monitor to run in 8k instead of 4k, (2) we have instituted triple buffering to PDP-10 for D/A data, (3) we have changed 10/15 protocols to gain robustness, (4) we have introduced subdevices to allow multiple displays, (5) we have added code for controlling a range finder and laser-finder, laser-pointer, and (6) we have removed dead code.

3. Support Software

Due to the constant over subscription of on-line disk storage, there is a need to have a means of saving user files on tape when there is not enough room on the disk drives to hold all the data desired. Some programs had to be written to create a stopgap archiving system while we waited for the Backup System (BSYS) software to be implemented by the Augmentation Research Center at SRI. Other file system checking programs have been written and are run often to check on the consistency of the running file system. Accounting software that was borrowed from Bolt, Beranek, and Newman has been modified locally , but we have ceased working on it in anticipation of a Network wide accounting system.

C. Hardware

In an exerted effort to deliver reliable computing service to both local and network users, we have purchased an IMP interface for the PDP-10, placed an order for 256k words of PDP-10 main memory, and are going to receive the 9 disk drives from ISI this spring after they take satisfactory delivery of their new 3330 disk system. Upon the installation of the used drives here at SRI-AI we will have available approximately 115,000 pages of disk space for online storage of programs and data (a page holds 2560 characters). The "decision to use minis to drive devices" was listed under software efforts because it was mainly due to the software costs involved in writing operating systems to control a multitude of unrelated devices that we decided to take advantage of the recent low prices of minicomputers.

D. Limitations of TENEX

Two significant limitations have been fought in the TENEX Virtual Machine. First, the virtual address space of 256k words is getting to be small for the data structures that are being used now, so some amount of research effort is being spent on "compacting" and "overlaying" and other such methods. Second, the Fork mechanism for interprocess communication, while elegant and rich in its capabilities, has a high cost (in cpu time) per transaction. The cost is on the order of 5-10 milliseconds per handshake. For

Lisp/Fortran systems that do a lot of grinding on items and pass things back and forth this cost can be quite high. We are addressing this problem currently.

VII. PUBLICATIONS AND PRESENTATIONS

A. Publications

Binford, T., and J. M. Tenenbaum, "Computer Vision," Published in Computer Magazine, Vol. 6, No. 5 (May 1973).

Bobrow, D. G., and B. Raphael, "New Programming Languages for Artificial Intelligence Research," Artificial Intelligence Center, Technical Note 82, Stanford Research Institute, Menlo Park, California (August 1973).

Coles, L. S., et al, "Forecasting and Assessing the Impact of Artificial Intelligence on Society," Preprints, Third IJCAI, Stanford University (August 1973).

Derkson, J. A., J. F. Rulifson, and R. J. Waldinger, "QA4: A Procedural Calculus for Intuitive Reasoning," Artificial Intelligence Center, Technical Note 73, Stanford Research Institute, Menlo Park, California (November 1972).

----------"The QA4 Language Applied to Robot Problem Solving," Proceedings, Fall Joint Computer Conference (December 1972).

Fikes, R. E., "Themes in Automatic Problem Solving," Computer Magazine, Vol. 6, No. 5 (May 1973).

Garvey, T. D., et al, "An Interactive Facility for Scene Analysis Research," Artificial Intelligence Center, Technical Note 87, Stanford Research Institute, Menlo Park, California (January 1974).

Nilsson, N. J., "A Hierarchical Robot Planning and Execution System," Artificial Intelligence Center, Technical Note 76, Stanford Research Institute, Menlo Park, California (August 1973).

Nitzan, D., "Object Recognition in Multisensory Scene Analysis," Artificial Intelligence Center, Technical Note 83, Stanford Research Institute, Menlo Park, California (December 1973).

Reboh, R., and E. Sacerdoti, "A Preliminary QLISP Manual," Artificial Intelligence Center, Technical Note 81, Stanford Research Institute, Menlo Park, California (August 1973).

Sacerdoti, E., "P LANNING IN A Hierarchy of Abstraction Spaces," Proceedings Third IJCAI, August 1973 (to appear in A. I. Journal).

Tenenbaum, J. M., "On Locating Objects by their Distinguishing Features in Multisensory Images," Presented at First U.S.-Japan Seminar on Scene Analysis, Kyoto, July 1973. Reprinted in Computer Graphics and Image Processing (December 1973).

-------------- "Summary of Third Pajuro Dunes Workshop on Computer Vision," April 1973. Published in SIGART Newsletter (August 1973).

B. Presentations

December 7, 1972 Fall Joint Computer Conference

Nils Nilsson: "Review of A.I. Languages"

J. Derkson, J. Rulifson, and R. Waldinger: "The QA4 Language Applied to Robot Planning"

March 22, 1973 Talk at New York University

Nils Nilsson: "Hierarchical Robot Planning and Execution"

April 1, 1973 Talk at University of California, Berkeley

J. M. Tenenbaum: "Computer Vision"

May 1973 Talk at General Motors Research Labs

J. M. Tenenbaum: "Machine Perception"

May 14-16, 1973 Pajaro Dunes Workshop on Automatic Problem Solving (Supported by ONR Contract N00014-73-C-0245 but covering some material supported in part by ARPA)

Nils Nilsson, Organizer: "A Hierarchical Robot Planning and Execution System"

Richard Fikes: "Modelling, Question-Answering and Route Finding in a Robot World"

Earl Sacerdoti: "QLISP" and "ABSTRIPS"

Richard Waldinger: "New A.I. Language Features for Robot Planning and Automatic Programming"

May 24, 1973 Presentation at ARPA, Washington, D.C.

