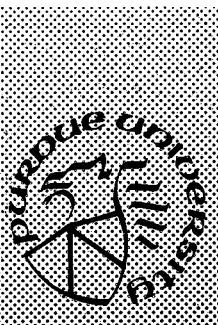
3-1-1985

Some Prototype Examples for Expert Systems v.2

K. S. Fu
Purdue University



Some Prototype Examples for Expert Systems

edited by
K.S. Fu

Volume II

TR-EE 85-1
March 1985

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

Table of Contents

FORWARD

This report consists of the nineteen term project reports for the graduate-level course EE695G "Expert Systems and Knowledge Engineering", which was offered for the fall semester of 1984 in the School of Electrical Engineering. The purpose of the term project is to provide each student an opportunity of designing and implementing a prototype expert system. The application area of each of these expert systems was selected by the student(s) working on the projects. This report is published for the purpose of documenting these results for future reference by the students of the above-mentioned course and, possibly, other workers in expert systems.

The nineteen reports are grouped into seven parts based on their application domains. Part I - Manufacturing consists of six reports, and Part II - Robotics contains three. Two reports in each of Part III - Vision and Part IV - Management, and one in each of Part V - Structural Engineering and Part VI - Automatic Programming. The last part, Part VII - Others, consists of four reports with different applications.

I would like to thank Mr. Edward K. Wong for his valuable help in putting the materials together for this report.

K. S. Fu
Instructor, EE695G
February 1985
Lafayette, Indiana

****

PART II

Robotics

Chapter 7

MR1: An Expert System for Configuration of Modular Robots

D. Dutta and S. Joshi

MR1: AN EXPERT SYSTEM FOR CONFIGURATION OF MODULAR ROBOTS

DEBASISH DUTTA
SANJAY JOSHI

1. INTRODUCTION

1.1 Expert Systems

Among the several areas in the field of artificial intelligence which have been very successful over the past decade, is the development of powerful new computer systems known as expert systems designed to represent and apply factual knowledge of specific areas of expertise to solve problems [A1]. An expert system can be described as a problem-solving computer program that achieves a high level of performance in a specialized problem domain, considered to be difficult and requiring specialized knowledge and skill. It has also been referred to as a knowledge-based system because its performance depends very much on the usage of facts and heuristics used by human experts while solving similar problems.

The usage of symbolic representations, symbolic inferences and heuristic searches, distinguishes general artificial intelligence applications from conventional data processing. But expert systems differ from the broad class of artificial intelligence tasks in several respects. Performance of difficult tasks at expert levels, employment of self-knowledge for reasoning and the emphasis of domain specific problem-solving strategies over the more general methods of artificial

intelligence, are some of the salient differentiating features of expert systems [A1]. Till date, expert systems have been used in a fairly wide variety of problem domains with remarkable success.

The following table illustrates the various categories of expert systems and their applications [A1].

Category	Associated Problem
Interpretation	Inferring situation description from sensor data
Prediction	Inferring likely consequences of given situation
Diagnosis	Inferrring system malfunction from observations
Design	Configuring objects under constraints
Planning	Designing actions
Monitoring	Comparing observations to plan vulnerabilities
Debugging	Prescribing remedies for malfunction
Repair	Executing a plan to administer a prescribed remedy
Instruction	Diagnosing, debugging, repairing student behavior
Control	Interpreting, predicting, repairing and monitoring system behavior

This report describes MR1 (Modular Robot, version 1), an expert system developed for EE 695D class project at Purdue University. Its intended area of application is the configuration of modular robots.

1.2

Modular Robots

The Robotics Institute of America defines an industrial robot as: "a reprogrammable, multifunction manipulator designed to move material, parts, tool or specialized devices through variable program motions for the performance of a variety of tasks". Modular as defined by the Websters New Collegiate Dictionary is: "constructed with standardized units or dimensions for flexibility and variety in use". Thus the combination of these words in the phrase "modular industrial robots" brings to light that these robots besides being reprogrammable, are reconfigurable too.

The area of modular robots is an out growth of the design of automatic assembly equipment. It was realized that machines usually move the workpiece in either straight lines or in a circular fashion. In an attempt to reduce the cost of automatic assembly equipment and to assist in the design of the machines, standard modules were created to perform these tasks. From these modules the machine could be assembled and then installed in the plant. This brought about a significant savings in cost and also if a particular part was discontinued, it allowed for the dismantling of the machine, to be used at a later date if necessary. Soon it was realized that this concept was well suited for robots and thus modular robots came into being.

Modular robots being reconfigurable, add a great flexibility to the shop, since the robot currently performing a specialized task in a particular environment, could well be used for a different task in a different environment. Very often it has been seen that a robot is over qualified for the job it performs, resulting in the user having to pay a high price for an uncalled for sophistication [M1]. In such situations modular robots are very appropriate because, as the task description changes, the robot can be reprogrammed and reconfigured, thereby using just the necessary capabilities. The following are some of the important benefits of using modular robots.

(1) Greater flexibility as a result of reconfigurability

(2) Greater accuracy as a result of using minimum links

(3) Easier maintenance

(4) Savings in cost

There exists one ambiguity though. Just how fundamental a unit has to be before it can be referred to as a module. Manufacturers differ on this issue. Some refer to their product as modular because it is made up of standard units to form a product family and can be dismantled easily [M3, M4]]. Others refer to their systems as being modular because the system can be reconfigured by adding additional five-axis robot arms to it [M2]. Yet another reason that manufacturers give for calling their product modular is, when a completely matched system of servo motors, amplifiers and controllers is offered as a whole or in parts. We shall concern ourselves with the first

definition, i.e robots assembled from standard units. Although the assembly of these modules is not dictated strictly, there are restrictions to the configurations that can be obtained from these modules. Each component is built to a specification and for each partial configuration there exists only a set of permissible extensions, because of new restrictions. This makes the configuration of such robots an important task.

1.3

MR1

MR1 is a rule-based production system for the use in modular robot configuration. Its central problem-solving method is matching. It exploits its knowledge of the task domain to generate a single acceptable solution. The specific domain of MR1's applicability, for the purposes of this class project, is the configuration of FIBRO modular robot systems. A typical FIBRO system is made up of three basic components- a linear elemnt, rotation element, and a parts gripper. Examples are shown in APPENDIX- . Eight standard linear elements are available ranging in length from 1.25" to 50". Load capacities for short-stroke elements up to 5" lengths range from 44 lbs. to 3600 lbs. The long stroke units can handle loads in excess of 4000 lbs. The rotation elements in standard styles are availble with rotations of 100, 200 and 360 degrees. Loads can be handled up to 3600 lbs. depending on system configuration. Fifty different gripper modules are offered- for external and internal gripping and with

either parallel or angular jaw action. Standard grippers can handle loads from 8 to 60 lbs [M1].

The configuration task for FIBRO robots consists of two parts:

1) Selection of appropriate modules (translatory, rotary and gripper) satisfying the customers specification.

2) Laying out one acceptable sequence of attachment for the selected modules, that can be used for assembly.

Its input is a customer order indicating the types of movements desired and its output is a set of modules with one attachment sequence.

At this point it will be worthwhile to digress and mention a few words justifying the need and existence of expert systems like MR1. The difficulty of configuring any complex system lies in the amount of component information and constraint knowledge that must be available to the configurer during the entire process. For example, there are more than 3000 component pieces, about which a VAX computer configurer must have information [A2]]. Also for any configuration it is difficult to estimate accurately, the amount of constraint knowledge required. Unfortunately enough, much of this knowledge is never available in any document but lies in the head of the human experts, thus making them indispensable. Such knowledge is acquired over a period of years and is normally passed down by the word of mouth. The availibility of an expert system can very well reduce this dependency and make the knowledge accessable to a wider group of people.

An expert system can also be viewed as a form of automation and inherent in it are the related advantages, e.g. high quality performance, speed and efficiency. It is not often that an expert has to seek new knowledge for his daily work and by a little stretch of imagination, his work can also be considered monotonous. As robots relieve blue-collar workers from monotonous and stereo-typed jobs, expert systems can very well be expected to do the same favor to white-collar intellectuals, in the near future.

1.4

APPROPRIATENESS OF MR1

The building of an expert which is to be used commercially for real world problems can take upto several man-years. Obviously such an expert system is "complete", i.e. capable of solving all problems in its domain. It is thoroughly tested and validated and is expected to perform as good as humans, if not better. This is infact true for all the expert systems in use today, e.g. PROSPECTOR, R1, DENDRAL, PUFF, MYCIN, etc [A1].

Let us now address ourselves the question: why is MR1 an appropriate project for this course? Immediately we are faced with a question underlying the previous one: why is a project, to be completed within the time frame of a semester, appropriate for this course? Though there are several expert systems being successfully implemented in the real world today, all human expert in the field of artifical intelligence agree that the area

of expert systems in just breaking out of its embryonic shell. It is nascent, exciting and holds a world of promise, just as any new born. No wonder more and more students in artificial intelligence are jumping into the expert systems bandwagon. A project for this course, apart from providing a "hands-on" experience, achieves a prime objective by making the incumbent realize the multitude of issues involved in the design of an expert system. There probably isn't a better way than "learning by doing".

We have been motivated to select this particular topic because of its closeness to reality. In todays era of automation, robotics is a high growth industry. With the concept of modular robots laid and a host of small-scale industries yearning for automation, we feel it is by no means outrageous to expect the modular robots to command a sizeable share of this growth, in the near future. Also the success of R1 in configuring VAX computers added to our inspiration and we forsee, not very long in the future, an expert system conceptually similar to our MR1, being used in the real world.

2.

SYSTEM OVERVIEW

2.1

Conceptual MR1

If the proposed MR1 were to be used commercially by Fibro Inc., its structure could be as shown in Fig1. It would contain a language processor for domain-specific communications between the user and MR1; knowledge bases comprising of the factual, constraint and procedural knowledge; an inference machine which applies the rules to yield a configuration; a justifier that rationalizes and explains the system behavior. Also included would be two expert sub-systems to make MR1 "complete". The first one would be a salesperson assistant, which would help the customer in his selection of an appropriate robot, for current and future use, depending on his proposed long-term plans. The other expert sub-system would supply the user with all pertinent information regarding installation and use of the robot on shopfloor and if requested, with programs for immediate use. Such a structure, we feel, will make MR1 complete in its domain.

The structure of the current version of MR1 is shown in Fig2. Clearly it is a subset of the structure in Fig1 and was intended to be thus. It captures the essential elements of the proposed system and with simple but conceptually important rules, demonstrates the functioning.

CONCEPTUAL MRI

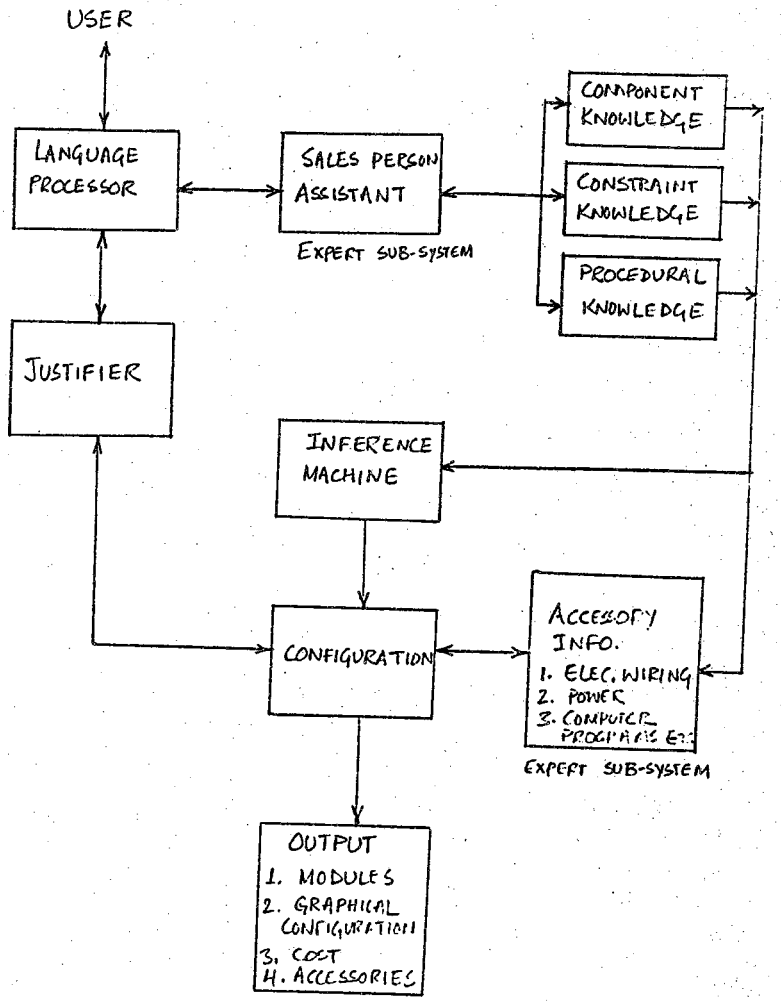


FIG. 1

CURRENT MR1

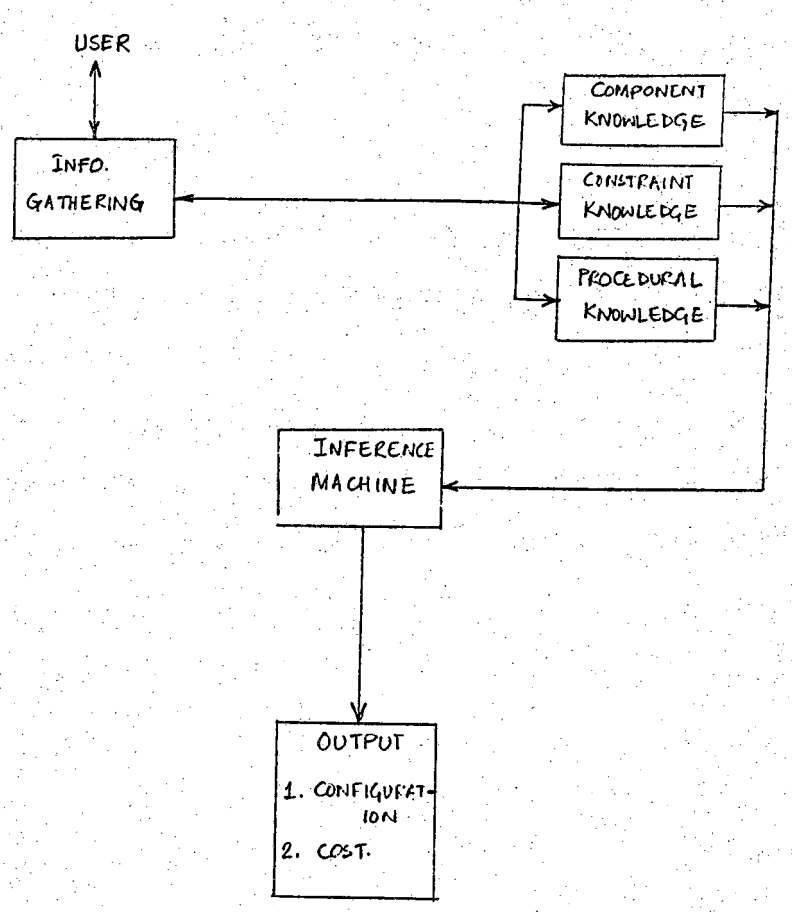


FIG 2.

2.2

How MR1 Works

The working of MR1 is a backward process from the end-effector to link 1. Once the user has answered all the questions posed by MR1, the selection process starts from gripper and terminates after link 1. The gripper selection depends on various criterion, e.g. size, weight and shape of job to be held, type of drive, type of clamping, etc. Once this has been accomplished, it checks whether rotational movement is required. If so, then it selects the appropriate rotary modules based on various other criterion, e.g. deg. of rotation, acceptable load, drive type, mounting type, pitch circle dia. of the attachment, etc. Subsequent selections of translory/rotary modules is carried out in a similar fashion. It follows the principle that the module with maximum load be so chosen, that its rotary or translatory motion is the least.

With the selection of each module the size of the partial configuration increases and additional factors have to be taken into account for subsequent selections. In essence, at each stage, MR1 looks for particular type of extensions, permissible or required, by the current configuration. Thus finally it provides an acceptable configuration or informs the user that his order is inconfigurable with the available Fibro modules.

2.3

Knowledge Base

The knowledge base of MR1 can be broadly classified as follows.

1) Component Knowledge.

This is all the pertinent information about each module available at Fibro Inc. These are represented as facts and given below are some examples.

Class: gripper

Gripper(name, drive-type, open-type, clamp-type, safety, job dia, job weight, pitch circle dia of attachment, module-weight, module cost)

Ex. gripper(g16, hyd, angular, outside, fail-safe, 98, 792, 158, 140, 69100)

Class: rotary

Rotary(name, drive-type, rotation in deg., base loading capacity, side loading capacity, base attachment p.c.d, side attachment p.c.d, module weight, module cost)

Ex. rotary(r5, hyd, 36000, 138100, 376200, 335, 236, 1200, 255500)

Class: linear

Linear(name, drive-type, stroke, endside-loading capacity, broadside-loading capacity, attachment p.c.d, module weight, module cost)

Ex. linear(17, hyd, 394, 885100, 482400, 630, 8040, 212800)

Note: all numerical are multiplied by 100 to eliminate the decimal point -- a PROLOG restriction.