Richard Fikes: "SHAKEY, STRIPS, and Learning"

August 20-23, 1973 Third International Joint Conference on Artificial Intelligence

Nils Nilsson: Program Chairman

Bertram Raphael, coauthor (with D. Bobrow) of: "Languages for Artificial Intelligence"

Earl Sacerdoti: "Planning in a Hierarchy of Abstraction Spaces"

September 19, 1973 Lecture at IBM Research, San Jose, California

Richard Fikes: "A Deductive Retrieval System"

February 27, 1974 Talk at Joint Meeting of Los Angeles Chapters of ACM Groups

Earl Sacerdoti: "QLISP - A Programming Language for Artificial Intelligence"

February 1974 Talk at University of California, Davis

Peter Hart: "Artificial Intelligence and Robots"

February 1974 Talk at Stanford

Peter Hart: "The Computer-Based Consultant"

March 7, 1974 Lecture at University of Texas

Richard Fikes: "Modelling and Planning Mechanisms for a Computerized Consultant"

APPENDIX A

DESCRIPTION OF A SIMPLE AIR COMPRESSOR

Appendix A

DESCRIPTION OF A SIMPLE AIR COMPRESSOR

The air compressor we have been working with is a Sears Model 17209 1/2-hp compressor; it is illustrated in Figure A-1.

The essential elements of this unit are:

a. Electric Motor--drives the air pump.

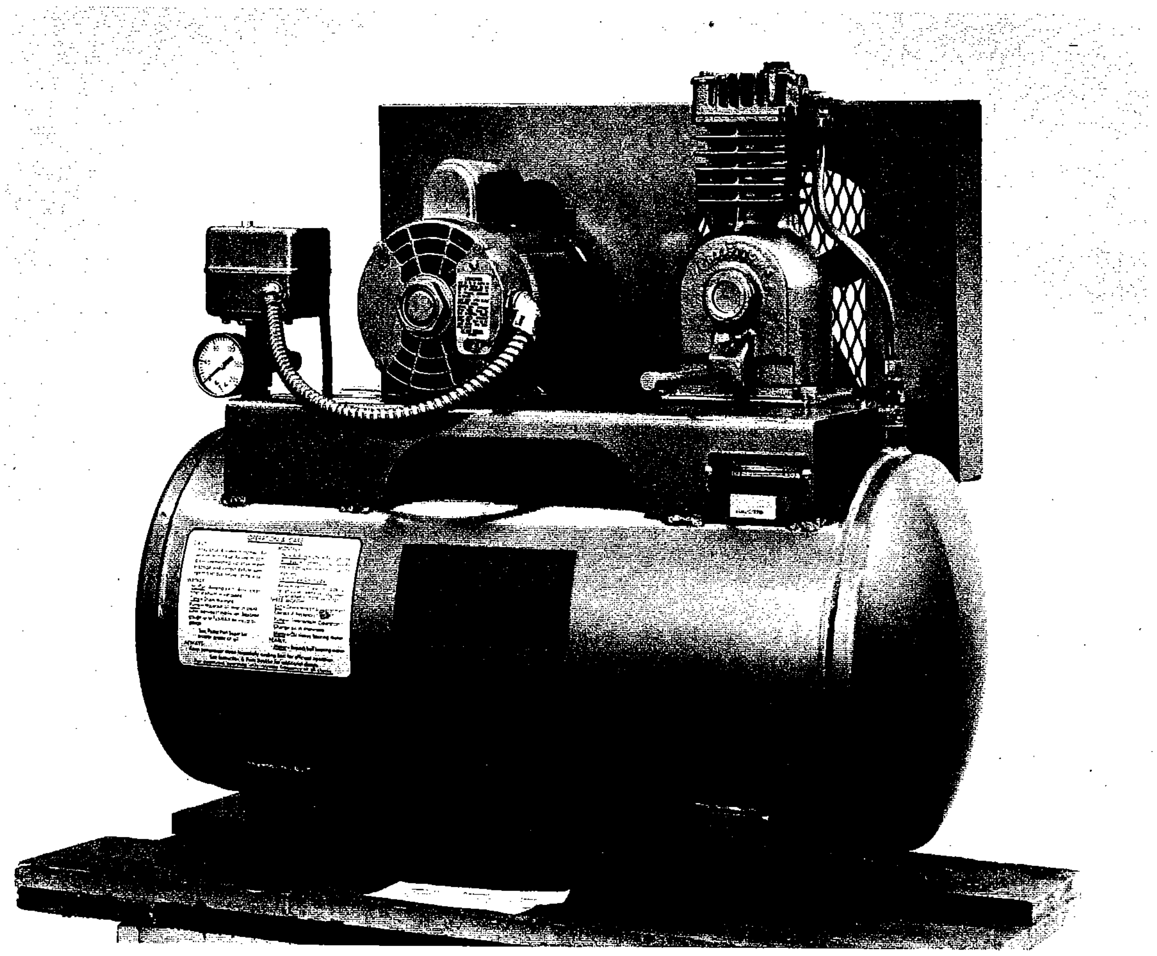
b. Pump--takes air from atmosphere and compresses it into tank.

c. Tank--stores compressed air.

d. Pressure Switch--starts and stops the motor at predetermined pressures.

e. Relief Valve--releases "back" pressure on pump when the motor stops.

f. Check Valve--holds air in the tank when compressor stops, thus allowing the relief valve to let air out of piping.



SA-1530-70

FIGURE A-1 AN AIR COMPRESSOR

We quote from the Sears Operating Instructions to describe the major features:

PUMPING. In action, when the piston of the pump goes down, air rushes in through the inlet valve. As the piston starts up the inlet valve closes and the air in the cylinder is compressed until the pressure in the cylinder slightly exceeds the pressure in the piping (aftercooler) connected to the check valve. This excess pressure causes the outlet valve to open, allowing the air in the cylinder to be forced out through the outlet valve into the piping. As the piston starts down, the outlet valve closes, preventing air in the aftercooler from rushing back into the cylinder and the cycle starts over again. When the pressure in the aftercooler exceeds that in the tank, the check valve opens and allows air to pass into the tank.

STOPPING. The compressor continues to pump until the pressure in the tank reaches the point at which the automatic switch is set to cut out. The pressure in the tank acts on the diaphragm of the automatic switch (through piping) and when the cut-out point is reached, this diaphragm distends and, through levers, opens switch contacts stopping the motor, and simultaneously opening the relief valve. The air trapped between the pump and the check valve is thus allowed to escape through the relief valve. The air in the tank cannot, of course, escape because the check valve does not allow the air in the tank to flow back.

STARTING. When the pressure in the tank is decreased by using the air. and finally reaches the "cut-in point" of the switch, the reduced pressure on the diaphragm allows the pressure spring to cause the lever to close the switch contacts and relief valve. This starts the motor and the compressor again begins to pump. There being no pressure in the aftercooler (it having escaped when the relief valve was opened) the motor can start under no pumping load and get up to speed during the time required to raise pressure in the aftercooler.

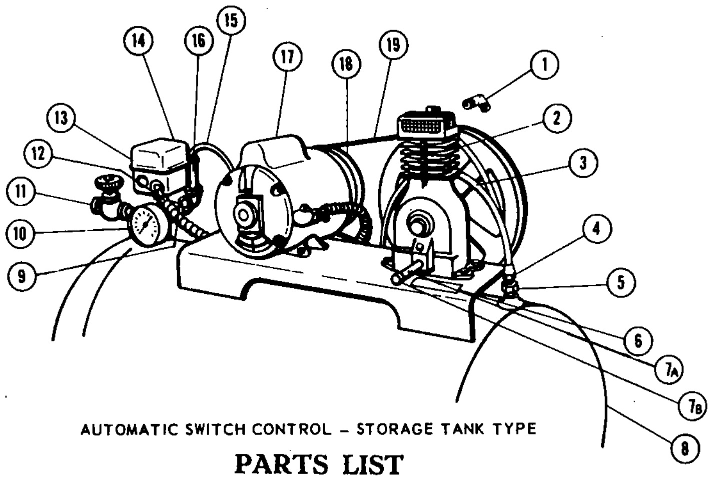
From the foregoing and from an examination of the diagram, it will be seen that not only must the electric motor and air compressor itself be kept in proper working condition, but that the check valve and relief valve must work properly. If the check valve leaks, then the air stored in the tank will escape through the relief valve when the compressor is not running because the relief valve is open all the time the switch contacts are open.

Also, if the relief valve does not close properly when the compressor is running, the air (or most of it) pumped by the compressor will not go into the tank at all, but will pass out through the leaky valve. Or, if the relief valve does not open properly then the back pressure cannot be relieved and the motor must start the compressor against pressure, which consumes extra power and may damage the motor.

The major parts of the compressor are listed in Figure A-2, taken from the Sears Manual. A parts list for the pump unit is given in Figure A-3.

Model Numbers

(17199) 102.17200 102.17220 (17218) 102.17021
(17209) 102.17210 102.17001 102.17046
102.17011 102.17041



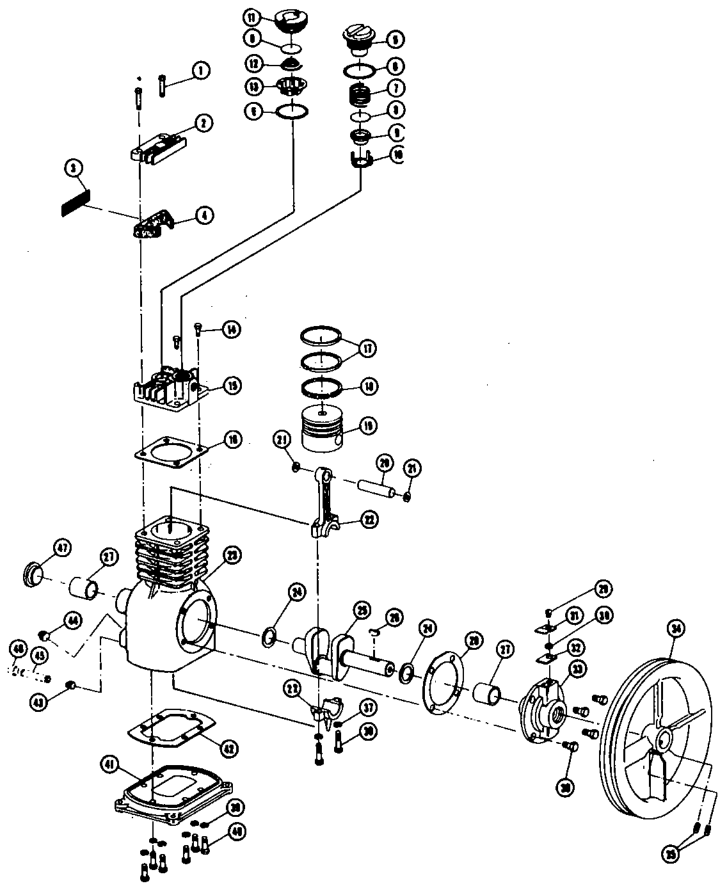
AUTOMATIC SWITCH CONTROL – STORAGE TANK TYPE