(2) Constraint Knowledge

These are knowledge about the domain which represent strict constraints on the selection of any module. This knowledge is represented as rules. Each such rule embodies the knowledge about physical constraints associated with a module. Given below are some examples.

```
If :  required dia is .lt. or .eq. max. dia handled by module
      required wt. is .lt. or .eq. max. wt. handled by module
      required drive type is module drive type
      required open type is module open type
Then: select Gripper(name, drive-type,.....,module cost)

If :  mounting is base
      required drive type is module drive type
      required rotation is .lt. or .eq. max rotation of module
      effective load of partial config .lt. module base load cap-
      p.c.d of partial config attachment .eq. module base p.c.d
Then: select Rotary(name,drive-type,.....,module cost)
```

(3) Procedural Knowledge

These are knowledge not explicitly represented by the constraint knowledge but can be implicitly derived. Essentially these pertain to the knowledge required for the configuration of the modules. Given below are some examples. The following rules are used to determine whether a rotary module is end-mounted or side-mounted with respect to a linear module.

```
If  : translation is required along X-axis
      translation is required along Y-axis
      translation is required along Z-axis
      max. trans along X is .gt. max trans along Y
      max. trans along Y is .gt. max trans along Z
Then: translation link under consideration is X

If  : translation link under consideration is X
      rotation is required about X
Then: mounting is end

If  : translation link under consideration is X
      rotation is not required about X
Then: mounting is side
```

2.4

Inference Engine

MR1 is a production system and its knowledge is represented in predicate logic. It has been implemented in PROLOG and so all its facts and rules are in appropriate clause form. This was possible because the knowledge was converted to modular form. This conversion was performed with ease for the component knowledge and with considerable effort for the constraint and procedural knowledge, which were mostly obtained from the human experts. Once this task was achieved, the clause form representation was fairly straight forward and also appropriate because of its preciseness and flexibility.

As mentioned earlier, the inference method used by MR1 is matching. Matching can be viewed as a search technique [A3]. It is analogous to Generate-and-Test or Means-End analysis. The advantage of matching is that, if there exists at least one solution state, a path will definitely be found from the initial state to the solution state without any false paths being generated [A2]. For MR1, the initial state is the customer order describing the desired robot motions and job specifications. The intermediate stages in the path to the final stage, are the descriptions of the partial configurations. At each stage of decision, there exists in MR1, constraint and procedural knowledge to indicate what the next step should be. Because the overall task is decomposable in the form of partial

configurations, matching is well suited. Conflict resolution among rules are taken care of by meta-rules.

2.5

Implementation Language

MR1 was implemented in PROLOG. This is a language being widely used by programmers for symbolic computation. The PROLOG approach is to describe known facts and relationships about a problem, rather than to describe the sequence of steps taken by a computer to solve the problem.

PROLOG was chosen as the language to implement MR1 because of its ability to perform well in artificial intelligence applications. the control information primarily backward chaining, is built into the system and only partial explicit control information need be supplied by the user. The use of PROLOG also made it easy to represent the knowledge in clause form. The inbuilt function of backtracking to re-satisfy failed goals also makes programming easy. Further PROLOG enables various structures and trees to be programmed easily, making the knowledge base easier to build and maintain.

3.

KNOWLEDGE ACQUISITION

3.1

Acquisition Process

The concept of modular robots is a new one and accessible experts in this field were very difficult to find. The knowledge for the purposes of this project was acquired from two sources: (1) Journal articles and (2) Human experts. The journal articles equipped us with a good overview of the modular robot concept and the issues involved. But we were unable to obtain any literature about the configuration process. This knowledge was acquired from human experts.

Prof. Robertson of EET at Purdue and Mr Will Dehne, General Manager of Fibro Inc. were the two accessable experts located. Prof. Robertson has had the oppurtunity to build two modular robots himself and they are currently being used in the EET laboratory at Purdue. The Fibro Inc. is located at Rockford, Ill., and we made a trip to their facilities. There we were able to converse with Mr. Dehne and also he supplied us with the necessary documentation about their modules. In the course of our discussion with with Mr. Dehne, he told us that one of the problems in configuring the modular robots is the customer himself. It was not uncommon for the customer to change the order specifications for a variety of reasons (cost, personal

preferences etc.) after the configuration had already been done. Thus a lot of effort was wasted in multiple configurations for a single customer.

It was very clear that configuration is specific to the environment and usage desired by the user. They always have a clear idea of the environmental restrictions the robot is going to be subjected to and from there on it is just a matter of experience-dictated actions. Prof. Robertson was obviously concerned about the available funding and his was more of a long term project to get equipment in his laboratory so that the students could benefit.

Most of the conversation we had with the experts were of the form:

Question: Under what cicumstances would you prefer a hydraulic drive over an electric drive for the end-effector ?

Answer : Oh! That really depends on a whole lot of things. Whether the gripper was connected to a rotary module. If so, then what kinds of job it would be holding; what would the angular velocity be on average; how heavy the job would be. See you got to make sure that the part does not fly off or gets displaced, so the damping is very important in such applications and hydraulic is best for that. Also you got to take into account what is the shop floor environment. So you see it really depends on what and how you intend to use it. Its very problem specific and I also go by my own preferences.

We were well aware that the knowledge extraction would not be the most straightforward task of this project and for once we hit the nail on the head. Formalizing the note-pad scribbles after the meetings itself was a formidable task ("gut feeling"

was a term used quite frequently by Mr. Dehne). A lot of our questions were not answered and during the course of the project, several others arose which we had to take care of. But nevertheless, the knowledge acquisition process was very challenging. It brought to light the structure of our own thought processes which is intricate, to say the least.

4.

RESULTS

4.1

Validation

MR1 is a prototype of a conceptual expert system for the configuration of modular robots. It has achieved success in configuring FIBRO modular robots. Several example customer orders were placed and MR1 gave correct configurations of each. Two of the examples have been included in APPENDIX-II. MR1 was subjected to a two week validation process. It was not difficult to judge the correctness of MR1's configurations, since the FIBRO technical manuals were made available to us. The erroneous configurations were carefully examined and new corrective rules were incorporated. At the end of this validation period, MR1 could correctly configure all previous examples and also additional new examples.

4.2

Performance Evaluation

Now the question is, can MR1 replace a human expert ? An affirmative answer to this question is clearly acceptable since we did not set out to achieve the impossible. MR1 is now in its adolescence and in its domain, is an expert. It does its job accurately and much faster than any human expert. Given an

appropriate environment, MR1 could definitely mature into a successful configurer for the commercial world.

5.

CONCLUSIONS

5.1

In Retrospect

We shall devote this chapter to review the process of building MR1 by trying to answer the following questions:

What has been learnt during this process ?

- There exists non-analytical real world domains with sufficient structure, about which knowledge can be extracted and formalised.
- Knowledge possessed by humans is acquired either by intelligence (i.e. formal learning) or by experience. It is the latter acquisition which can defy sincere attempts of formalization.
- The heart of an expert system is in its knowledge base and every attempt should be made to insure it embodies the problem domain.
- One significant critera for the long-term success of an expert system is, how modular its knowledge representation is. How easy it is to insert new knowledge ? How easy it is to revise old knowledge ?

Is MR1 complete and consistent ?

Where consistency is the property that "every thing produced by the system is true", completeness is the other way round: "every true statement is produced by the system". But clearly we do not mean every true statement in the world- we mean only those which belong to the domain which we are attempting to represent in the system. Therefore completeness can be redifined as: "every true statement which can be expressed in the notation of the system, is a theorem" [A4]. The validation process of MR1 has not been exhaustive due to time limitations, but in the realm of the above definitions, MR1 appears to be complete and consistent.

5.2

Concluding Remarks

Ours is an era of automation brought about by the advent of computers. Today there exists computers beyond the wildest dreams of Charles Babbage, the first human to concieve of the immense computing potential of machinery. By early 1950's mechanized intelligence seemed only a stone's throw away and yet for each barrier crossed, there always cropped up some new barriers to the actual creation of a genuine thinking machine. Even today, intellectuals in the field of artificial intelligence are not unanimous in agreeing to the existence of a sharp borderline between intelligent and non-intelligent behavior [A4]. Nevertheless they realize certain essential abilities for intelligence as:

- to respond to situations very flexibly
- to make sense out of ambiguous or contradictory messages
- to find similarities between situations despite differences which separete them
- to find distinctions between situations despite similarities which link them

Fast though they may be, computers are most inflexible and rule following objects. How can then intelligent behavior be programmed? Research in artificial intelligence concentrates on putting together, sets of formal rules which tell the inflexible machines how to be flexible, in a particular domain. This is no

mean task and is precisely what building an expert system is all about.

6.

REFERENCES

6.1

Modular Robots:

M1 Stauffer, R., "The Fibro/Manca Parts Handling System", Robotics Today, (circa. 1980)

M2 Romeo, G. and Camera, A., "Robots for Flexible Assembly Systems", Robotics Today, (circa. 1979)

M3 Smith, R. C., and Cazes, R., "Modularity In Robotics - Technical Aspects And Applications", Proccedings, Robots In The Automotive Industry An International Conference, April 1982, pp 115-21.

M4 Stanescu, A. M., et al. "Architechture, Control, and Software of a Modular Robot System With Pneumatic Stepping Motors and Vision", Proceedings, The 12th International Symposium on Industrial Robotics, 1982, pp 143-54.

M5 Sellner, J., and Feichtl, R., "Flexible Automation with Modular Industrial Robot System", Proccedings, Autofact Europe, 1983, pp42-60.

6.2

Artificial Intelligence & Expert Systems:

A1 Hayes-Roth, Waterman and Lenat (eds.), "Building Expert Systems", Addison Wesley, Reading, Mass., 1983.

A2 McDermott, J., "R1: A Rule-Based Configurer of Computer Systems", Tech. Report, Dept. of Computer Science, Carnegie-Mellon University.

A3 Newell, A.,--Heuristic programming: ill structured problems; in "Progress in Operations Research", J.S.Aronofsky (ed.) John Wiley and Sons, 1969.

A4 Hofstadter, D. G., "Godel, Escher, Bach: An Eternal Golden Braid", Vintage Books, New York, Sept 1980.

APPENDIX - I

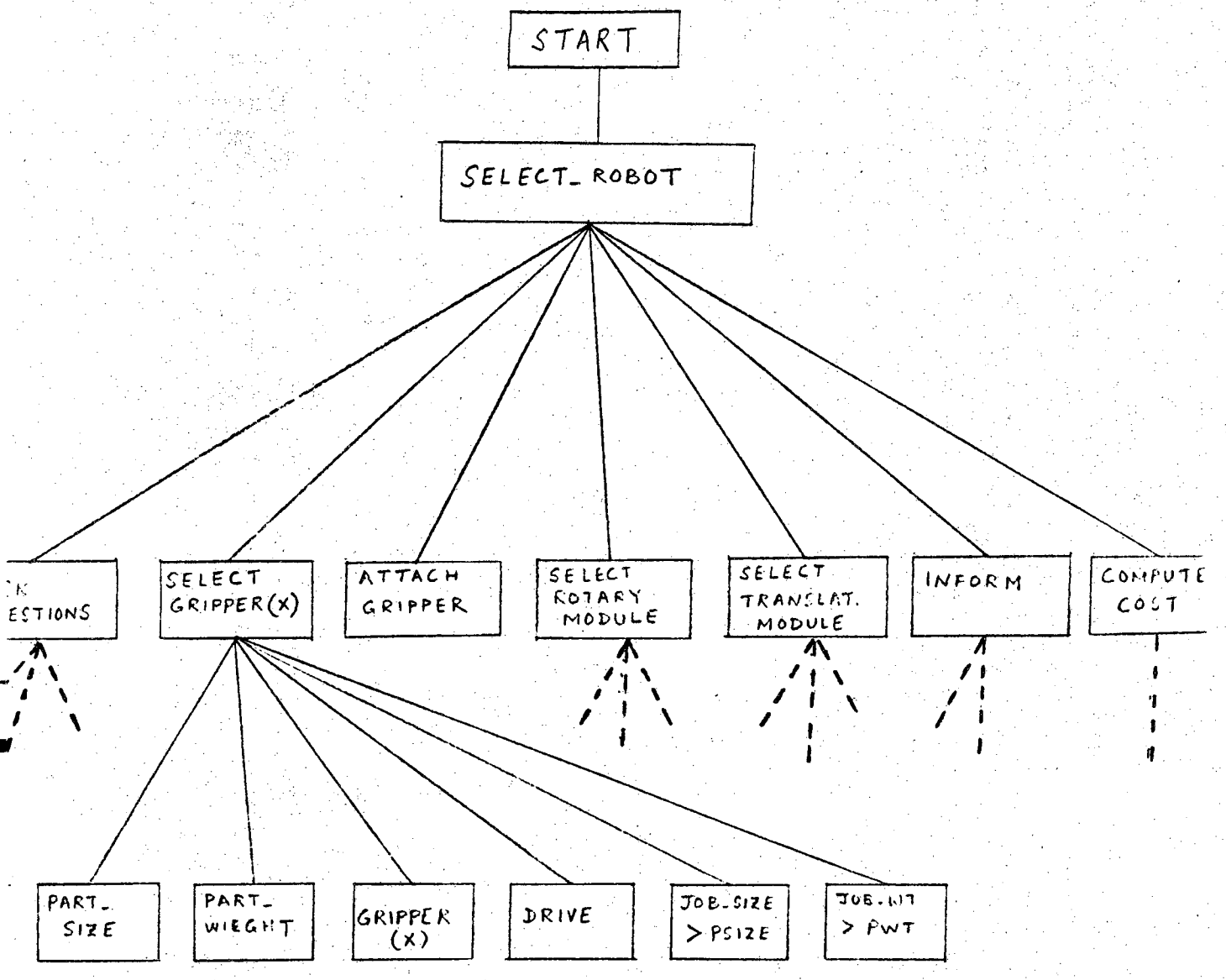
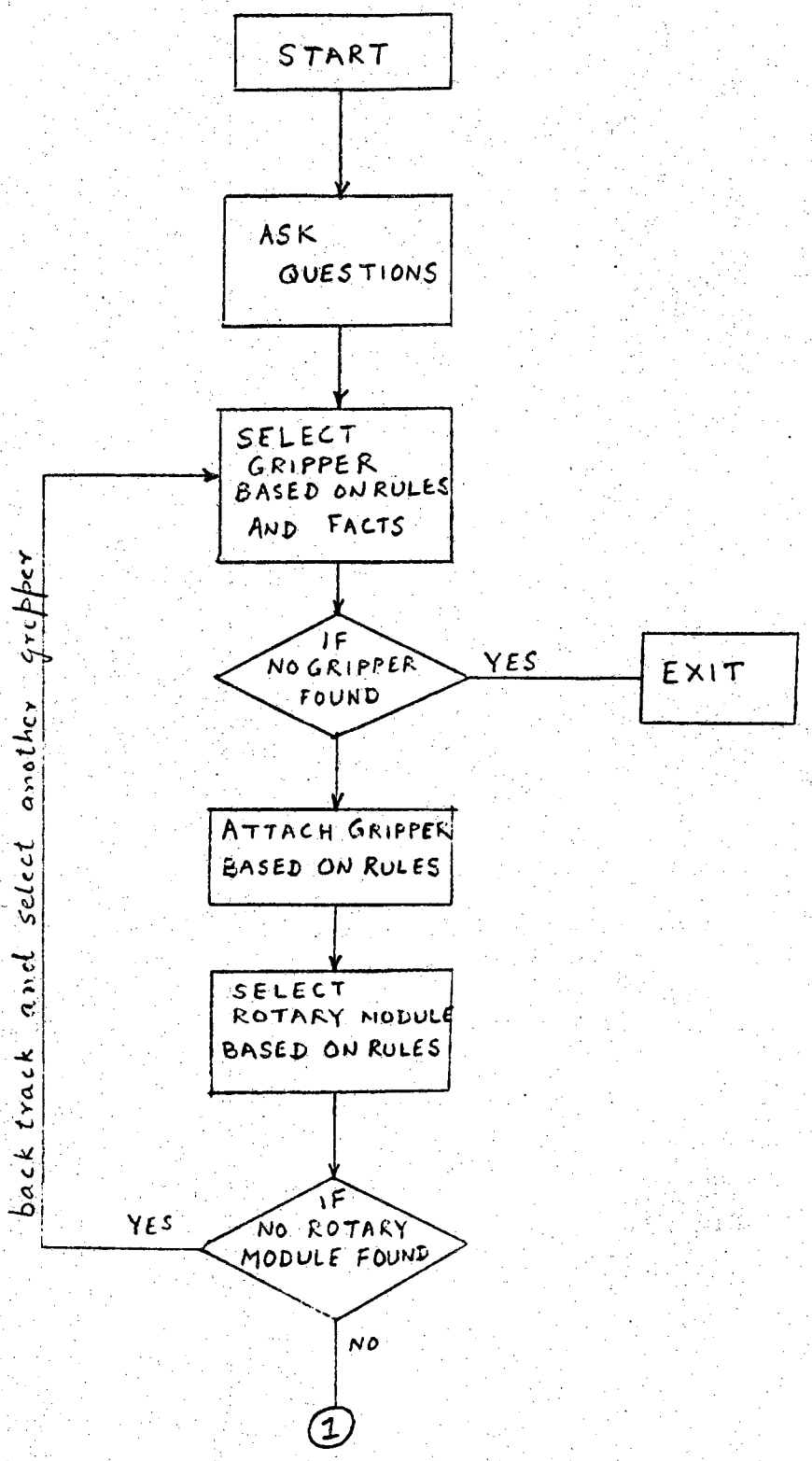
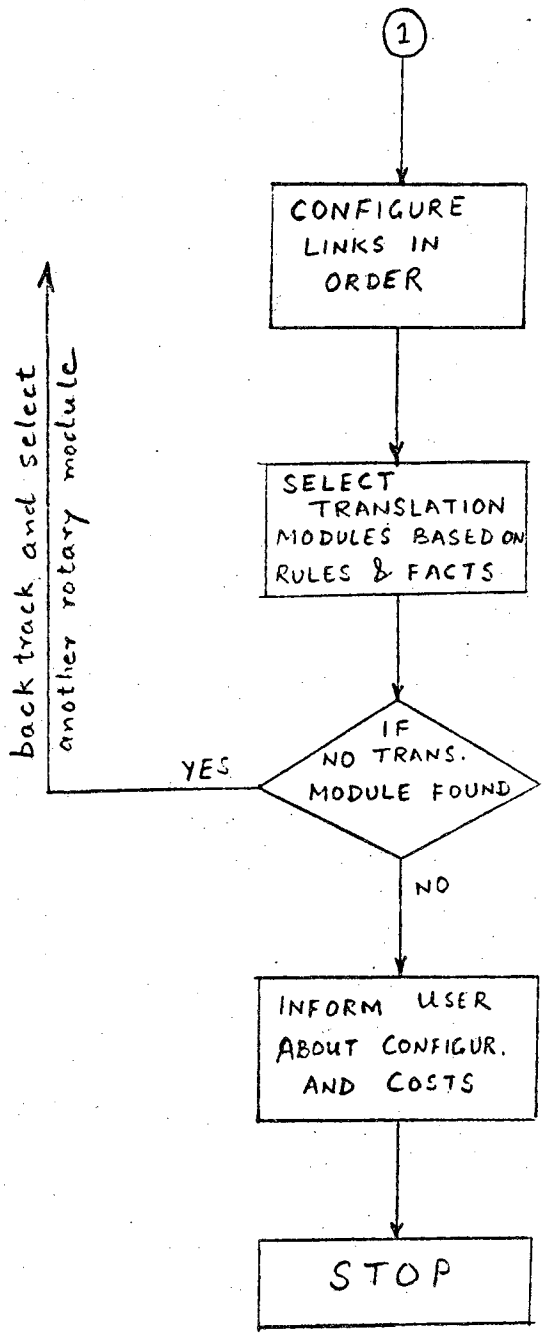


FIG SHOWING PORTIONS OF GOALS & SUBGOALS



PROGRAM FLOW CHART


1
CONFIGURE LINKS IN ORDER
SELECT TRANSLATION MODULES BASED ON RULES & FACTS
IF NO TRANS. MODULE FOUND
YES
back track and select another rotary module
NO
INFORM USER ABOUT CONFIGUR. AND COSTS
STOP

APPENDIX - II

```
-CProlog version 1.3
-| ?- [robot].
 cest consulted 25304 bytes 5.18333 sec.
-db_grippers consulted 3060 bytes 1.11667 sec.
-db_rotary_modules consulted 1344 bytes 0.533336 sec.
-db_linear_modules consulted 540 bytes 0.250003 sec.
-robot consulted 30248 bytes 7.2 sec.

 yes
-| ?- start.


 MR1 :  EXPERT SYSTEM FOR CONFIGURATION OF MODULAR ROBOTS

                    EE 695 TERM PROJECT

Is the part size variable (yes/no):no.

-ax. size of part in inches :2.

 eight of part in lbs :30.

 s the environment clean(yes/no) :no.

Is the environment dusty(yes/no) :yes.

Is the environment humid(yes/no) :yes.

Is compressed air available(yes/no) :no.

-ny preference of drive -hyd,pne,elec,none :none.

-re translation motions required (yes/no):yes.

-ranslation along X-axis(yes/no) :yes.

-ranslation along Y-axis(yes/no) :yes.

-ranslation along Z-axis(yes/no) :yes.

-ax. translation along X-axis in inches :20.

-ax. translation along Y-axis in inches :25.

-ax. translation along Z-axis in inches :15.

 s rotation desired for the gripper(yes/no) :yes.

-ax angle of rotation in deg :200.

-xis of rotation :y.
```

ROBOT CONFIGURATION

```
no. of joints : 4
no. of links  : 5
link5 : gripper -g4
link4 : rotary module -r4
gripper mounted base
link3 : translation module for motion in axis y-14
      load on link3 broadside
link2 : translation module for motion in axis x-12
      load on link2 endside
link1 : translation module for motion in axis z-11
      load on link1 endside
Drive Type :hyd
[total cost of major coponents ,55981
yes
| ?- ^D
[ Prolog execution halted ]
```

```
CProlog version 1.3
| ?- [robot].
test consulted 25304 bytes 5.1 sec.
db_grippers consulted 3060 bytes 1.15 sec.
db_rotary_modules consulted 1344 bytes 0.533334 sec.
db_linear_modules consulted 540 bytes 0.233336 sec.
robot consulted 30248 bytes 7.13333 sec.

yes
| ?- start.


 MR1 :  EXPERT SYSTEM FOR CONFIGURATION OF MODULAR ROBOTS

                        EE 695 TERM PROJECT

Is the part size variable (yes/no):yes.

max. size of part in inches :4.

weight of part in lbs :45.

Is the environment clean(yes/no) :no.

Is the environment dusty(yes/no) :yes.

Is the environment humid(yes/no) :yes.

Is compressed air available(yes/no) :no.

any preference of drive -hyd,pne,elec,none :elec.

The drive choice of elec  wanted by the user is not suited
  for the working environment. A hyd drive is suggested
and the components selected using that drive


Are translation motions required (yes/no):yes.

translation along X-axis(yes/no) :yes.

translation along Y-axis(yes/no) :yes.

translation along Z-axis(yes/no) :no.

Max. translation along X-axis in inches :23.

Max. translation along Y-axis in inches :45.

Is rotation desired for the gripper(yes/no) :yes.

Max angle of rotation in deg :360.

Axis of rotation :z.
```

ROBOT CONFIGURATION

```
no. of joints : 3
no. of links  : 4
link4 : gripper -g10
link3 : rotary module -r9
gripper mounted side
link2 : translation module for motion in axis y-16
      load on link2 endside
link1 : translation module for motion in axis x-12
      load on link1 endside
Drive Type :hyd
[total cost of major coponents ,9130]
yes
| ?- ^D
[ Prolog execution halted ]
```

Chapter 8

RP -- An Expert System for Robot Programming

H. Zhang

RP - An Expert System for Robot Programming

Hong Zhang

1. Introduction

The difficulty one usually faces in programming a robot is the translation from a task specification, such as "grasp bananas that hang on the ceiling of the room", to a robot program that executes the task, since special training and experience are necessary for such a programmer. Efforts have been made to solve this problem, one of which, for instance, is to replace dedicated robot languages by high level programming languages. A Robot Control C Library (RCCL)[1] has been developed at Purdue, which contains a set of C functions suitable for the kinematic controls of a robot manipulator. However, although the functions are coded in C language, experience has shown that it is still quite difficult to program the robot even for someone with extensive C knowledge.

RP is a rule-based expert system[2] which works interactively with the users to create RCCL programs ready to be compiled and executed for a particular set of tasks. The robot is assumed to run in a known workspace with

relevant positions and objects stored in a knowledge database. A robot task that RP is able to generate can be made of one or more of the following: the robot can move to positions either known by or given to the robot; the robot can bring an object from one place to another; and the robot can print information of the world model when requested. The following text is a typical help session of RP system:

```
% RP
RP: what to do ?          (move, bring, print, quit)
User: move
RP: where to go ?         (name of position)
User: conner
RP: please define this location, enter X & Y
User: 123 24
RP: what to do ?          (move, bring, print, quit)
User: bring
RP: which object ?
User: P1
RP: object defined
RP: to where ?
User: a
RP: please specify the location, enter X & Y
User: 10 100
RP: do you want to release it ? (yes / no)
User: yes
RP: what to do ?
User: quit
```

RP: Do you want output program ? (yes / no)

User: yes

RP: what name ?

User: task

RP: Goodbye

%

Once the program is created, it is ready to be compiled and work.

Equipped with a limited command set and coded in LISP[3], RP is simple enough to adapt to with a limited command set while complicated enough to control the robot kinematically. User may know only the objects that exist in the work space and positions of interest while the technical aspect of the programming has been taken care of by RP, which usually requires an expert's knowledge and understanding.

2. Robot Programming

A robot in our context is a programmable manipulator with a pair pneumatic grippers to the end of it. A robot task consists of sequence of moves while taking actions along the way. It may be equipped with different kind of sensors which help to locate the positions of the objects and and guide to robot to accomplish a desired task.

A 4x4 homogeneous transform may be used to describe the

relative position from one object to the other in a three dimensional space. A coordinate frame can be set up for each link of the manipulator by known method and transforms from one link to the next can be calculated. The position of the end effector of the robot relative to its base T_6 can then be solved by multiplying these transforms in proper order. A position the robot desires to move to can be defined by a position equation which equates T_6 to a known transform and the satisfaction of the equation implies the completion of the motion. Of course, more structures can be added to the position equation to reflect the complexity of the task. The robot may be described relative to a world coordinate frame, a tool can be attached to the end of the arm, and such positions on an object may be described further relative to the base of the object.

A robot may move in either one of the two modes, joint mode and Cartesian mode. Joint motion only controls the trajectory end points since each joint works by itself. However, time optimization can be achieved as time is determined by the joint rate. Cartesian motion, on the other hand, is natural to a Cartesian coordinate frame and the trajectory, i.e., the intermediate positions, is well defined. Therefore, for gross motions, joint mode should be selected, and Cartesian mode should be applied in fine motions when the robot is to manipulate objects and precise control is desired. Of course, users are not aware of how motion is actually accomplished in RP and they need only to

specify the end points and the task, and the rest is left to the RP to decide.

Associated with each object are two positions: the approach position and grasp position, all relative to the object's base. The approach position defines where the object should be approached for grasping and grasp position defines where on the object the object should be grasped. When the robot moves to a position in the work space, similar transforms need to be defined for such a position so that a well defined trajectory is obtained at the end of the motion while the gross motion time is is minimized.

The objects in the workspace are restricted to one of the three types: a cube, a ring or a cylinder. The dimensions of the objects, however, can vary. All objects are assumed to be situated on a work bench initially. Users introduce new objects by specifying the x and y coordinates on the bench with default z coordinate and orientation which coincide with the world coordinate frame. Before moving an object to a position, RP will make sure that the position is not occupied by other objects. If it is, warning will be issued to the user and position must be respecified.

3. RCCL Implementation

RCCL contains a set of C functions suitable for robot kinematic control. A RCCL program consists of three parts: variable definition, world model definition and motion

requests.

3.1. Variable Definition

Variables can be of two types: a TRSF_PTR or transform pointer, and a POS_PTR or position pointer. Each time a new position is defined or a new object is introduced, a transform pointer is defined for that transform after the RP makes sure it does not already exist in the world model. Furthermore, since two more transforms are associated with each object or position, their pointers must also be defined. A position is pointed to by a position pointer as its label to specify a desired motion. A new move or bring initiates such a definition.

3.2. World Model

World model refers to the combination of transform definitions and position equation definitions. A transform can be created in RCCL by a call to one of several gentr_'s functions. RP has chosen to use gentr_eul, gentr_pao, and gentr_trsl. The application of one gentr_ depends upon the the particular situation.

A position equation is defined by a call to makeposition() such as

```
pc = makeposition(PC, Z, t6, tool, EQ, c, TL, tool);
```

It is assumed that the robot is defined relative to a base frame by Z transform and the pneumaric grippers are defined

by tool. "PC" is the name of the position, and EQ and TL are known constants to the system to signal the left and right hand sides of the equation and the location of the tool frame.

3.3. Motion Requsts

Since each position equation is labeled, a motion to that position can be requested by a call to move with the label as the parameter. When a user requests to move to a position, the motion is actually accomplished in two steps: a move to the approach position of the position in joint motion with a high velocity, and a move to the final position in Cartesian mode with a low velocity. The parametes are set automatically by the system and users do not need to have the knowledge. This is obtained by calling setmod() and setvel() function primitives. In case of a bring request, a task is programmed in a similar fashion except that users will be asked whether to release the object at the end of the bring, which requires two more RCCL functions CLOSE and OPEN.

3.4. Miscellaneous

RCCL programs must include a rccl.h file which contains the data type definitions. Format will be properly adjusted to make the program readable.

4. Knowledge Base

RP is a ruled-based analysis system. Its knowledge base contains the partial description of the world model the robot is working in. The model can be easily expanded by users and by the system itself. Initially, the knowledge contains position of the robot, some positions in the work space, models of the objects, and instances of these models with their current positions in the work space. The knowledge base is expandable as pointed out early as the user and RP interaction proceeds by including more objects. Even though users only specify the type of the object and the dimensions, the grasp and approach positions are computed automatically for successful assembly once the new objects are introduced.

5. Inference System

The development of RP is based on the following rules to set up the logic of the system.

Rule 1. If begin
there is a move
then start function move

Rule 2. If begin
there is a bring
then start function bring

Rule 3. If begin
there is a print
then start function print

Rule 4. If begin
there is a quit
then start function quit

Rule 5. If move
the robot holds an object

```
        then release the object
             remain in move
Rule 6. If move
           position exists
        then go to the position
             remain in move

Rule 7. If move
           position exists
           position occupied by an object
        then approach the object
             remain in move

Rule 8. If move
           position does not exist
        then define the position
             remain in move

Rule 9. If move
           position has approach position
        then go to the position thru approach position
             remain in move

Rule 10. If move
           the position is known
        then declare the transform pointers
             remain in move

Rule 11. If move
           the position is known
           transform pointers already declared
        then define transforms
             remain in move

Rule 12. If move
           the position is known
           transform pointers already declared
           transforms defined
        then define transforms
             remain in move

Rule 13. If move
           the position is known
           transform pointers already declared
        then define transforms
             change to begin step

Rule 14. If bring
           the arm is holding an object
        then ask for what to bring
             remain in bring

Rule 15. If bring
           the arm is holding an object
```

```
          held object same as the object to bring
       then ask for destination
            remain in bring

Rule 16. If bring
          the arm is holding an object
          held object different from object to bring
       then release the object and update its position
            remain in bring

Rule 17. If bring
          the arm has the actual object to bring
       then ask for what to bring
            remain in bring

Rule 18. If bring
          destination is known
       then declare the transform pointers
            remain in bring

Rule 19. If bring
          transform pointer have been declared
       then define the transforms
            remain in bring

Rule 20. If bring
          transforms have been declared
       then define position equation
            remain in bring

Rule 21. If bring
          position equation has been defined
       then move to the approach position of the destination
            remain in bring

Rule 22. If bring
          the object is at approach position
       then set default approach velocity
            change mode to Cartesian
            move to the destination
            remain in bring

Rule 23. If bring
          the arm is at the destination
       then ask if release
            go to release step

Rule 24. If release
          if want to release the object
       then release the object
          update the object transform
          back up the approach position of the object
          set default joint motion velocity
          change mode to joint
```

```
            go to begin step

Rule 25. If bring
            if arm is holding an object
            the position has been defined
            the position already occupied
         then ask for a new position

Rule 26. If bring
            the arm is not holding any object
            the object is not in the database
         then ask for the object

Rule 27. If bring
            the arm is not holding any object
            the object is known
         then go to grasp step

Rule 28. If grasp
            the object is known
            the transform pointers undefined
         then define transform pointers
              remain in grasp

Rule 29. If grasp
            the transform pointers defined
         then define transforms
              remain in grasp

Rule 30. If grasp
            transforms defined
         then define approach position equition

Rule 31. If grasp
            approach position defined
         then move to the approach position

Rule 32. If grasp
            at approach position
         then set Cartesian motion mode
              set default Cartesian motion velocity
              move to the grasping position
              close fingers
              go to bring

Rule 33. If bring
            arm is holding an object
            the object is in its original position
         then move to the object approach position
              set velocity to default joint velocity
              set motion mode to joint mode
              remain in bring

Rule 34. If print
```

```
          want to print object and object unknown
       then ask again
            remain in print

Rule 35. If print
          want to print object and object known
       then print its origin
            go to begin step

Rule 37. If print
          want to print a position
       then print names of transforms in the position
            go to begin

Rule 38. If quit
       then ask if output wanted
            go to output

Rule 39. If output
          yes
       then ask for name
            go to cat
Rule 40. If cat
       then concatenate all the files
            exit

Rule 41. If output
          no
       then eixt
```

6. Summary

A robot programming expert system is discussed and implemented in this project and its simulation conducted. Tests have shown the applicability of such a system to the limited kinematic control of a robot manipulator within a simplied work space. The easy interaction with the system makes it possible for a user to learn to program the robot within a very short time.