PARTS LIST

Reference Number	Part Number	NAME OF PART
1	690	Aftercooler Elbow 3/8" Flare Tube x 1/4" Pipe
2	-	Pump – See Figure 9
3	37055	Aftercooler for 102.17001, 102.17011, 102.17021, 102.17046 and 102.17041
3	37057	Aftercooler for (17199) 102.17200, (17209) 102.17210, and 102.17220 (17218)
4	98556	Elbow Connector–1/4" Tube x 1/8" Pipe Thread
5	15681	Check Valve–See Figure 4 for Parts
6	12776	Nameplate
7a	9609	Oil Drain Nipple–1/8" x 5" Long
7b	14292	Oil Drain Cap–1/8"
8	43779	Tank and Platform Assembly for (17199) 102.17200, (17209) 102.17210, 102.17220 (17218)
8	43780	Tank and Platform Assembly to Massachusetts Specifications for (17199) 102.17200, (17209) 102.17210, and 102.17220 (17218)
8	43781	Tank and Plarform Assembly 102.17001 and 102.17011. Use for Massachusetts also.
8	43782	Tank and Platform Assembly for 102.17021, 102.17046 and 102.17041
8	43783	Tank and Plarform Assembly to Massachusetts Specifications for 102.17021, 102.17046 and 102.17041
9	16614	Manifold for mounting Pressure Switch, etc. for 102.17021, 102.17046 and 102.17041. Use for all Mass. Specifications
9	43724	Manifold for mounting Pressure Switch and Gauge, (Safety Valve and Discharge Valve included) for (17199) 102.17200, (17209) 102.17210, 102.17220 (17218) and 102.17001. Not for Massachusetts Specifications.
10	12571	Pressure Gauge for 102.17001, 102.17011, 102.17021, 102.17046 and 102.17041
10	15793	Pressure Gauge for (17199) 102.17200, (17209) 102.17210, and 102.17220 (17218)
10	17088	Pressure Gauge for State of Massachusetts
11	95939	Discharge Valve for 102.17001, 102.17011, 102.17021, 102.17046 and 102.17041. Also used on Mass. Specification models (17199) 102.17200, (17209) 102.17210 and 102.17220 (17218)
12		BX conduit–Available locally (See Installation Instructions for wire size)
13	12191	Pressure Switch for (17209) 102.17210, 102.17220 (17218) 102.17011, 102.17021 and 102.17041
13	13858	Pressure Switch for 102.17046
13	32473	Pressure Switch for (17199) 102.17200
14	-	Pressure Switch Relief Valve. When ordering Relief Valve give Pressure Switch Nameplate Data. (Ref. No. 13)
15	18591	Check Valve Relief Tube (also available locally 1/4" aluminum or copper tube 31½ long.
16	18550	Safety Valve for 102.17001, 102.17011, 102.17021, 102.17046 and 102.17041
16	18544	Massachusetts Safety Valve for (17199) 102.17200, (17209) 102.17210, 102.17220 (17218), 102.17001, 102.17011, 102.17021, 102.17046 and 102.17041
17	-	Motor or Gas Engine–See Motor Nameplate and Gas Engine Part Booklet for characteristics; when writing regarding motor or engine service include complete nameplate data.
18	45053	Motor Pulley for (17199) 102.17200
18	45058	Motor Pulley for (17209) 102.17210 and 102.17021
18	45055	Motor Pulley for 102.17001 and 102.17011
18	45059	Motor Pulley for 102.17220 (17218) and 102.17041
18	30545	Motor Pulley for 102.17046
19	17469	Belt for 102.17001 (½" wide x 44" long)
19	17470	Belt for 102.17220 (17218), 102.17021 and 102.17041 (½" wide x 47" long)
19	18356	Belt for (17209) 102.17210 (½" wide x 46" long)
19	18481	Belt for 102.17011 and 102.17046 (½" wide x 45" long)
19	32622	Belt for (17199) 102.17200 – ½" wide x 43" long
Not Shown	96570	Aftercooler Vent Coco-Gas Engine Only
Not Shown	96800	Tank Drain Valve
Not Shown	13231	Tank Drain Valve Only for State of Mass.
Not Shown	43429	Cord and Plug Assembly for 102.17301, 102.17400 and 102.17410
Not Shown	43430	Adapter Ground Plug for 102.17304 102.17400 and 102.17410