Only a small number of functions in RCCL have been included into the RP system due to limited time. However, the system is easily expandable to cover a larger portion of

RCCL functions. An improvement that should be made is to include the dynamics control or force control into the system to make RP a reality rather than an experiment as in tasks such as assembly when the robot is in contact with the environment and constrained by the task geometry, control in the constrained directions requires compliance, the compliant specification needs to be carefully determined, and assembly strategies devised. However, since such a programming system does not exist today, RP does not consider the complications involved in force control.

Appendix1: Examples

In example one, we demonstrate how moves are programmed in RP and how the user RP interaction takes place.

```
RP: what to do ?          (move, bring, print, quit)
User: print
RP: what ?        ('name' 'objects' 'positions' 'quit')
User: positions
RP:       (Z tool)
RP: what to do ?          (move, bring, print, quit)
User: print
RP: what ?        ('name' 'objects' 'positions' 'quit')
User: objects
RP:       (P1 R1 C1)
RP: what to do ?          (move, bring, print, quit)
User: move
RP: where to go ?         (name of position)
User: a
RP: please define this location, enter X & Y
User: 10 100
RP: what to do ?          (move, bring, print, quit)
User: move
RP: where to go ?         (name of position)
User: b
RP: please define this location, enter X & Y
User: 200 300
RP: what to do ?          (move, bring, print, quit)
User: move
RP: where to go ?         (name of position)
User: c
RP: please define this location, enter X & Y
User: 300 100
RP: what to do ?          (move, bring, print, quit)
User: move
RP: where to go ?         (name of position)
User: a
RP: what to do ?          (move, bring, print, quit)
User: print
RP: what ?        ('name' 'objects' 'positions' 'quit')
User: positions
RP:       (c b a Z tool)
RP: what to do ?          (move, bring, print, quit)
User: print
RP: what ?        ('name' 'objects' 'positions' 'quit')
User: objects
RP:       (P1 R1 C1)
RP: what to do ?          (move, bring, print, quit)
User: quit
RP: Do you want output program ?          (yes/no)
User: yes
RP: what name ?
User: task
RP: Goodbye
```

Output file "task.c":.

```
#include "rccl.h"

pumatask()
{
        TRSF_PTR tool;
        TRSF_PTR Z;
        TRSF_PTR a_a;
        POS_PTR pa_a;
        TRSF_PTR a;
        POS_PTR pa;
        TRSF_PTR b_a;
        POS_PTR pb_a;
        TRSF_PTR b;
        POS_PTR pb;
        TRSF_PTR c_a;
        POS_PTR pc_a;
        TRSF_PTR c;
        POS_PTR pc;

        tool = gentr_trsl("TOOL", 0, 0, 170);
        Z = gentr_trsl("Z", 0, 0, 330);
        a_a = gentr_trsl("A_A", 0, 0, 100);
        pa_a = makeposition("PA_A", Z, t6, tool, EQ, a, a_a, TL, tool);
        a = gentr_trsl("A", 10, 100, 0);
        pa = makeposition("PA", Z, t6, tool, EQ, a, TL, tool);
        b_a = gentr_trsl("B_A", 0, 0, 100);
        pb_a = makeposition("PB_A", Z, t6, tool, EQ, b, b_a, TL, tool);
        b = gentr_trsl("B", 200, 300, 0);
        pb = makeposition("PB", Z, t6, tool, EQ, b, TL, tool);
        c_a = gentr_trsl("C_A", 0, 0, 100);
        pc_a = makeposition("PC_A", Z, t6, tool, EQ, c, c_a, TL, tool);
        c = gentr_trsl("C", 300, 100, 0);
        pc = makeposition("PC", Z, t6, tool, EQ, c, TL, tool);

        setmod('j');
        setvel(300, 50);
        move(pa_a);
        setmod('c');
        setvel(10, 30);
        move(pa);
        setmod('j');
        setvel(300, 50);
        move(pb_a);
        setmod('c');
        setvel(10, 30);
        move(pb);
        setmod('j');
        setvel(300, 50);
        move(pc_a);
        setmod('c');
        setvel(10, 30);
        move(pc);
```

```
    setmod('j');
    setvel(300, 50);
    pa_a = makeposition("PA_A", Z, t6, tool, EQ, a, a_a, TL, tool);
    move(pa_a);
    setmod('c');
    setvel(10, 30);
    pa = makeposition("PA", Z, t6, tool, EQ, a, TL, tool);
    move(pa);
    setmod('j');
    setvel(300, 50);
    release();
}
```

In example two, we demonstrate how new objects can be introduced to the system and how the bring actions take place. We can also see how data in knowledge base is updated and obtained by the user.

```
RP: what to do ?          (move, bring, print, quit)
User: print
RP: what ?       ('name' 'objects' 'positions' 'quit')
User: objects
RP:      (P1 R1 C1)
RP: what to do ?          (move, bring, print, quit)
User: bring
RP: which object ?
User: P1
RP: object defined
RP: to where ?
User: a
RP: please define this location, enter X & Y
User: 10 100
RP: do you want to release it ? (yes / no)
User: yes
RP: what to do ?          (move, bring, print, quit)
User: bring
RP: which object ?
User: R1
RP: object defined
RP: to where ?
User: a
RP: the position choosen is already occupied.
RP: to where ?
User: b
RP: please define this location, enter X & Y
User: 20 200
RP: do you want to release it ? (yes / no)
User: yes
RP: what to do ?          (move, bring, print, quit)
User: bring
RP: which object ?
User: P1
RP: object defined
RP: to where ?
User: c
RP: please define this location, enter X & Y
User: 100 300
RP: do you want to release it ? (yes / no)
User: yes
RP: what to do ?          (move, bring, print, quit)
User: bring
RP: which object ?
User: C2
RP: object C2 must be defined
RP: enter data or quit any time
RP: object type ?         (cylinder, ring, cube)
User: cube
```

```
RP: enter X, Y & Z please (no commas)
User: 50 50 50
RP: enter initial location (X & Y)
User: 300 200
RP: object defined
RP: to where ?
User: a
RP: do you want to release it ? (yes / no)
User: yes
RP: what to do ?          (move, bring, print, quit)
User: print
RP: what ?       ('name' 'objects' 'positions' 'quit')
User: objects
RP:       (C2 c b a P1 R1 C1)
RP: what to do ?          (move, bring, print, quit)
User: print
RP: what ?       ('name' 'objects' 'positions' 'quit')
User: positions
RP:       (Z tool)
RP: what to do ?          (move, bring, print, quit)
User: bring
RP: which object ?
User: C2
RP: object defined
RP: to where ?
User: d
RP: please define this location, enter X & Y
User: 150 10
RP: do you want to release it ? (yes / no)
User: yes
RP: what to do ?          (move, bring, print, quit)
User: print
RP: what ?       ('name' 'objects' 'positions' 'quit')
User: positions
RP:       (a Z tool)
RP: what to do ?          (move, bring, print, quit)
User: print
RP: what ?       ('name' 'objects' 'positions' 'quit')
User: objects
RP:       (d C2 c b P1 R1 C1)
RP: what to do ?          (move, bring, print, quit)
User: quit
RP: Do you want output program ?          (yes/no)
User: yes
RP: what name ?
User: task
RP: Goodbye
```

Source code "task.c"

```
#include "rccl.h"

pumatask()
{
        TRSF_PTR tool;
        TRSF_PTR Z;
        TRSF_PTR P1_a;
        POS_PTR pP1_a;
        TRSF_PTR P1_g;
        POS_PTR pP1_g;
        TRSF_PTR a;
        POS_PTR pa;
        TRSF_PTR R1_a;
        POS_PTR pR1_a;
        TRSF_PTR R1_g;
        POS_PTR pR1_g;
        TRSF_PTR b;
        POS_PTR pb;
        TRSF_PTR c;
        POS_PTR pc;
        TRSF_PTR C2_a;
        POS_PTR pC2_a;
        TRSF_PTR C2_g;
        POS_PTR pC2_g;
        TRSF_PTR a_a;
        POS_PTR pa_a;
        TRSF_PTR d;
        POS_PTR pd;

        tool = gentr_trsl("TOOL", 0, 0, 170);
        Z = gentr_trsl("Z", 0, 0, 330);
        a = gentr_trsl("A", 10, 100, 0);
        pa = makeposition("PA", Z, t6, tool, EQ, a, TL, tool);
        b = gentr_trsl("B", 20, 200, 0);
        pb = makeposition("PB", Z, t6, tool, EQ, b, TL, tool);
        c = gentr_trsl("C", 100, 300, 0);
        pc = makeposition("PC", Z, t6, tool, EQ, c, TL, tool);
        a_a = gentr_trsl("A_A", 0, 0, 100);
        pa_a = makeposition("PA_A", Z, t6, tool, EQ, a, a_a, TL, tool);
        d = gentr_trsl("D", 150, 10, 0);
        pd = makeposition("PD", Z, t6, tool, EQ, d, TL, tool);

        setmod('j');
        setvel(300, 50);
        P1_a = gentr_eul("P1_A", 300, 25, 25, 0, 30, 0);
        pP1_a = makeposition("PP1_A", Z, t6, tool, EQ, P1, P1_a, TL, tool);
        move(pP1_a);
        setmod('c');
        setvel(10, 30);
        P1_g = gentr_eul("P1_G", 25, 25, 25, 0, 30, 0);
        pP1_g = makeposition("PP1_G", Z, t6, tool, EQ, P1, P1_g, TL, tool);
        move(pP1_g);
```

```
CLOSE;
move(pa);
OPEN;
P1= assigntr(P1, a);
pP1_a = makeposition("PP1_A",Z,t6,tool,EQ,P1,P1_a,TL,tool);
move(pP1_a);
setmod('j');
setvel(300, 50);
R1_a = gentr_eul("R1_A",300,0,12.5,0,90,180);
pR1_a = makeposition("PR1_A",Z,t6,tool,EQ,R1,R1_a,TL,tool);
move(pR1_a);
setmod('c');
setvel(10, 30);
R1_g = gentr_eul("R1_G",0,0,12.5,0,90,180);
pR1_g = makeposition("PR1_G",Z,t6,tool,EQ,R1,R1_g,TL,tool);
move(pR1_g);
CLOSE;
move(pb);
OPEN;
R1= assigntr(R1, b);
pR1_a = makeposition("PR1_A",Z,t6,tool,EQ,R1,R1_a,TL,tool);
move(pR1_a);
setmod('j');
setvel(300, 50);
pP1_a = makeposition("PP1_A",Z,t6,tool,EQ,P1,P1_a,TL,tool);
move(pP1_a);
setmod('c');
setvel(10, 30);
pP1_g = makeposition("PP1_G",Z,t6,tool,EQ,P1,P1_g,TL,tool);
move(pP1_g);
CLOSE;
move(pc);
OPEN;
P1= assigntr(P1, c);
pP1_a = makeposition("PP1_A",Z,t6,tool,EQ,P1,P1_a,TL,tool);
move(pP1_a);
setmod('j');
setvel(300, 50);
C2_a = gentr_eul("C2_A",300,25,300,0,90,180);
pC2_a = makeposition("PC2_A",Z,t6,tool,EQ,C2,C2_a,TL,tool);
move(pC2_a);
setmod('c');
setvel(10, 30);
C2_g = gentr_eul("C2_G",(quotient (nth 0 geometry) 2),25,40,0,90,18
pC2_g = makeposition("PC2_G",Z,t6,tool,EQ,C2,C2_g,TL,tool);
move(pC2_g);
CLOSE;
move(pa_a);
setmod('c');
setvel(10, 30);
pa = makeposition("PA",Z,t6,tool,EQ,a,TL,tool);
move(pa);
setmod('j');
setvel(300, 50);
```

```
OPEN;
C2= assigntr(C2, a);
pC2_a = makeposition("PC2_A", Z, t6, tool, EQ, C2, C2_a, TL, tool);
move(pC2_a);
setmod('j');
setvel(300, 50);
pC2_a = makeposition("PC2_A", Z, t6, tool, EQ, C2, C2_a, TL, tool);
move(pC2_a);
setmod('c');
setvel(10, 30);
pC2_g = makeposition("PC2_G", Z, t6, tool, EQ, C2, C2_g, TL, tool);
move(pC2_g);
CLOSE;
move(pd);
OPEN;
C2= assigntr(C2, d);
pC2_a = makeposition("PC2_A", Z, t6, tool, EQ, C2, C2_a, TL, tool);
move(pC2_a);
setmod('j');
setvel(300, 50);
release();
}
```

Appendix2: Lisp Source Code

```
; Define the data base by property lists
; for the 695c project purpose
; 695c/1database.l

; Global variables needed

(setq *names_list* nil)

; We need a particular function to manage the inputs in the property
; list.
; l_partprop = (t_name
;                g_model          { type of object                          }
;                l_geometry       { geometric form x , y & z dimensions     }
;                l_location       { where the object-origin is located      }
;                l_approach       { where the cartesian movement starts     }
;                l_grasploc)      { where grasp the object from the obj.    }
;                                 { origin                                  }

(defun add_name (l_partprop)
  (loop for item from 0 below (length (cdr l_partprop))
        do
        (putprop (car l_partprop)
                 (nth item (cdr l_partprop))
                 (nth item '(model
                             geometry
                             location
                             approach
                             grasploc)))
        finally (setq *names_list* (cons (car l_partprop) *names_list*)))
  (car l_partprop))

; definition of tool and Z

(add_name '(tool
            ()
            ()
            (trsl 0 0 170 )
            ))

(add_name '(Z
            ()
            ()
            (trsl 0 0 330 )
            ))

; definition of the objects

(add_name '(C1
            cylinder
            (0 12.5 127)
            (pao (1 0 0 0) (0 1 0 0) (0 0 1 0) (300 0 100 1))
            (eul 300 0 100 0 90 180)
```

```
            (eul 0 0 100 0 90 180)
            ))

(add_name '(R1
            ring
            (26 51 25)
            (pao (1 0 0 0) (0 1 0 0) (0 0 1 0) (300 300 0 1))
            (eul 300 0 12.5 0 90 180)
            (eul 0 0 12.5 0 90 180)
            ))
(add_name '(P1
            cube
            (50 50 50)
            (pao (1 0 0 0) (0 1 0 0) (0 0 1 0) (500 0 0 1))
            (eul 300 25 25 0 30 0)
            (eul 25 25 25 0 30 0)
            ))
```

"2main.l"

```
; this file contains the structure of the interactive main to control
; the whole expert.
; 695c/2main.l

; function "RP", RP stands for expert robot programer.

(defun RP ()
  (setq *robot_set* (other_trsf))
  (setq *hand_full* ())
  (setq *declarations* ())
  (setq *joint_vel* '(300 50))
  (setq *approach_vel* '(10 30))
  (setq out_def (outfile 'inc.h))        ; declarations to be included
  (setq out_stm (outfile 'stm.c))        ; statement for the world
  (setq out_act (outfile 'act.c))        ; actions to take
  (princ (concat "#include " '|"| "rccl.h" '|"| (ascii 13) (ascii 10)
                 "pumatask()" (ascii 13) (ascii 10) "{"
                 (ascii 13) (ascii 10)) out_def)
  (princ (concat (ascii 13) (ascii 10)) out_stm)
  (princ (concat (ascii 13) (ascii 10)) out_act)
  (declare_trsf_ptr 'tool)
  (declare_trsf_ptr 'Z)
  (gentr 'tool (get 'tool 'location) out_stm)
  (gentr 'Z (get 'Z 'location) out_stm)
  (set 'mod 'j)
  (set 'vel *joint_vel*)
  (prog ()
        lb1
        (princ "RP: what to do ?        (move, bring, print, quit)
User: ")
        (setq *current_action* (read))
        (cond ((equal *current_action* 'move)
               (moverp) (go lb1))
              ((equal *current_action* 'bring)
               (bringrp) (go lb1))
              ((equal *current_action* 'print)
               (printrp) (go lb1))
              ((equal *current_action* 'quit)
               (princ (concat " release();"
                              (ascii 13) (ascii 10)
                              "}" (ascii 13) (ascii 10)) out_act)
               (close out_act)
               (close out_stm)
               (close out_def)
               (prog ()
                     lb2
                     (princ (concat
                              "RP: Do you want output program ?  (yes/no)"
                              (ascii 13) (ascii 10) "User: "))
                     (cond ((equal (setq t_arg (read)) 'yes)
                            (princ (concat "RP: what name ?"
                                           (ascii 13) (ascii 10) "User: ")
```

```
                        (setq t_aux (concat (read) ".c"))
                        (*process (concat "rm " t_aux))
                        (*process (concat "cat inc.h stm.c act.c >" t_au
                       ((equal t_arg 'no) ())
                       (t (princ (concat "RP: misspelling in your answer
                                         (ascii 13) (ascii 10)) (go lb2)
            )

           (t (princ "RP: please be nice and answer what you are suposed
              (terpri)
              (go lb1)))))

; function "other_trsf", set the global variable *robot_set* with the
; default components of the transform equation for the actual robot
; settings. Information is retrieved from the data base.

(defun other_trsf ()
  (cond ((and (member 'tool *names_list*) (member 'Z *names_list*))
         (list 'Z 't6 'tool 'EQ 'newpos 'TL 'tool))
        (t (princ "RP: transform -tool- or -Z- are missing in the
    database. Please define them.") (break))))
```

"3move.l"

```
; file 695c/3move.l
; contains functions related with moving the arm alone

; function "moverp", primitive, set one of the path on the e.s. graph
; suposse to return nil.

(defun moverp ()
  (prog ()
        (if *hand_full*
            (prog ()
                  (princ "RP: leaves what is holding in actual position
")
                  (open_fingers *hand_full*) ;Includes the disapproach move
                  (setq *hand_full* nil)))
        lb2
        (princ "RP: where to go ?       (name of position)
User: ")
        (setq *next_pos* (read))
        (cond ((member *next_pos* *names_list*) (go_onto *next_pos*))
              (t (new_position *next_pos*) (go_onto *next_pos*)))
        (return ())))

; function "achieve", now checks for the properties and situation
; of the position, and eventualities with the possible existence of
; an object in that location.

(defun achieve (t_arg)
  (prog ()
        (cond ((get t_arg 'model)
               (princ "RP: the position choosen is already occupied.
") (return 't))
              (t (codify_pos t_arg (get t_arg 'location))
                 (return ())))))

; function "go_onto", write the rccl instructions to move
; the arm onto an object

(defun go_onto (t_arg)
  (codify_pos t_arg (get t_arg 'location))
  )

; function "go_then", move to a free position no approach necessary

(defun go_then (t_arg)
  (cond ((member t_arg *declarations*) (move (concat 'p t_arg)))
        (t (declare_trsf_ptr t_arg)
           (gentr t_arg (get t_arg 'location) out_stm)
           (setq p_next (makeposition (list t_arg) *robot_set* out_stm))
           (move p_next))))

; function "new_position", when a location has been not defined then
; the system will prompt for it.
```

```
(defun new_position (t_arg)
  (prog (inp xval yval)
        (princ (concat "RP: please define this location, enter X & Y"
                       (ascii 13) (ascii 10) "User: "))
        (setq inp (list (read) (read)))
        (setq xval (car inp))
        (setq yval (cadr inp))
        (add_name (list t_arg
                        ()
                        ()
                        '(trsl ,xval ,yval 0 )
                        '(trsl 0 0 100 )
                        ()))
        ))

; function "codify_pos", does the preprogram for the rccl code.
; sets in order into a file the transforms to be defined.

(defun codify_pos (t_arg l_trans)
  (prog (app_pos p_a p_next)
        (setq app_pos (concat t_arg '_a))       ; approach pos related
        (cond ((member app_pos *declarations*)
               (setq p_a (makeposition (list t_arg app_pos)
                                       *robot_set* out_act))
               (move p_a))
              (t (declare_trsf_ptr app_pos)
                 (gentr app_pos (get t_arg 'approach) out_stm)
                 (setq p_a (makeposition (list t_arg app_pos)
                                         *robot_set* out_stm))
                 (move p_a)))
        (set 'mod 'c)
        (set 'vel *approach_vel*)
        (cond ((member t_arg *declarations*)
               (setq p_next (makeposition (list t_arg) *robot_set* out_act))
               (move p_next))
              (t (declare_trsf_ptr t_arg)
                 (gentr t_arg (get t_arg 'location) out_stm)
                 (setq p_next (makeposition (list t_arg) *robot_set* out_stm
                 (move p_next)))
        (set 'mod 'j)
        (set 'vel *joint_vel*)))

; function "declare_trsf_ptr", prints in a file the declaration
; for a transform pointer.

(defun declare_trsf_ptr (t_arg)
        (princ "        TRSF_PTR " out_def)
        (princ t_arg out_def)
        (princ ";
" out_def)
        (setq *declarations* (cons t_arg *declarations*)))

; function "declare_pos_ptr", prints in a file the declaration
; for a position pointer.
```

```
(defun declare_pos_ptr (t_arg)
        (princ "        POS_PTR " out_def)
        (princ t_arg out_def)
        (princ ";
" out_def)
        (setq *declarations* (cons t_arg *declarations*)))

; function "gentr", find the transform definition of a position or whateve
; coming from the database

(defun gentr (t_arg1 l_aux t_port)
  (prog ()
        (cond ((or (equal (car l_aux) 'eul)              ; set for euler an
                   (equal (car l_aux) 'rot)              ; set for axis rot
                   (equal (car l_aux) 'rpy)              ; set for rpy angl
                   (equal (car l_aux) 'trsl))            ; set for translat
               (princ " " t_port)
               (princ t_arg1 t_port)
               (princ " = gentr_" t_port)
               (princ (car l_aux) t_port)
               (princ "(" t_port)
               (princ (capitals t_arg1) t_port)
               (loop for item in (cdr l_aux)
                     do (princ "," t_port) (princ item t_port))
               (princ ");
" t_port))
              ((equal (car l_aux) 'pao)
               (princ " " t_port)
               (princ t_arg1 t_port)
               (princ " = gentr_" t_port)
               (princ (car l_aux) t_port)
               (princ "(" t_port)
               (princ (capitals t_arg1) t_port) (princ "," t_port)
               (princ (caadddr l_aux) t_port) (princ "," t_port)
               (princ (cadadddr l_aux) t_port) (princ "," t_port)
               (princ (caddadddr l_aux) t_port) (princ "," t_port)
               (princ (caaddr l_aux) t_port) (princ "," t_port)
               (princ (cadaddr l_aux) t_port) (princ "," t_port)
               (princ (caddaddr l_aux) t_port) (princ "," t_port)
               (princ (caadr l_aux) t_port) (princ "," t_port)
               (princ (cadadr l_aux) t_port) (princ "," t_port)
               (princ (caddadr l_aux) t_port) (princ ");
" t_port))
              )))


; function "makeposition", write the statement for rccl makeposition

(defun makeposition (l_arg1 l_arg2 t_port)
  (prog (p_t_arg)
        (setq p_t_arg (concat 'p (car (last l_arg1))))
        (cond ((member p_t_arg *declarations*) ())
              (t (declare_pos_ptr p_t_arg)))
        (princ "        " t_port)
```

```
        (princ p_t_arg t_port)
        (princ " = makeposition(" t_port)
        (princ (capitals p_t_arg) t_port)
        (loop for item in l_arg2
              when (equal item 'newpos)
              do (loop for item1 in l_arg1
                       do (princ "," t_port) (princ item1 t_port))
              else
              do (princ "," t_port) (princ item t_port))
        (princ ");
" t_port)
        (return p_t_arg)))

; function "capitals", return the argument capitalized

(defun capitals (t_arg)
        (loop for item in (exploden t_arg)
              when (and (> item (car (exploden '|`|)))
                        (> (car (exploden '{)) item))
              collect (diff item 32) into auxlist
              else collect item into auxlist
              finally (return (concat '|"| (maknam auxlist) '|"|)))))

; Function "move", write on the action file the adequate sentence

(defun move (t_arg)
  (princ "      move(" out_act)
  (princ t_arg out_act)
  (princ ");
  " out_act))
```

347

"4bring.l"

```
; file 695c/4bring.l
; Lisp functions related to program the manipulator to take something
; from one place to another.

; Function "bringrp", first level of call.

(defun bringrp ()
  (prog (t_aux)
        lb1
        (cond (*hand_full* (princ (concat "RP: I am holding  "
                                          *hand_full*
                                          ", which object ?" (ascii 13)
                                          (ascii 10) "User: "))
                           (cond ((equal (setq t_aux (read)) *hand_full*)
                                  (go lb2))
                                 (t (open_fingers *hand_full*)
                                    (setq *hand_full* t_aux))))
              (t (princ (concat "RP: which object ?" (implode '(13 10))
                                "User: "))
                 (setq *hand_full* (read))))
        (cond ((member *hand_full* *names_list*) ())
              (t (prog (model geometry location)
                 lb11
                 (princ (concat "RP: object " *hand_full* " must be definec
                                (implode '(13 10))
                                "RP: enter data or quit any time "
                                (implode '(13 10))
                                "RP: object type ?      (cylinder, ring, c
                                (implode '(13 10))
                                "User: "))
                 (cond ((equal (setq model (read)) 'cylinder)
                        (princ (concat "RP: enter radius and heigh please"
                                       (implode '(13 10))
                                       "User: "))
                        (setq geometry (list (read) (read)))
                        (princ (concat "RP: enter initial location (X & Y)'
                                       (implode '(13 10))
                                       "User: "))
                        (setq location '(pao (1 0 0 0) (0 1 0 0) (0 0 1 0)
                                         (,(read) ,(read) 0 1)))
                        (add_name (list *hand_full*
                                        model
                                        geometry
                                        location
                                        '(eul 300 0 ,(diff (nth 1 geometry)
                                              0 90 180)
                                        '(eul 0 0 ,(diff (nth 1 geometry) 1
                                              0 90 180))))
                       ((equal model 'ring)
                        (princ (concat "RP: enter two radii"
                                       " and height please      (no commas)
                                       (implode '(13 10))
```

```
                    "User: "))
          (setq geometry (append (if (< (setq intr (read))
                                        (setq extr (read)))
                                     (list intr extr)
                                     (list extr intr))
                                 (list (read))))
          (princ (concat "RP: enter initial location (X & Y)"
                         (implode '(13 10))
                         "User: "))
          (setq location `(pao (1 0 0 0) (0 1 0 0) (0 0 1 0)
                               (,(read) ,(read) 0 1)))
          (add_name (list *hand_full*
                          model
                          geometry
                          location
                          `(eul 300
                                ,(quotient (nth 2 geometry) 2)
                                300
                                0 90 180)
                          `(eul 0 0
                                ,(quotient (nth 2 geometry) 2)
                                0 90 180))))

         ((equal model 'cube)
          (princ (concat "RP: enter X, Y & Z please (no commas
                         (implode '(13 10))
                         "User: "))
          (setq geometry (list (read) (read) (read)))
          (princ (concat "RP: enter initial location (X & Y)"
                         (implode '(13 10))
                         "User: "))
          (setq location `(pao (1 0 0 0) (0 1 0 0) (0 0 1 0)
                               (,(read) ,(read) 0 1)))
          (add_name (list *hand_full*
                          model
                          geometry
                          location
                          `(eul 300
                                ,(quotient (nth 1 geometry) 2)
                                300
                                0 90 180)
                          `(eul (quotient (nth 0 geometry) 2)
                                ,(quotient (nth 1 geometry) 2)
                                ,(diff (nth 2 geometry) 10)
                                0 90 180))))
         (t (princ (concat "RP: misspelling in your answer"
                           (implode '(13 10))) (go lb11)))))))
(go_and_grasp *hand_full*)
(princ (concat "RP: object defined" (maknam '(13 10))))
lb2
(princ (concat "RP: to where ?" (implode '(13 10)) "User: "))
(setq *next_pos* (read))
(cond ((member *next_pos* *names_list*)
       (if (achieve *next_pos*) (go lb2)))
```

```
                (t (new_position *next_pos*) (go_then *next_pos*)))


        lb3
        (princ "RP: do you want to release it ? (yes / no)
User: ")
        (cond ((equal (setq t_aux (read)) 'no) 't)
              ((equal t_aux 'yes) (open_fingers *hand_full*) 't)
              (t (princ "RP: you have misspelled your answer
") (go lb3)))

  ))

; Function "go_and_grasp", move the arm to reach the grasping pos
; of the object mentioned, and does the grasp or close fingers
; operation.

(defun go_and_grasp (t_arg)
  (prog ()
        (go_pos t_arg 'approach)
        (set 'mod 'c)
        (set 'vel *approach_vel*)
        (close_fingers t_arg)
        (return 't)))

; Function "close_fingers", move the hand to the grasping position
; and close the fingers.

(defun close_fingers (t_arg)
        (go_pos t_arg 'grasploc)
        (princ "        CLOSE;
" out_act)
        (putprop (get t_arg 'place) () 'model)
        )

; Function open_fingers, open the hand and move away the hand from
; the grasping position to the approach position.

(defun open_fingers (t_arg)
        (princ "        OPEN;
" out_act)
        (update t_arg)
        (go_pos t_arg 'approach)
        (setq *hand_full* ())
        (set 'mod 'j)
        (set 'vel *joint_vel*)
        )

; Function "go_pos", move to this location

(defun go_pos (t_arg1 t_arg2)
  (prog (t_aux1 p_next)
        (cond ((equal 'approach t_arg2)
               (setq t_aux1 (concat t_arg1 '_a)))
```

```
              ((equal 'grasploc t_arg2)
               (setq t_aux1 (concat t_arg1 '_g)))
              (t (break (in go_pos))))
        (cond ((member t_aux1 *declarations*) ())
              (t (declare_trsf_ptr t_aux1)
                 (gentr t_aux1 (get t_arg1 t_arg2) out_act)))
        (setq p_next (makeposition (list t_arg1 t_aux1)
                                   *robot_set* out_act))
        (move p_next)))

; Function "set", sets whatever is necessary for the environment
; of the robot.

(defun set (t_arg1 t_arg2)
        (cond ((equal t_arg1 'conf)
               (princ (concat " set" t_arg1 "('" t_arg2 "');
") out_act))
              ((equal t_arg1 'time)
               (princ (concat " set" t_arg1 "(" (car t_arg2)
                              ", " (cadr t_arg2) ");
") out_act))
              ((equal t_arg1 'mod)
               (princ (concat " set" t_arg1 "('" t_arg2 "');
") out_act))
              ((equal t_arg1 'vel)
               (princ (concat " set" t_arg1 "(" (car t_arg2)
                              ", " (cadr t_arg2) ");
") out_act))
              (t (break "set function"))))

; Function "update", updates database and current transforms in rccl
; world.

(defun update (t_arg)
  (prog (l_aux1 l_aux2)
        (setq l_aux1 (get t_arg 'location))
        (setq l_aux2 (get *next_pos* 'location))

        (cond ((or (equal (car l_aux2) 'eul)
                   (equal (car l_aux2) 'rot)            ; set for axis rotati
                   (equal (car l_aux2) 'rpy)            ; set for rpy angles
                   (equal (car l_aux2) 'trsl))          ; set for translatior
               (setq l_aux1
                     (reverse
                      (cons
                       (list (cadr l_aux2) (caddr l_aux2) (cadddr l_aux2) 1)
                       (cdr (reverse l_aux1))))))
              ((equal (car l_aux2) 'pao)
               (setq l_aux1 (reverse
                             (cons
                              (last l_aux2) (cdr (reverse l_aux1))))))
              (t (break (in update))))
        (putprop t_arg l_aux1 'location)
```

```
        (princ (concat "          " t_arg "= assigntr(" t_arg ", " *next_pos
") out_act)
        (putprop t_arg *next_pos* 'place)
        (putprop *next_pos* (get t_arg 'model) 'model)
        ))
```

```
; file 695c/5print.l
; this file contains the functions related with the print option
; to look through the database.

(defun printrp ()
  (prog (inp)
        lb0
        (princ (concat "RP: what ?      ('name' 'objects' 'positions' 'quit')
                       (ascii 13) (ascii 10) "User: "))
        (cond ((equal (setq inp (read)) 'quit) (return ()))
              ((member inp *names_list*) 't)
              ((equal inp 'objects)
               (loop for item in *names_list*
                     when (get item 'model)
                     collect item into auxlist
                     finally (progl (princ "RP: ")
                                    (princ auxlist)
                                    (princ (concat (ascii 13) (ascii 10)))))
               (return ()))
              ((equal inp 'positions)
               (loop for item in *names_list*
                     when (not (get item 'model))
                     collect item into auxlist
                     finally (progl (princ "RP: ")
                                    (princ auxlist)
                                    (princ (concat (ascii 13) (ascii 10)))))
               (return ()))

              (t (princ (concat "RP: object " inp " is not defined"
                                (ascii 13) (ascii 10))) (go lb0)))
        (setq l_aux1 (get inp 'location))
        (show_contens l_aux1)))

; this part of code is for debugging
;
;       lb1
;       (cond ((get inp 'model)
;              (princ (concat "RP: which position ?  (origin, approach, grasp
;                             (ascii 13) (ascii 10)
;                             "User: "))
;              (cond ((equal (setq inp1 (read)) 'origin)
;                     (setq l_aux1 (get inp 'location)))
;                    ((equal inp1 'approach)
;                     (setq l_aux1 (get inp 'approach)))
;                    ((equal inp1 'grasp)
;                     (setq l_aux1 (get inp 'grasploc)))
;                    (t (princ (concat "RP: error of misspelling"
;                                      (ascii 13) (ascii 10)))
;                                      (go lb1)))
;              (show_contens l_aux1))
;             (t (princ (concat "RP: which position ? (origin, approach)"
;                               (ascii 13) (ascii 10)
;                               "User: "))
;                (cond ((equal (setq inp1 (read)) 'origin)
```

```
;                        (setq l_aux1 (get inp 'location)))
;                       ((equal inp1 'approach)
;                        (setq l_aux1 (get inp 'approach)))
;                       (t (princ (concat "RP: error of misspelling"
;                                         (ascii 13) (ascii 10))) (go lb1))
;                 (show_contens l_aux1)))))

; function "show_contens", give a nice look of the contens of the
; transforms

(defun show_contens (l_arg)
  (cond ((equal (car l_arg) 'eul)
         (princ (concat "RP:   X= " (cadr l_arg) "   Y= " (caddr l_arg)
                        "   Z= " (cadddr l_arg) (ascii 13) (ascii 10)
                        "RP:   PHI= " (caddddr l_arg)
                        "   THETA= " (cadddddr l_arg)
                        "   PSI= " (caddddddr l_arg)
                        (ascii 13) (ascii 10)
                        )))
        ((equal (car l_arg) 'pao)
         (princ "RP:       matrix ")
         (princ (cadr l_arg))
         (terpri) (princ "RP:              ")
         (princ (caddr l_arg))
         (terpri) (princ "RP:              ")
         (princ (cadddr l_arg))
         (terpri) (princ "RP:              ")
         (princ (caddddr l_arg))
         (terpri))
        ((equal (car l_arg) 'rot)
         (princ (concat "RP:   X= " (cadr l_arg)
                        "   Y= " (caddr l_arg)
                        "   Z= " (cadddr l_arg)
                        (ascii 13) (ascii 10)
                        "RP:   VECTOR= " (caddddr l_arg)
                        "   THETA= " (cadddddr l_arg)
                        (ascii 13) (ascii 10))))
        ((equal (car l_arg) 'rpy)
         (princ (concat "RP:   X= " (cadr l_arg) "   Y= " (caddr l_arg)
                        "   Z= " (cadddr l_arg) (ascii 13) (ascii 10)
                        "RP:   ROLL= " (caddddr l_arg)
                        "   PITCH= " (cadddddr l_arg)
                        "   YAW= " (caddddddr l_arg)
                        (ascii 13) (ascii 10))))
        ((equal (car l_arg) 'trsl)
         (princ (concat "RP:   X= " (cadr l_arg) "   Y= " (caddr l_arg)
                        "   Z= " (cadddr l_arg) (ascii 13) (ascii 10))))
        (t (break (in show_contens)))))
```

References

[1] Hayward, V., and Paul, R.P., ``Robot Manipulator Control Under UNIX," The 13th ISIR and Robot VII, Chicago, 1983.