SA-1530-56

FIGURE A-2 PARTS LIST FOR AIR COMPRESSOR



Reference Number	Part Number	NAME OF PART	Quan. Used	Reference Number	Part Number	NAME OF PART	Quan. Used
1	96472	Head Bolt	2	30	21614	Breather Valve Spacer	1
2	45007	Cylinder Head Cover	1	31	38941	Breather Valve Plate	1
3	45020	Filter Screen	1	32	38944	Breather Valve	1
4	45021	Filter	1	33	45003	End Cover Assembly-includes Ref. No. 27	1
5	45024	Outlet Valve Stop	1	34**	45025	Pulley for 101 TV	1
6	45014	Copper Washer	2	35	98627	Pulley Set Screw-$\frac{5}{16}$-18 x $\frac{1}{2}$" Socket Head	2
7	45011	Outlet Valve Spring	1	36	18060	End Cover Bolt-$\frac{1}{4}$-20 x $\frac{1}{2}$"	4
8	45013	Valve	2	37	9522	Lockwasher-$\frac{1}{4}$"	2
9	45016	Outlet Valve Seat	1	38	11109	Hex Head Bolts-$\frac{1}{4}$-20 x 1"	2
10	45019	Outlet Valve Retainer	1	39	9522	Lockwashers-$\frac{1}{4}$"	6
11	45017	Inlet Valve Seat	1	40	980	Hex Head Bolts-$\frac{1}{4}$-20 x $\frac{3}{4}$" long	6
12	45012	Inlet Valve Spring	1	41	45006	Base Plate	1
13	45018	Inlet Valve Retainer	1	42	45009	Base Plate Gasket	1
14	18061	Head Bolt	2	43	9349	Pipe Plug $\frac{1}{8}$"-Furnished with 102.17500 only	1
15	45005	Cylinder Head	1	44	9348	Pipe Plug $\frac{1}{4}$"	1
16	45008	Cylinder Head Gasket	1	45	9609	Pipe Nipple-$\frac{1}{8}$" x 5"-Furnished with compressor only	1
17	31901	Piston Rings	2	46	14292	Pipe Cap-$\frac{1}{8}$" Furnished with compressor only	1
18	17083	Oil Ring	1	47	45027	Crankcase Plug	
19	38932	Piston	1	Not Shown	43797	Parts List	.
20	12992	Wrist Pin	1	**	45050	Pulley for 100 TV	1
21	7890	Wrist Pin Retainer Washer	2	*	45051	Crankshaft for 100 TV	1
22	45001	Connecting Rod Assembly	1	Not Shown	45060	Head Plate Spacer	2
23	45004	Crankcase Assembly-includes Ref. No. 27	1	Not Shown	45244	Set-Gaskets	1
24	45023	Thrust Washer	2	Not Shown	45237	Set-Rings	1
25*	45002	Crankshaft for 101 TV	1	Not Shown	45245	Set-Valve & Spring	1
26	9397	Woodruff Key No. 9	1				
27	45026	Bronze Bushing	1				
28	45010	End Cover Gasket	1				
29	38951	Breather Valve Screw-8-32 x $\frac{5}{16}$"	1				

SA-1530-57

FIGURE A-3 PARTS LIST FOR THE PUMP UNIT—MODEL NUMBER 102:17500 (17501)

APPENDIX B

COST AND CONFIDENCE COMPUTATIONS

Appendix B

COST AND CONFIDENCE COMPUTATIONS

In planning strategies, it is sometimes necessary to know the expected cost and resulting recognition confidence associated with a sequence of tests. In this appendix, we derive the cost of acquiring samples of some desired object by filtering with a given set of tests, and we derive the confidence that the acquired samples actually belong to the desired object.

In these computations, the system utilizes characterizations of surface attributes (such as HUE and HEIGHT) which are stored as histograms of probability values. A 'bucket' of the histogram contains, in general, a list of objects, and for each, the probability that the measured attribute from a given sample of the object will fall within that bucket. A detector, D_i, specifies an attribute and a range of buckets in the histogram of that attribute. The detector is converted into a predicate which examines a sample from a picture and if the value of the measured attribute falls within the range of values specified by the detector accepts it, otherwise, the sample is rejected. A set of detectors, $\{D_n, D_{n-1}, \ldots, D_j\}$ is written, D_j^n, with the convention that $D^n = D_1^n$.

Confidence $P(O|D^n)$, is defined as the probability that a sample belongs to some object, O, given that it has been accepted by the set of detectors, D^n.

The expression for confidence is expanded by a recursive Bayes procedure as follows:

$$P(O|D^n) = \frac{P(D_n|O)*P(O|D^{n-1})}{P(D_n|D^{n-1})}$$

$P(D_n|O)$ is the percentage of samples of O that will be accepted by D_n. In order to compute $P(D_n|D^{n-1})$, the probability that detector D_n will also accept a sample already accepted by the set of detectors, D^{n-1}, it must be noticed that the detectors are linked through the object descriptions. Therefore, the probability is computed by expanding over all objects in the model:

$$P(D_n|D^{n-1}) = \sum_{O_i} P(D_n|O_i)*P(O_i|D^{n-1})$$

That is, the probability of D_n accepting a sample already accepted by D^{n-1} is equal to the probability of D_n accepting a sample belonging to an object, O_i, times the probability of the sample belonging to O_i, given that the detectors D^{n-1} have accepted it.

$P(O_i|D^{n-1})$ is computed recursively, terminating with $P(O_i|D_1)$. This term is rewritten in a straightforward way:

$$P(O_i|D_1) = \frac{P(D_1|O)*P(O)}{P(D_1)}$$

The computation of $P(D_1)$ is similar to that for $P(D_n|D^{n-1})$.

$$P(D_1) = \sum_{O_i} P(D_1|O_i)*P(O_i)$$

$P(O_i)$ is the a priori likelihood of a randomly selected sample belonging to object O_i. It is computed as the ratio of the expected two-dimensional size of O to the size of the window, W, being examined (with the implicit assumption that O appears in the window exactly once. That is,

$$P(O_i) = \frac{\text{expected size of } O_i}{\text{size of W}}$$

To reiterate:

$$P(O|D^n) = \frac{P(D_n|O)*P(O|D^{n-1})}{P(D_n|D^{n-1})}$$

$P(D_n|O)$ = percentage of samples of O accepted by D_n.

$$P(D_n|D^{n-1}) = \sum_{O_i} P(D_n|O_i)*P(O_i|D^{n-1})$$

$$P(O_i|D_1) = \frac{P(D_1|O_i)*P(O_i)}{P(D_1)}$$

$$P(D_1) = \sum_{O_i} P(D_1|O_i)*P(O_i)$$

$$P(O_i) = \frac{\text{expected size of } O_i}{\text{size of } W}$$

The next parameter to be computed is the expected cost of application of a set of detectors. Here, the set of detectors is ordered, since the outcome of a detector determines whether succeeding detectors are actually applied.

It is assumed that the first detector, D_1, will be applied to all samples in the window, W. Any sample accepted by D_1 will then be tested with D_2, and so on until all detectors have accepted the sample, or until it is rejected. Therefore, the cost of applying a sequence of detectors to a sample is written:

$$C_s(D^n) = C_s(D_1) + C_s(D_2)*P(D_1) + C_s(D_3)*P(D_1,D_2) + \dots$$
$$= \sum_{i=1}^{n} C_s(D_i)*P(D_1^{i-1}).$$

where $C_s(D^n)$ is the cost per sample of the sequence, and $C_s(D_i)$ is the cost per sample of the i-th detector.

To complete the calculation of cost requires the computation of the number of samples to be tested. Given that it is desired that some predetermined number of samples, N_D, on the surface being acquired should be accepted by the detector set, D^n, it is possible to compute a sampling density, δ. The total expected number of accepted samples on the object, N_T, is obviously a function of the expected object size S(O). In fact,

$$N_T = \delta*S(O)$$

But since only a fraction of any random sampling of O can be expected to be accepted by D^n (in fact, only $P(D^n|O)$ of them),

$$N_D = N_T*P(D^n|O).$$

Therefore,

$$N_D = \delta * S(0) * P(D^n | 0)$$

and

$$\delta = \frac{N_D}{S(0) * P(D^n | 0)}$$

Since the system does not retain the correlated data necessary to determine $P(D^n|0)$, it instead uses the minimum of the set of probabilities, $\{P(D_1|0), P(D_2|0), \ldots, P(D_n|0)\}$.

The total number of samples to be checked is the window size

$S(W)$ times δ.

That is

$$N_W = \delta * S(W).$$

giving a total cost of

$$C(D^n) = N_W * C_S(D^n).$$

$$= N_D \cdot \frac{S(W)}{S(0)} \cdot \frac{C_S(D^n)}{P(D^n|0)} = N_D \frac{C_S(D^n)}{P(0)\ P(D^n|0)} = \frac{N_D\ C_S(D^n)}{P(0|D^n) \cdot P(D^n)}$$

In this system, the costs per sample, are expressed in milliseconds of computation time required to perform the measurement. These costs can be computed beforehand for each measurement, as they vary relative to one another as a function of system load.

It should be pointed out that we could have defined $P(0|D^n)$ as the probability that a sample belongs to 0 given the outcomes of the set of detectors, D^n, applied to it. This changes none of the confidence computations, and allows the system to generate sample decision trees where the next test to be performed is determined by the outcome of the last test. This allows for the handling of situations where it might be easier to eliminate a sample by planning for a test to reject rather than accept it. It also allows the cascading of individually indeterminate tests to gradually increase confidence to a suitable level, without worrying that a single failure will cause the test to fail. The computations of cost for this definition will be more complex, since they must account for each branch of the tree and the expected frequency with which each one will be taken. We have not yet programmed the system to generate detectors which are not simple conjunctions, but will do so in the future.

APPENDIX C

XDEMON: A CONSTRAINT SATISFIER

Appendix C

XDEMON: A CONSTRAINT SATISFIER

A. Implementation Details

A constraint satisfaction system has been implemented utilizing cooperating independent processes coupled through a global data base. The data base consists of a set of VARIABLES (i.e., records), each containing four components:

a FORM (procedure in the XDEMON program)

a VALUE

a set of RELATIVES (a list of related variables)

VARPROPS (property list).

There are selecting and updating functions for each of the above components. The form of a variable is an internal representation of some s-expression. This internal representation, known as an h-expression, is composed of either a LISP atom, or a list of other variables. Each variable corresponds to a particular s-expression: in LISP a variable corresponds to a particular atom, and we have generalized this notion [Ref. 6]. For example, (FOO X Y) has a corresponding variable, as do FOO, X and Y. So also does (FIE (FOO X Y) Z). As in LISP, character strings are normalized to yield a unique internal representation--the variable.

(NORMEXPR [expression]) is a function that returns the variable corresponding to the given expression.

Variables are initialized to have the value Undef and Nil for relatives and varprops.

A collection of variables may be linked to form a network. (CONNECT [variable]) puts [variable] on the lists of relatives of all the variables in its form. (CONNECT! [variable]) does the same thing recursively for the variables in the form as well. An example is given in Figure C-1. There are corresponding inverse functions DISCONNECT and DISCONNECT!.

Note that pointers to subexpressions are available via the form and pointers to super-expressions via the relatives.

The value of a variable is normally set by the function HSET. (HSET [variable] [value]) returns [value] as its result. HSET is executed for its side-effects: if the new value is the same as the old value, (under the equivalence HSETEQ, initially EQUAL), nothing happens: if the new value is different, then the variable is evaluated (or rather, the evaluation of the related variable is added to a list of jobs to be run). Note that other schemes could be used here; e.g., some relatives might be evaluated before the variable is reset.