[2] Winston, P.H., ``Artificial Intellegence" Addison-Wesley Publishing Company, 1984

[3] Wilensky, R., ``LISPcraft" W.W.Norton & Company, 1984

[4] Paul, R.P., ``Robot Manipulators: Mathematics, Programming, and Control", MIT Press 1981.

[5] Hayward, V., ``RCCL Version 1.0 and Related Software Source Code" Purdue University, TR-EE83-42, October 1983

[6] Hayward, V., ``Introduction to RCCL: A Robot Control C Library" Purdue University, TR-EE83-43, October 1983

[7] Hayward, V., ``RCCL Version 1.0 and Related Software Source Code" Purdue University, TR-EE83-47, October 1983

Chapter 9

Spatial Planner, A Rule-Based System in Robotics

Y. L. Gu

SPATIAL PLANNER, A RULE-BASED SYSTEM IN ROBOTICS

You-Liang Gu

The two critical problems in spatial planning involve finding a safe position for placing or grasping objects, and discovering a collision-free path for a robot moving through a space littered with obstacles. In the literature, many researchers dedicated in investigating and solving the spatial planning problems, such as motion sequences [1], human spatial reasoning [2], [3], some solutions to Findspace and Findpath problems in restricted domains [4] and highly mathematical and computational characterization [5], [6].

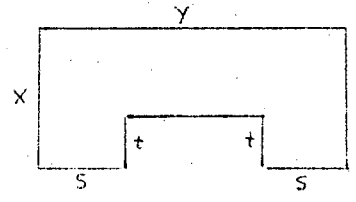
Among these works, the last one is to build a model which actually can solve the trajectory planning and pick-and-place problems, but does so through computational geometry and topology without trying to model intuitiveness or human processes. Therefore, how can we construct a spatial reasoner that is useful to spatial planning systems, and that has the intuitive appeal of providing a characterization or computational analogue for the way human beings reason about three-dimensional space, becomes an attractive problem.

Recently, Bruce Donald [7] and others have proposed for hypothesizing channels in which successful paths may lie. Instead of representing free-space as the compliment of the obstacles, we seek to construct it directly. Based on the concept of channel, the 3 dimension spatial

planning problem may expect to be settled. Now let us employ the channel construction to investigate how a concave object safely go through a given complex channel. The channel is considered in 2-D case, since 3-D is the extension of the 2-D case in the sense of their representation. Although 3-D case has much more complicated problems, such as overlap problem and so forth which do not exist in 2-D case, but once we construct an effective rule based system for 2-D treatment, it is not hard to extend it to the 3-D case by adding more knowledge data and new production rules.

In this project, the objective is to answer following questions:

(1) given a rectangular object with a concave side, shown in Fig. 1, and a complex channel, can the object pass through the channel safely?



$$\begin{cases} x < y, \text{ and} \\ t < x, \text{ and} \\ 2s \leq y. \end{cases}$$

Fig. 1

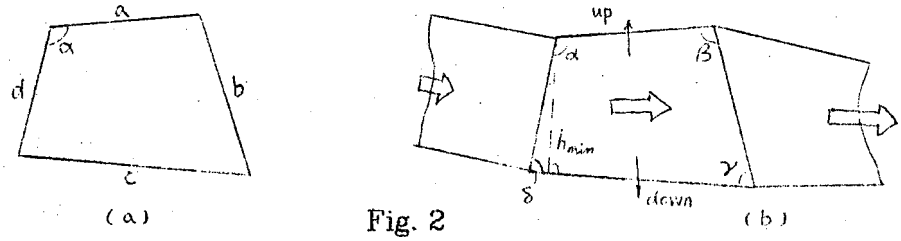
(2) If cannot pass successfully, where was it blocked?

(3) If the object has passed successfully, what orientation in the 2-D plane did it take at each segment of the channel?

Apparently, the objective is to simulate the "desk-mover" in the real life. In order to represent the knowledge, we propose following hypotheses:

(1) The given planar channel can always be divided to many connected segments, each of which is a quadrilateral. We call the common boundary between i-th and i+1st segments as i-th corner of the channel.

(2) For each segment, the lengths of four sides and an internal angle can be known, i.e. (a, b, c, d, α), see Fig. 2 (a), since any quadrilateral is uniquely determined by any five independent parameters. And for the object, four parameters must be given, see Fig. 1, (x, y, s, t).



(a) (b)

Fig. 2

According to these hypotheses, we represent the channel as follows:

(1) Based on the known parameters (a, b, c, d, α) in each segment, compute other four concerned parameters (β, γ, δ, h_{min}) shown in Fig. 2 (b), where h_{min} is the minimum height of the quadrilateral with respect to the direction perpendicular to the forward direction the object is moving to.

(2) For the orientation of the object with concave mouth, we appoint upward and downward directions with respect to the forward direction as shown in Fig. 3. If it is downward oriented, the "updown" in database = 0, if upward, "updown" = 1.

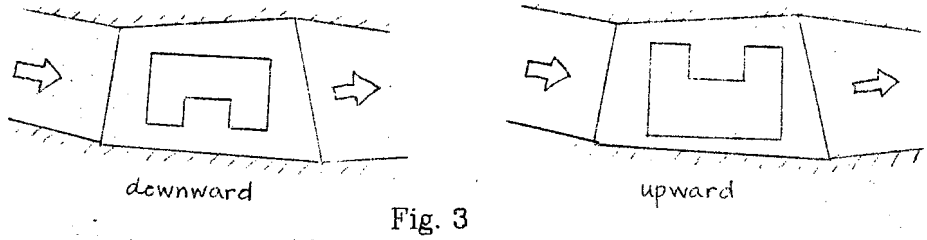


Fig. 3

(3) There are ***eight*** different states to characterize each corner of the channel, i.e. state = 0, 1, 2, 3, 4, 5, 6, 7 shown in Fig. 4.

Fig. 4 See Page 9

So far we have a representation procedure for both given object with a concave mouth and given channel. Thus, a static subdatabase can be constructed in terms of above hypotheses and the appointments.

According to our objective, the "updown" that indicates the orientation of the object in each moving step and "change" which illustrates where the object has to change its orientation are sufficiently used as a desired result. Thus a structure consisting of "updown" and "change" becomes a dynamic subdatabase. And both static and dynamic subdatabases form a global database with short-term memory in the expert system.

The controller of the expert system contains interpreter and organizer. Their function is to decide which rule is currently applied, and to associate the rules which are stored in the knowledge base mentioned later on and the following four subroutines which play an action role:

(1) Determine each corner state of the channel.

(2) Determine whether the object could safely pass each corner of the channel taking advantage of its concave mouth and faced convex corner of the wall.

(3) Determine whether a rectangle could safely pass each corner of the channel when the concave mouth is facing a concave corner of the wall or a line of the wall.

(4) Determine whether a rectangle could safely turn 180 degrees so that its orientation "updown" is changed at some corner of the channel with concave corner of the wall.

Each of four action subroutines is based on pure geometrical principles. Since our problem is of 2-D planning, the computation complexity is reasonable, and the certainty of each decision can be made one. If we are going to extend it to 3-D planning, the calculus of certainty may possibly replace some unmanageable pure geometrical computations. A main reason is that human being never precisely calculate each geometrical size when they are moving a object through the hallway with many obstacles, on the contrary, they only estimate and make inference to complete the task.

The knowledge base includes all productions, or rules, which are:

Rule 1, IF there is any one corner where the object has asked changing "updown" twice,

THEN it is blocked at that corner of the channel.

Rule 2, IF it goes back to entrance, and askes for rotating,

THEN change "updown" once and step plus one.

Rule 3, IF its width exceeds h_{min} of current segment of the channel,

THEN it is blocked at this segment.

Rule 4, IF it asks for rotating, and meets the corner which belongs to one of the states 3, 4, 5, 6, 7,

THEN go backward one step to be rechecked.

Rule 5, IF h_{min} of current segment of the channel is greater than the maximum dimension (i.e. the diagonal size) of the object and it asks for rotating,

THEN change "updown" once and step plus one.

Rule 6, IF it asks for rotating, and meets the corner which is one of the states 0, 1, 2,

THEN check the possibility of rotation by subroutine 4.

Rule 7, IF Rule 3 is false, and the segment of the channel is one of the states 0, 1, 2, and it does not ask for rotating,

THEN go forward one step.

Rule 8, IF Rule 3 is false, and the segment of the channel is of state 3 or 5, and it does not ask for rotating, and its concave mouth is downward,

THEN check that by subroutine 2 and 3.

Rule 9, IF same as Rule 8, except that its concave mouth is upward,

THEN check it by subroutine 3 only.

Rule 10, IF Rule 3 is false, and the segment of the channel is of state 4 or 6, and it does not ask for rotating, and its concave mouth is upward,

THEN check that by subroutine 2 and 3.

Rule 11, IF same as Rule 8, except that its concave mouth is downward,

THEN check it by subroutine 3 only.

Rule 12, IF Rule 3 is false, and the segment of the channel is of state 7, and it does not ask for rotating,

THEN check that by subroutine 2 only.

Rule 13, IF there is a checking ask from Rule 8, and both subroutines 2 and 3 are failed,

THEN it is blocked at current corner of the channel.

Rule 14, IF there is a checking ask from Rule 8, and one of subroutines 2 and 3 is succeeded,

THEN go forward one step.

Rule 15, IF there is a checking ask from Rule 9, and subroutine 3 is failed,

THEN ask for rotating, and go back one step.

Rule 16, IF there is a checking ask from Rule 9, and subroutine 3 is succeeded,

THEN go forward one step.

Rule 17, IF there is a checking ask from Rule 10, and both subroutines 2 and 3 are failed,

THEN it is blocked at current corner of the channel.

Rule 18, IF there is a checking ask from Rule 10, and one of subroutines 2 and 3 is succeeded,

THEN go forward one step.

Rule 19, IF there is a checking ask from Rule 11, and subroutine 3 is failed,

THEN ask for rotating, and go back one step.

Rule 20, IF there is a checking ask from Rule 11, and subroutine 3 is succeeded,

THEN go forward one step.

Rule 21, IF there is a checking ask from Rule 12, and subroutine 2 is failed, and it has made ask for rotating once at current corner of the channel,

THEN it is blocked at current corner of the channel.

Rule 22, IF there is a checking ask from Rule 12, and subroutine 2 is failed, and it has never made ask for rotating at current corner of the channel,

THEN ask for rotating and go back one step.

Rule 23, IF there is a checking ask from Rule 12, and subroutine 2 is succeeded,

THEN go forward one step.

Rule 24, IF the object arrives at the exit of the channel,

THEN it successfully passes whole channel and terminate the procedure.

The above 24 rules and 4 subroutines create a knowledge base of the expert system. However, it needs control strategy to organize them making inference and decision in specified order. As the controller allows the knowledge base "walking" a step, it always fetches some information from the static database, then it makes a medium result telled to the dynamic database. Finally, the expert system will provide us the final conclusion accumurated in the dynamic database.

From whole knowledge base, it can be seen that asserting the object is blocked at somewhere is not often realized by single rule, in many cases, it may go through several rules to determine success or failure, and the dynamic database is frequently revised. This situation may be considered a simple inference process.

The flowchart of the programming is in Appendix. There are two examples, each of which has two different cases to be tested. In the output of computer, first paragraph is to print given object's dimension, the second one indicates the state of each corner of

given channel in terms of the classification code illustrated in Fig. 4, the third one is an explanation for the final conclusion of the dynamic database, and the last one is the final contents of the dynamic database itself which have been entirely expressed by third explanation paragraph. Each example has a diagram below, two cases for each example have only one difference at some internal angle.

The rule-based expert system for the spatial planning in Robotics is based on human being's mode of thinking to be simulated in computer, thus it avoids a lot of complicated, even redundant geometrical computation more or less, and lies more stress on reasoning and inference, hence it is superior to the highly mathematical approachs. If it is attached some learning process, i.e. a dynamic rule-knowledge base, then it works more perfectly and is more close to the human's thought. I think that really to solve the 3-D planning problem is only by developing and completing the artificial intelligence approach.

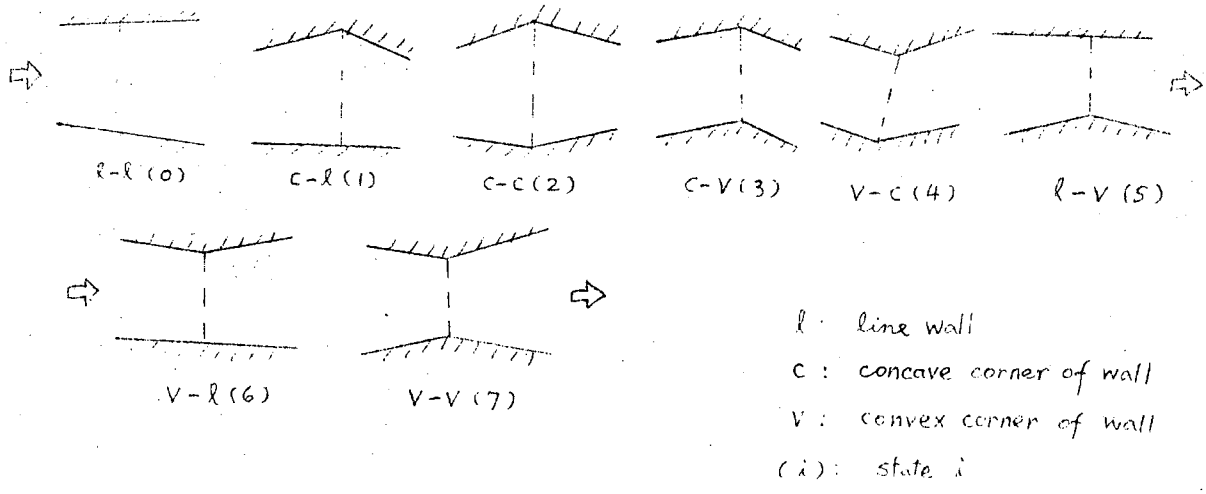


Fig. 4

References

[1] Winston, Patrick *Artificial Intelligence*, Addison-Wesley, Reading, Mass., 1977.

[2] Minsky, Marvin "A Framework for Representing Knowledge," in *The Psychology of Computer Vision*, ed. Patrick Winston, McGraw-Hill Book Co., New York, 1975.

[3] Kuipers, Benjamin J. "Representing Knowledge of Large-Scale Space," Massachusetts Institute of Technology AI Lab, AI-TR-418, 1977.