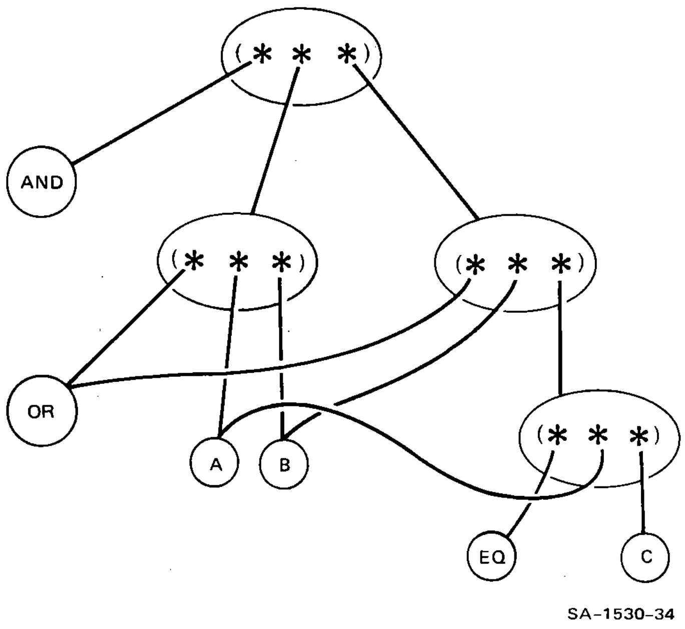


FIGURE C-1 REPRESENTATION FOR THE EXPRESSION
(CONNECT!
(NORMEXPR
(QUOTE
(AND(OR A B)
(OR B (EQ A C))))))

Variables are evaluated by (HEVAL /variable/). HEVAL evaluates the form of the variable, and then HSETs the variable to the new value, perhaps causing its relatives to be evaluated. It returns the new value as its result. The value of an atomic H-expression is the value of the variable: otherwise, the value is the result of APPLYing the value of the CAR of the H-expression to the CDR.

Note that unlike LISP the definition of a function is kept in the value of a corresponding atomic variable, not in a special cell. Definitions are established by executing (HDEF /function definition/). /function definition/ is a list of two elements, the name and the body (as for DEFINE). The body can be the name of a lisp function, or a lambda expression. Note also that when a variable is HEVALuated, the evaluation is not recursive; the immediate values of the variables forming the H-expression are utilized without HEVALuating them.

It should be clear that variables are only evaluated when the value of something to which they are relatives is changed, never just because a variable higher in the expression is evaluated. Values are thus remembered and not recomputed unnecessarily.

A list of outstanding jobs to be run is held in the global variable JOBLIST. Those jobs are executed by a call (RUNJOBS), which will successively execute and then delete the jobs on the list. RUNJOBS terminates when there are no jobs left (including those which have been added dynamically).

Readers familiar with SRI's QLISP may be wondering why the constraint satisfier was not implemented in that system using canonic net expressions instead of H-expressions. There are three reasons, all manifestations of the present state of QLISP development.

1. H-expressions are simple n-tuples. The QLISP discrimination net handles n-tuples relatively inefficiently, due to the generality required to accommodate less constrained data types (i.e., sets and bags).

2. Expressions cannot directly reference the canonic representation of other expressions in the net. Access to nodes is available only by dropping the referenced expression through the discrimination mechanism.

3. QLISP is a very large system. Without overlays, there is inadequate space to both store a large number of semantic constraints and represent a scene with hundreds of regions.

If current QLISP development, partially supported by NASA, succeeds in overcoming these objections, it is likely that XDEMON will be reimplemented to take advantage of features such as associative retrieval, pattern matching, contexts, and processes.

B. Applications

The XDEMON system has been used to solve a variety of simple problems, including a cryptarithmetic puzzle and instant insanity. The system was also used to run Waltz's filtering algorithm [Ref. 20] on the artificial room scene previously used by Duda [Ref. 16] to illustrate interpretation by tree search. Altogether, 343 different interpretations were found to be consistent with the given semantic constraints on legally adjacent objects. (This result indicates that adjacency by itself does not provide sufficient constraint.) The Waltz algorithm was implemented in a half day using the system primitives described above and a single short function written to test the legality of adjacent interpretations.

REFERENCES

1. P. E. Hart et al., "Artificial Intelligence--Research and Applications," Annual Technical Report to ARPA, Contract DAHC04-72-C-0008, Stanford Research Institute, Menlo Park, California (December 1972).

2. N. J. Nilsson et al., "Plan for a Computer-Based Consultant," Draft Technical Report, Stanford Research Institute, Menlo Park, California (January 1974).

3. R. E. Fikes and N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving in Problem Solving," Artificial Intelligence, Vol. 2, pp. 189-208 (1971).

4. R. E. Fikes, P. E. Hart, and N. J. Nilsson, "Learning and Executing Generalized Robot Plans," Artificial Intelligence, Vol. 3, pp. 251-288 (1972).

5. E. D. Sacerdoti, "Planning in a Hierarchy of Abstraction Spaces," Advance Papers of the Conference, 3rd International Joint Conference on Artificial Intelligence, Stanford University, Stanford, California, August 1973. To appear in Artificial Intelligence, 1974.

6. J. F. Rulifson, J. A. Derkson, and R. J. Waldinger, "QA4: A Procedural Calculus for Intuitive Reasoning," Technical Note 73, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California (1972).

7. R. Reboh and E. D. Sacerdoti, "A Preliminary QLISP Manual," Technical Note 81, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California (August 1973).

8. D. G. Bobrow and B. Wegbreit, "A Model for Control Structures for Artificial Intelligence Programming Languages," Advanced Papers of the Conference, 3rd International Joint Conference on Artificial Intelligence, Stanford University, Stanford, California (August 1973).

9. G. Hendrix, "Modeling Simultaneous Actions and Continuous Processes," Artificial Intelligence, Vol. 4, pp. 145-180 (1973).

10. C. Green, "The Application of Theorem-Proving to Question-Answering Systems," Doctoral Dissertation, Elec. Eng. Dept., Stanford University, Stanford, California, June 1969. Also printed as Stanford Artificial Intelligence Project Memo AI-96 (June 1969).

11. M. R. Quillian, "Semantic Memory," Semantic Information Processing MIT Press, Cambridge, Massachusetts (1968).

12. T. Winograd, "Procedures as Representation for Data in a Computer Program for Understanding Natural Language," Technical Report AI TR-231, Artificial Intelligence Laboratory, MIT, Cambridge, Massachusetts (1970).

13. G. J. Sussman, "A Computational Model of Skill Acquisition," Technical Note AI TR-297, Artificial Intelligence Laboratory, MIT. Cambridge, Massachusetts (August 1973).

14. B. G. Deutsch, "The Structure of Task Oriented Dialogs," Proceedings, IEEE Speech Symposium, Carnegie-Mellon University, Pittsburgh, Pennsylvania, Technical Note 90, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California (April 1974).

15. J. M. Tenenbaum et al., "An Interactive Facility for Scene Analysis Research," Technical Note 87, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California (January 1974). Summary to appear in Proceedings, 2nd International Joint Conference on Pattern Recognition, Copenhagen, Denmark (August 1974).

16. R. O. Duda, "Some Current Techniques for Scene Analysis," Technical Note 46, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California (October 1970).

17. A. Guzman, "Analysis of Curved Line Drawings Using Context and Global Information," Machine Intelligence, Vol. 6, B. Meltzer and D. Michie, eds., pp. 325-376, Edinburgh University Press, Edinburgh, Scotland (1971).

18. F. P. Preparata and S. R. Ray, "An Approach to Artificial Non-Symbolic Cognition," Information Science, Vol. 4, pp. 65-86 (1972).

19. Y. Yakimovsky and J. A. Feldman, "A Semantics-Based Decision Theory Region Analyzer," Advance Papers of the Conference, 3rd International Joint Conference on Artificial Intelligence, Stanford University, Stanford, California (August 1973).

20. D. G. Waltz, "Generating Semantic Descriptions from Drawings of Scenes with Shadows," A.I. TR-271, Artificial Intelligence Laboratory, MIT, Cambridge, Massachusetts (August 1972).

21. C. R. Brice and C. L. Fennema, "Scene Analysis Using Regions," Artificial Intelligence, Vol. 1, No. 3, pp. 205-226 (1970).

22. D. Nitzan, "Scene Analysis Using Range Data," Technical Note 69, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California (August 1972).

23. R. E. Fikes, "REF-ARF: A System for Solving Problems Stated as Procedures," Artificial Intelligence, Vol. 1(1), Technical Note 14, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California (September 1969).

24. W. H. Paxton, "A Best-First Parser," Proceedings, IEEE Symposium on Speech Recognition, Carnegie-Mellon University, Pittsburgh, Pennsylvania, p. 218 (April 1974). Technical Note 92, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California (May 1974).

25. D. R. Reddy et al., "The Hearsay Speech Understanding System; An Example of the Recognition Process," Advance Papers of the Conference, 3rd International Joint Conference on Artificial Intelligence, Stanford University, Stanford, California (August 1973).

26. R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis, John Wiley & Sons, New York (1973).

27. -------------------, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," CACM, Vol. 15, No. 1, p. 11 (January 1972).

28. E. C. Charniak, "Towards a Model of Children's Story Comprehension," AI TR-266, MIT, Cambridge, Massachusetts (1972).

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Stanford Research Institute, 333 Ravenswood Avenue, Menlo Park, California 94025	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP n/a

3. REPORT TITLE

ARTIFICIAL INTELLIGENCE--RESEARCH AND APPLICATIONS

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Progress Report: 9 October 1972 through 8 March 1974.

5. AUTHOR(S) *(First name, middle initial, last name)*
Nils J. Nilsson, Alfred E. Brain, Thomas D. Garvey, J. Martin Tenenbaum, Gerald Agin, Barbara Deutsch, Daniel C. Lynch, Harry Barrow, Richard E. Fikes, Earl D. Sacerdoti

6. REPORT DATE May 1974	7a. TOTAL NO. OF PAGES 226	7b. NO. OF REFS 28

8a. CONTRACT OR GRANT NO. DAHC04-72-C-0008	9a. ORIGINATOR'S REPORT NUMBER(S) SRI Project 1530
b. PROJECT NO. 1530	
c. Program Code No. 2D30	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d. ARPA Order No. 1943	

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Defense Advanced Research Projects Agency, Arlington, Virginia 22209

13. ABSTRACT

This progress report describes the work performed during the most recent year and a half of a program of research in the field of Artificial Intelligence (AI) being conducted at SRI. Our research program concentrates especially on the development of systems that can automatically generate and execute complex plans and that can obtain information about their environments through the sense of vision.

To provide a specific focus for our research, we have set ourselves the goal of building a "computer-based consultant" system that will serve as an expert consultant to a human apprentice. Together the system and the apprentice will be engaged in a task of checking out and repairing electromechanical equipment in a "workstation" domain. We have developed a plan in which we have specified the capabilities of a system that can be demonstrated after five years of work (1978), and in which we have identified and discussed the major research problem areas involved. These include modeling and representation, automatic planning and problem solving, machine vision, natural language communication, and system integration.

The report describes progress we have made in these areas.

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Computer-Based Consultant						
Artificial Intelligence						
Machine Vision						
Problem Solving						
Modeling						
Laser Range Finder						
Scene Analysis						
Natural Language						