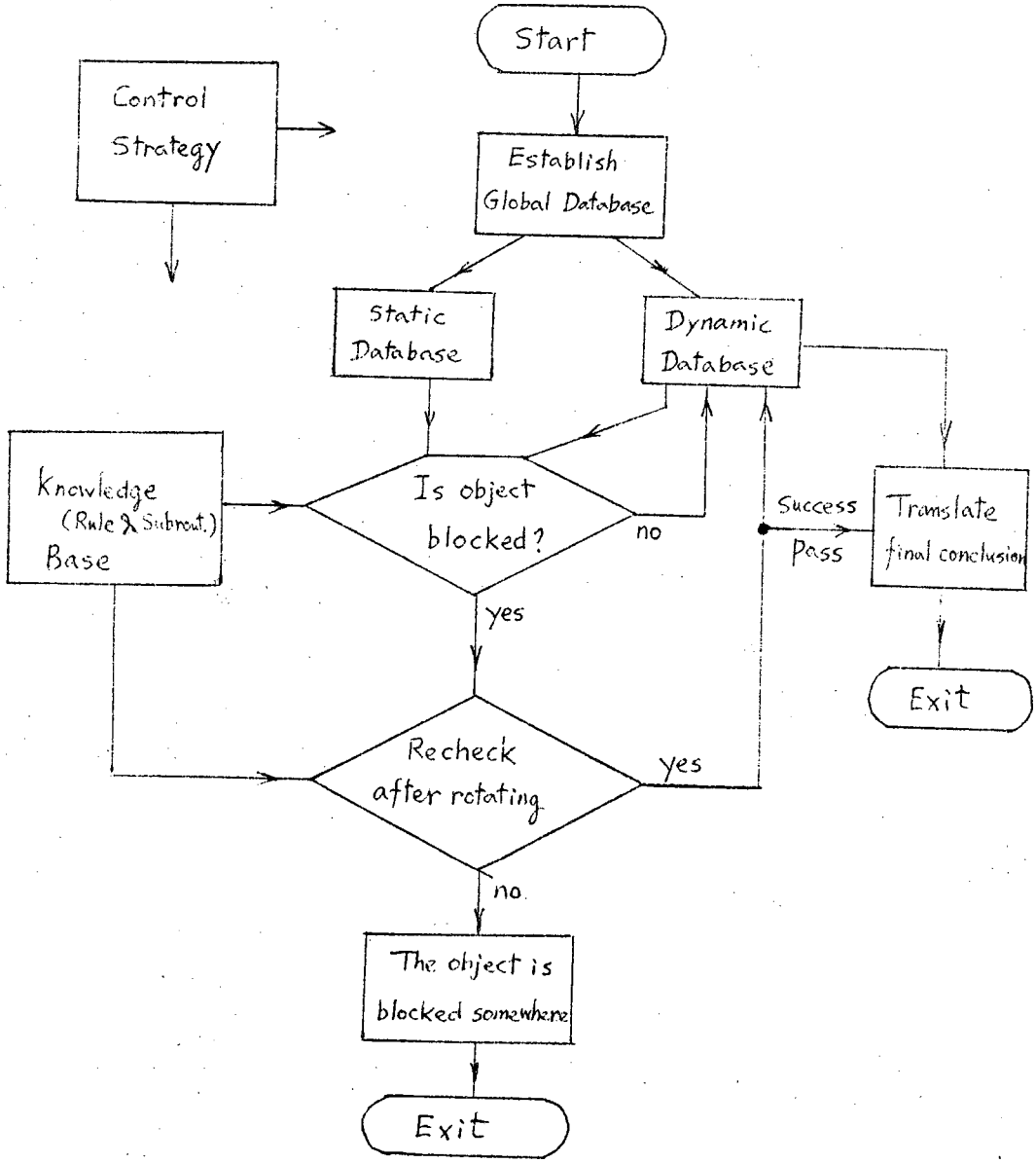
[4] Brooks, Rodney "Solving the Find-Path Problem by Representing Free Space as Generalized Cones," MIT AI Lab, Tech. Report 647, 1982.

[5] Lozano-Perez, Tomas "Spatial Planning: A Configuration Space Approach," MIT AI Lab Memo No.605, 1980.

[6] Schwartz, Jacob T. & Micha Sharir "On the 'Piano Movers' Problem," 1982.

[7] Donald, Bruce "Knowledge Representation and Spatial Reasoning Using Channels," MIT AI Lab, 1982.

Appendix.



Flowchart.

OUTPUT ONE

The concave object's dimension:
x= 2.30, y= 3.00, s= 0.80, t= 0.80.

The corners' states of the channel:
entrance 3 3 4 2 4 4 3 exit.

The object is blocked at 5 -th corner.

The final database is
updown= 0 0 0 0 2 2 2 2
change= 0 0 0 0 0 0 0 0

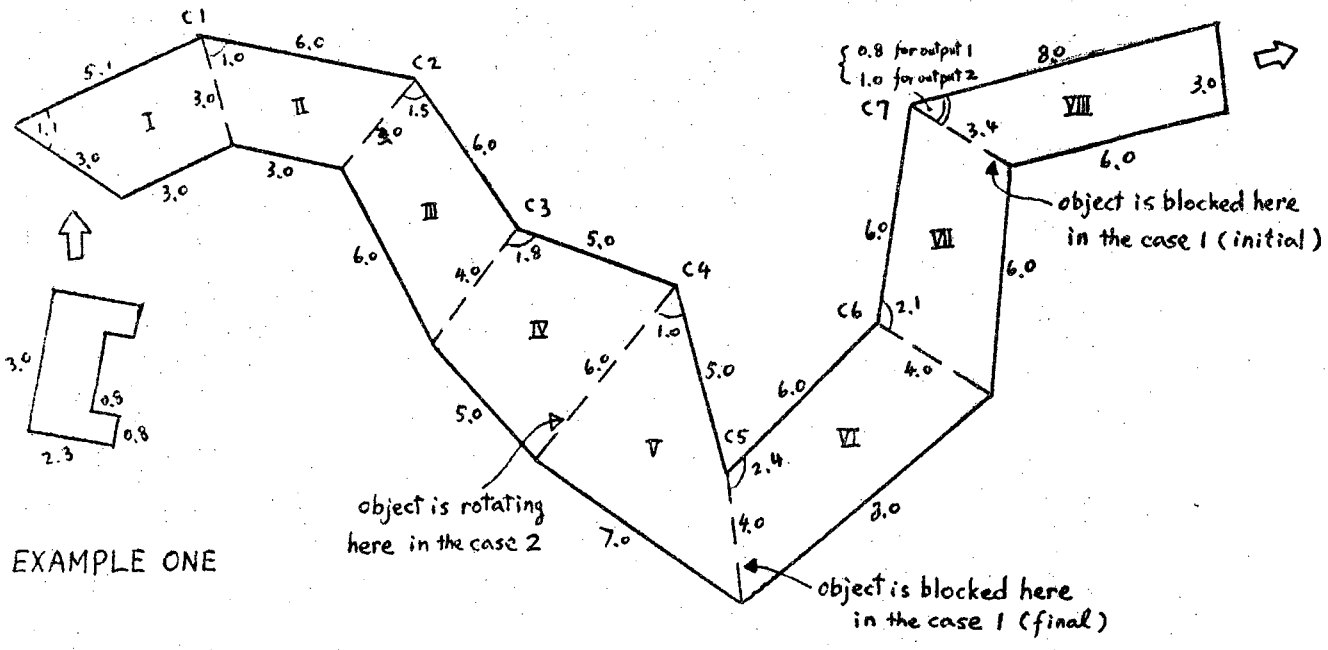
OUTPUT TWO

The concave object's dimension:
x= 2.30, y= 3.00, s= 0.80, t= 0.80.

The corners' states of the channel:
entrance 3 3 4 2 4 4 3 exit.

The object sets into the channel with concave-downward, change the object to be concave-upward at 4 -th corner, and finally, the object safely passes throush the channel and arrive at the destination with concave-upward.

The final database is
updown= 0 0 0 1 1 1 1 1
change= 0 0 0 0 0 0 0 0



EXAMPLE ONE

OUTPUT THREE

The concave object's dimension:
x= 2.30, y= 3.00, s= 0.80, t= 0.80.

The corners' states of the channel:
entrance 3 7 2 7 3 4 4 4 exit.

The object sets into the channel with concave-upward,
and finally, the object safely passes through the channel
and arrive at the destination with concave-upward.

The final database is
updown= 1 1 1 1 1 1 1 1 1
change= 0 0 0 0 0 0 0 0 0

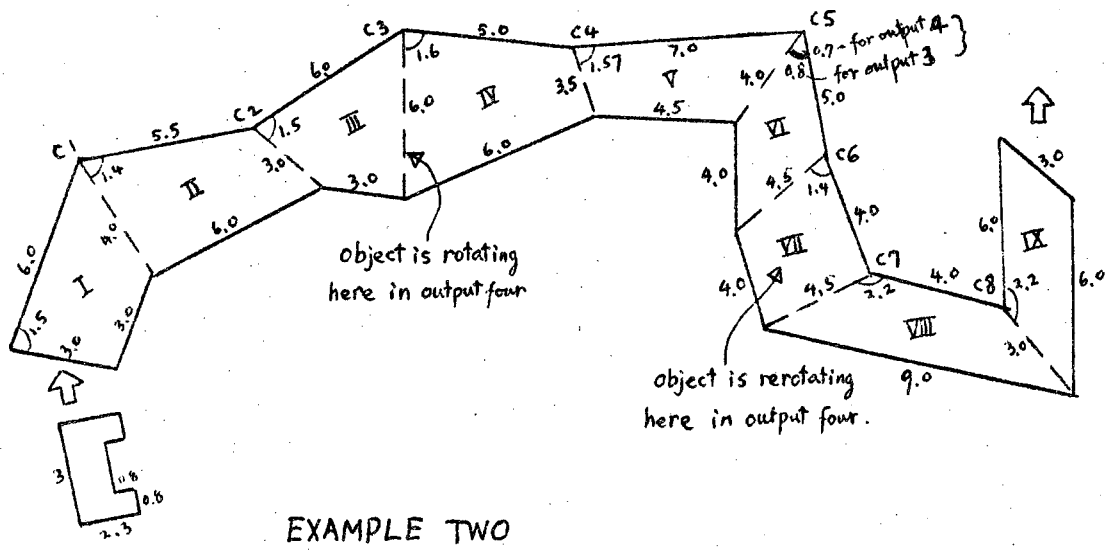
OUTPUT FOUR

The concave object's dimension:
x= 2.30, y= 3.00, s= 0.80, t= 0.80.

The corners' states of the channel:
entrance 3 7 2 7 3 4 4 4 exit.

The object sets into the channel with concave-upward,
change the object to be concave-downward at 3 -th corner,
change the object to be concave-upward at 7 -th segment,
and finally, the object safely passes through the channel
and arrive at the destination with concave-upward.

The final database is
updown= 1 1 0 0 0 1 1 1 1
change= 0 0 0 0 0 1 0 0 0



EXAMPLE TWO

****

PART III

Vision

Chapter 10

An Application of Expert System Approach in Detection of Boundaries Between Textures

K. B. Eom and C. Chatterjee

AN APPLICATION OF EXPERT SYSTEM APPROACH IN DETECTION OF BOUNDARIES BETWEEN TEXTURES

Kie-Bum Eom and Chanchal Chatterjee
School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

ABSTRACT

This project is concerned with developing a method for detecting boundaries in images of multiple textures. This work in two parts presents a statistical model-based method of detecting boundaries in its first part and a rule-based method of joining them to obtain the true boundary in its second part. The statistical method consists of dividing the entire image into a series of strips. A long correlation model is fitted to each of these strips and a cost function is evaluated to detect the existence and the location of the boundary, if any in the strip. Ths method gives a series of lines and points some of which consist the true boundary and some of which do not. The rule-based method consists of a set of 42 line/point joining rules. These rules take into account the location, orientation, length and variance of each line with respect to other lines and join them to form the true boundary. The method also deletes spurious lines and points. A considerable amount of success is achieved by this method.

Keywords: statistical model, rule-based, long correlation model.

AN APPLICATION OF EXPERT SYSTEM APPROACH IN DETECTION OF BOUNDARIES BETWEEN TEXTURES

Kie-Bum Eom and Chanchal Chatterjee

PART I

1. Introduction

Edge detection problem was the one of the most important problems in Image Processing for long time and there were many approaches in this area. However, the detection of the boundaries between textures was not as easy as the usual edge detection and only a few approaches were made recently.

Chen and Pavlidis (1972) used co-occurence matrix for detecting edges between textures. Thompson(1977) used the edge enhancement operators (e.g., Sobel, Robert's) for textures. He suggested that the boundaries between different textures be characterized by an abrupt change in the textural features of the neighboring regions. However, the features can be vary considrably at some interior areas inside the texture and the change at the boundary can be small depending on the textures. These areas will be classified incorrectly.

Statistical model based approach to edge detection in texture is developed by Kashyap and Bauer(1983). Kashyap and Khotanzad(1984) also suggested a statistical edge detection algorithm in textures. The image was divided by small windows and the parameters for each window was estimated by a given statiatical image model. The image model used was Simultaneous Autoregressive (SAR) models in both cases. From the parameters which was estimated for each window, the location of the boundary between textures was estimated.

Even though these short memory statistical model (e.g. ,SAR) based edge detection methods give promising results, the ***long memory*** model is known to be a better choice in modelling real images(Lapsa 1982). The ***long memory*** model requres smaller number of parameters but better in modelling long

memory characteristics. A consistent estimation algorithm for this long memory model was also developed by Kashyap and Eom(1984). These results for long memory model was applied for detecting edges between textures in this report. The whole image is divided by a small (4 by 24) size of non overlapping windows for x and y directions. The parameters for this window, assumed that there is an edge at the location n, is estimated by the consistent estimation algorithm and the cost function for each location is computed. The cost function will be explained later. The edge location is estimated by the location which minimize this cost function.

The detection of no edge in the window is estimated by the low variance of the cost function $J(n)$ over all range of n. This method gives a good estimation of the location of the existing edges, but produces many spurious edges or deletion of the true edges. For even simple type of edges, a clean-up operation is required. The objective of this report is to show a possibility of applying expert system for clean up these spurious edges.

There were also some approaches in applying expert system for the edge detection and low level segmentation (Nazif 1984). However, the system performance also depends on the feature extraction and it is more important in this low level processing. The macro edge detection like the edges between textures requires full utilization of the feature extraction methods and the application of expert system. We are going to make an approach which is combined the advantages of these two methods in the next chapters.

2. Two Dimensional Long Correlation Model

The importance of *long correlation* model in image processing is well discussed by Lapsa (1982). Suppose that $\{x(i,j)\}$ is a two dimensional time series sequence of *long correlation* model and $\{w(i,j)\}$ is a two dimensional white noise sequnce with variance ρ for generating $\{x(i,j)\}$. Then it is represented by the following equation.

$$x(i,j) = (1-z_1^{-1})^{-d_1}(1-z_2^{-1})^{-d_2}w(i,j) \tag{2.1}$$

$$i=1,\cdots,N_1-1\ ,\ j=1,\cdots,N_2-1$$

where z_1^{-1} and z_2^{-1} are unit delay operators in x and y directions respectively.

Let $\{X(k,l)\}$ and $\{W(k,l)\}$ be DFT's of $\{x(i,j)\}$ and $\{w(i,j)\}$ obeying (2.1) respectively. Then

$$X(k,l) = (1-e^{\frac{-j2\pi k}{N_1}})^{-d_1}(1-e^{\frac{-j2\pi l}{N_2}})^{-d_2}W(k,l)$$

$$= e^{\frac{-j\pi kd_1}{N_1}}\ e^{\frac{-j\pi ld_2}{N_2}}\ [2j\sin(\frac{\pi k}{N_1})]^{-d_1}[2j\sin(\frac{\pi l}{N_2})]^{-d_2}W(k,l)$$

$$,\qquad k=0,1,\cdots,N_1-1\ ,\ \ l=0,1,\cdots,N_2-1$$

$$|X(k,l)| = [2|\sin(\frac{\pi k}{N_1})|]^{-d_1}[2|\sin(\frac{\pi l}{N_2})|]^{-d_2}|W(k,l)| \tag{2.2}$$

$$,\qquad k=0,1,\cdots,N_1-1,\ \ l=0,1,\cdots,N_2-1$$

By applying logarithm operator to modulus of $X(k,l)$, DFT of $x(i,j)$, a linear equation in parameter d_1 and d_2 is obtained.

$$\log|X(k)| = -d_1\log|2\sin(\frac{\pi k}{N_1})| -d_2\log|2\sin(\frac{\pi l}{N_2})|+\log|W(k,l)| \tag{2.3}$$

$$, \quad k=0,1,\cdots,N_1-1, \quad l=0,1,\cdots,N_2-1$$

Now define $\alpha=-E[\log|W(k,l)|]$ and $V(k,l)=\log|W(k,l)|+\alpha$. Thus $V(..)$ has zero mean and (2.3) can be rewritten as follows by using above definitions.

$$\log|X(k,l)| = -d_1\log|2\sin(\frac{\pi k}{N_1})| - d_2\log|2\sin(\frac{\pi l}{N_2})| - \alpha + V(k,l) \qquad (2.4)$$

Then the least square estimator $\hat{\vartheta}$ is the argument which minimizes the functional $J(\vartheta)$.

$$J(\vartheta) = \sum_{k=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}}(\log|X(k,l)| + d_1\log|2\sin(\frac{\pi k}{N_1})| + d_2\log|2\sin(\frac{\pi l}{N_2})| + \alpha)^2$$

$$= \sum_{k=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}}(\log|X(k,l)| - \vartheta^T Z(k,l))^2 \ , \ \vartheta = (d_1,d_2,\alpha)^T. \qquad (2.5)$$

where

$$Z(k,l) = \begin{bmatrix} -\log|2\sin(\frac{\pi k}{N_1})| \\ -\log|2\sin(\frac{\pi l}{N_2})| \\ -1 \end{bmatrix} \qquad (2.6)$$

Then the least square estimator $\hat{\vartheta}$ is obtained by the standard formula.

$$\hat{\vartheta} = (\hat{d},\hat{\alpha})^T = (\sum_{k=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}} Z(k,l)Z^T(k,l))^{-1}(\sum_{k=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}} Z(k,l)\log|X(k,l)|) \qquad (2.7)$$

where $Z(k,l)$ is given in (2.6).

We will introduce the following two lemmas on transformed noise sequences, $\{|W(k,l)|\}$ and $\{\log|W(k,l)|\}$. The lemma 1 is from Brillinger(1981).

Lemma 1: The sequence $\{|W(k,l)|, k=0,1,\ldots,\frac{N_1}{2}, l=0,1,\ldots,\frac{N_2}{2}\}$, the magnitude of discrete fourier transformed white Gaussian noise sequence, is also an white

sequence with the following Rayleigh densities.

$$f_{|W(k,l)|}(w) = \begin{cases} \frac{2w}{\rho N_1 N_2} \exp\left(\frac{-w^2}{\rho N_1 N_2}\right) & w \geq 0 \\ 0 & otherwise \end{cases} \tag{2.8}$$

For further investigation of sequence $\{\log|X(k,l)|\}$, we need to find characteristics of the noise term $\log|W(k,l)|$. The following Lemma 2 gives another nice property of this noise sequence.

Lemma 2: The transformed frequency domain noise sequence $\{\log|W(k,l)|, k=0,1,\ldots,\frac{N_1}{2}, l=0,1,\ldots,\frac{N_2}{2}\}$ is a white sequence and

$$E[\log|W(k,l)|] = -\frac{\gamma}{2} + \frac{1}{2}\log\rho N_1 N_2 \tag{2.9}$$

$$Var[\log|W(k,l)|] = \frac{\pi^2}{24} \tag{2.10}$$

where γ is Euler's constant (= 0.5772157) and ρ is the variance of time domain noise sequence $\{w(i,j)\}$.

Proof: The above expressions are obtained by using the density function in Lemma 1 and standard integration formula.

■

The following theorems show the consistency of the estimators which was derived in the last section.

Theorem 1: The estimators $\hat{\alpha}_1$ and $\hat{\alpha}_2$ and α given by (2.7) are unbiased and consistent in the mean square sense with variances $\frac{2}{N_1 N_2}$, $\frac{2}{N_1 N_2}$ and $\frac{\pi^2}{6N_1 N_2}$ respectively for large N_1 and N_2.

Proof : It will be shown in appendices.

The estimator α in Theorem 1 is an estimator of a function of noise variance ρ. The derivation of a consistent estimator of ρ will be given in Theorem 2. In Theorem 2 we will show that α is a Gaussian random variable by using the Central Limit Theorem. The consistency of a proposed estimator will be shown by the property of the moment generating function of Gaussian random variable. The proposed estimator of ρ is an exponential function of α and the MGT of Gaussian random variable α depends only on the mean and variance. Since the mean and variance of α are given in Theorem 1, the consistency of ρ can be proved.

Theorem 2: With α given in (2.7), define an estimator of ρ as following

$$\rho = \frac{1}{N_1 N_2}\exp(\gamma - 2\alpha - \frac{\pi^2}{3N_1N_2}) \tag{2.11}$$

Then ρ is unbiased and consistent in mean square sense with variance $\frac{\rho^2\pi^2}{3N_1N_2}$.

Proof: By the Central Limit Theorem, $\alpha \sim N(\alpha, \frac{\pi^2}{6N_1N_2})$. Thus, the Theorem 2 can be proven easily. ∎

3. Statistical Edge Detection Algorithm

Our approach of boundary detection between textures is quite different from the traditional edge detection algorithms. First the image which is composed with several textures are divided into a large number of horizontal and vertical strips. (e.g., 4x24) The strip size is kept as small as possible to have less than one edge in a strip but to have close estimator to the real parameter value. The existence of the edge in an window is decided by using least squares argument. If the decision is made for existence of an edge, the column (or row) number of the edge location is estimated.

The detail of this algoithm will be explained in below. Consider an RxC horizontal window where the each entry of the array is the pixel intensity. It is assumed that the image was made up of two kinds of textures as shown below with o indicating one texture and x indicating the other.

```
o o o o o o o o o o o o o x x x x x x x x x x
o o o o o o o o o o o o o x x x x x x x x x x
o o o o o o o o o o o o o x x x x x x x x x x
o o o o o o o o o o o o o x x x x x x x x x x
```

Figure 1

The boundary between two textures are located at the n^0 th column, n^0 is unknown. It is also assumed that the textures on the left side of n^0 and right hand side of n^0 follows the following 2-D LC model with the parameters d_1^l, d_2^l, ρ^l and d_1^r, d_2^r, ρ^r.

$$y(i,j) = (1-z_1^{-1})^{-d_1}(1-z_2^{-1})^{-d_2}w(i,j)$$

where $\{w(i,j)\}$ is a 2-D white noise sequence with variance ρ.

The variances ρ^l and ρ^r can be considered as intensity of residuals for fitting to 2-D LC models. Therefore if the segment (left or right) contains an

edge, then the value of ρ will be larger than that of the segment without an edge.

Now consider the following cost function

$$J(n) = \rho^l(n) + \rho^r(n)$$

This cost function will give minimum value at $n = n^0$ by above argument. So, the edge location can be estimated by minimizing the above cost function.

The detection of the existence of an edge is more difficult than the estimation of the location of an edge with the knowledge of existence of an edge. Consider an window with one edge. Since this window is consisted of two different types of textures, the residual density for all n, except for n = n^0 in this window will be larger than that of a window which is consisted of only one type of textures. In other words, the average of ratio $J(n)$ over $J(n^0)$ will be larger for the window with windows. This can be used as an measure of existence of an edge in the window. The following algorithm summarizes this edge detection method.

Algorithm:

1. Divide the input image by non overlapping small strips(4x24) in x and y directions.
2. For each window (consider horizontal window without loss of generality), consider the imaginary vertical edges at n(from 4 to 16) and compute the followings for each n.

   a) Compute the consistent estimators of ρ for left of n($\rho^l(n)$) and for right of n($\rho^r(n)$).

   b) Compute the cost function $J(n) = \rho^l(n) + \rho^r(n)$.
3. Compute the average of $J(n)$ and the find the value of n which minimize $J(n)$(nmin).

4. If the ratio of the average to the minimum $J(n)$ exceeds the threshold, then accept this n as a true edge location.

This algorithm is simple but very good in picking true edge locations. However, the decision of edge existence does not give very good results and gives many unwanted edges. In the next section, we will show the expert system based approach for this clean-up operation.

4. Clean-Up Operation

As we discussed in the last section, the statistical edge detection algorithm for finding boundaries between textures generates many unwanted spurious edges. It is necessary for using clean-up operations or edge tracking operations to recognize the shape of edges by the computer. Even though there are many known approaches for this operations, these approaches are not very flexible to do such a job for reasonably complex edges.

4.1. Conventional Cleanup Operation

4.1.1. Shrinking and Expanding

The noisy edges detected by the algorithm discussed in the last section can be cleaned by simple shrinking and expanding operations. First widen the edge widths by adding three more one's to the both sides of detected edges. This converts the original image contains edges to the image contains blocks which will be connected each other if two blocks are located close enough. Therefore the converted image contains long elongated parts and some isolated blocks. We can consider these isolated blocks as noisy edges which need to be deleted.

Since the height of unit edge is four pixels, the isolated blocks are eliminated by two consecutive simultaneous deletion of border points. This operation is called as shrinking. After shrink two times, the elogngated parts are recovered by two consecutive adding of neighbor points. This operation is called as expanding. after these operations, we will have only long elongated but thick edges. The final cleaned edges are obtained by thinning operation.

The advantage of this method is the simplicity and good performance of cleaning in horizontal and vertical edges. However, this operation can delete slant edges.

4.1.2. Edge Tracking

The detected edges can be cleaned by edge tracking algorithm with a pre determined threshold on the reward function associated with length and curvature of the edges. We employed a row to row tracking method. Specifically, in each row we accept any point whose pixel intensity is 'one'. In addition, we also employed a search technique in tracking edges. Once a point (x,y) on the y-th row has been accepted, we search for all neighbor points with 'one'. The reward function is designed to increase by the length of connected edges, and to decrease by the curvature and the length of disconnected portions.

The above procedure is applied recursively and total reward at the end of the line seqment is compared with a predetermined threshold to accept as a legal edge.

4.2. Expert System Based Cleanup Operation

The if-then rule based expert system approach will be used for this cleanup operation by the following reasons.

1) There are many knowledges in line tracking and edge linking methods. A method is required for organizing these to function together.

2) The expert system is so flexible to accomodate more knowleges in the future. The statistical boundary detection method which is explained in the last section is not the only method to find the boundaries between textures. This system can be more powerful by accomodating the other algorithms in the future.

3) It is necessary to represent some heuristic algorithms by rule based knowledge representation methods.

4) In some cases, quantitative evaluation of features are not practical.

The following is the block diagram of the expert system.

In the above diagram, statistical edge detection and feature detection is explained in the last section. We will explain mostly on knowledge representation in this section.

Knowledge representation will largely depend on the composition of the rules constituting the model. A rule has the following format:

IF conditions THEN actions

The left hand side is composed of a set of *conditions* evaluated on the data. The *actions* on the right hand side specify particular modifications to the data.

Each rule depicts a certain situation that might be present in the data. A process will match a rule on the features. Only line rules are constructed because line rules are good for edge linking and spurious edge deleting but easy to implement by known algorithms. More details on rules and inference will be given in Appendices.

5. Experimental Results

The edge detection algorithm and the Clean-up algorithms which explained in the last sections are applied to real images which is composed of different kinds of textures. All images which used in this experiment are 128x128. First image is composed with straw texture and herringbone weave texture and have a boundary at 50 th column (Fig. 2). Second image is composed with straw texture and wood grain texture and also have boundary at 50-th column (Fig. 3). Third image is composed with straw texture herringbone weave texture and wood grain texture and straw texture is on the left of 50-th column herringbone weave is on the right above of 64-th row and wood grain texture is on the right below (Fig. 4).

The statistical edge detection algorithm is applied to the above three kinds of textures. Figures 5, 6 and 7 shows the result for Figures 2, 3, and 4 in x direction. Figure 8 shows the result for Figure 4 in y direction. We can easily observe that these images contains so many unwanted noises. It seems that it is not easy to remove all unwanted edges from these detected edges. To examine the performances of the clean-up operations which we discussed in the last section, these clean-up operations are applied to the above three kinds of images.

As we discussed in the last section, the Shrinking and Expanding is the simplest algorithm to clean-up unwanted noises. This algorithm is applied to Figure 5 and 6 and obtained the cleaned edges Figure 9 and 10. The first image generates a connected edge at the right position of 50-th column and the second image generates an edge at the right position but it lose the upper portion of the edge. This confirms the discussion at the last section on this algorithm.

The edge tracking algorithm with searching capability is more complicate than the shrinking and expanding algorithm but better in edge linking and removing unwanted noises. This algorithm is applied in x direction to Figure 7

and obtained Figure 11. This algorithm is also applied in y direction to Figure 8 and obtained an image Figure 12. As we can observe thise figures, this algorithm is better than the first algorithm but still have unwanted edge at Figure 12.

Expert system is very flexible as we discussed before, but the cost of using this is very expensive. For example, the time to cleanup one image by expert system approach is more than 20 times than that of conventional methods. However if we consider to accomodate more detection algorithms for detecting boundaries between textures and find a best edge between many possible choices of edge combinations, one possible solution is the expert system. The result of applying the expert system approach on these images will be given in the part 2.

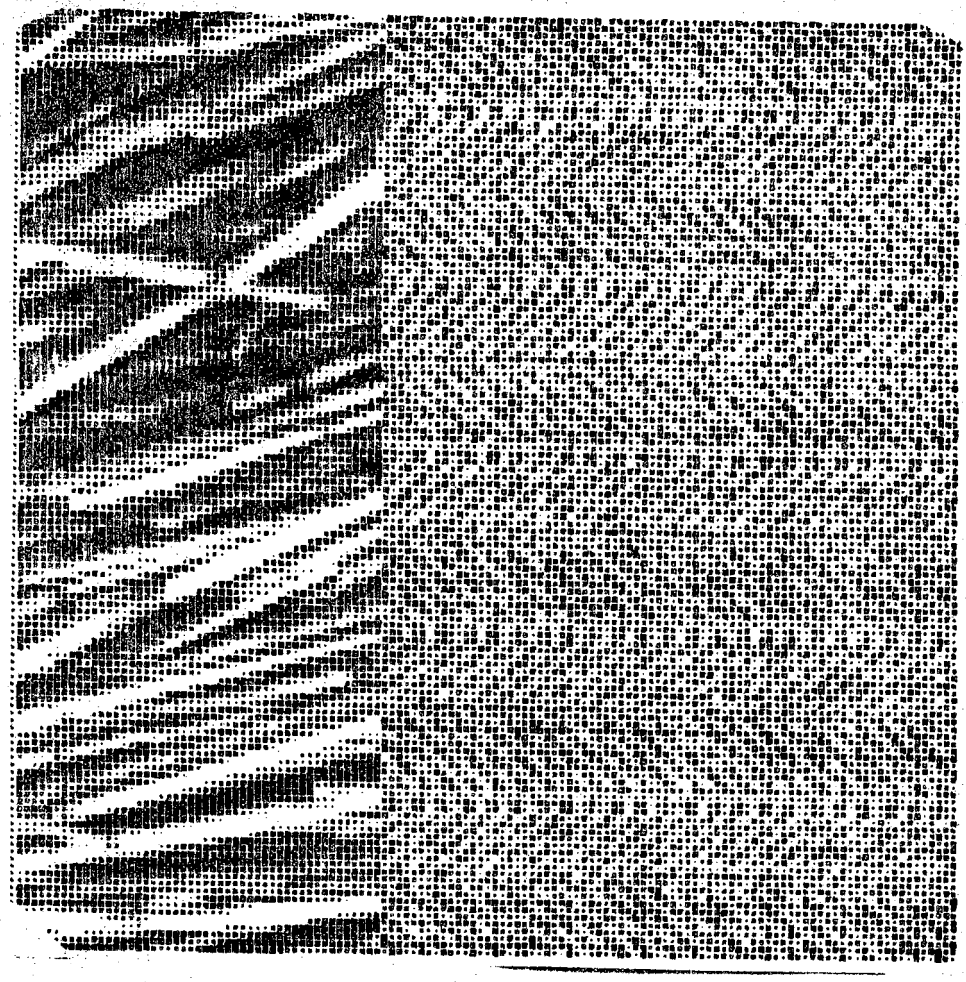


Figure 2. First test image

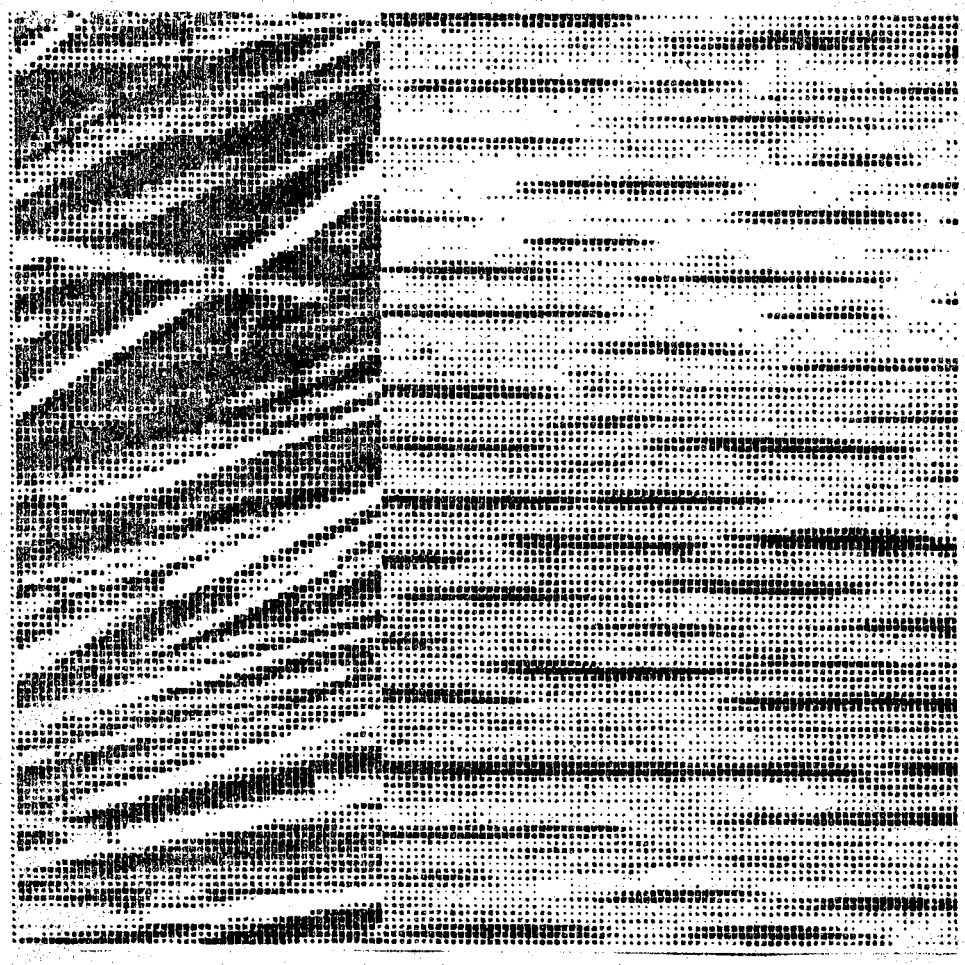


Figure 3. Second test image

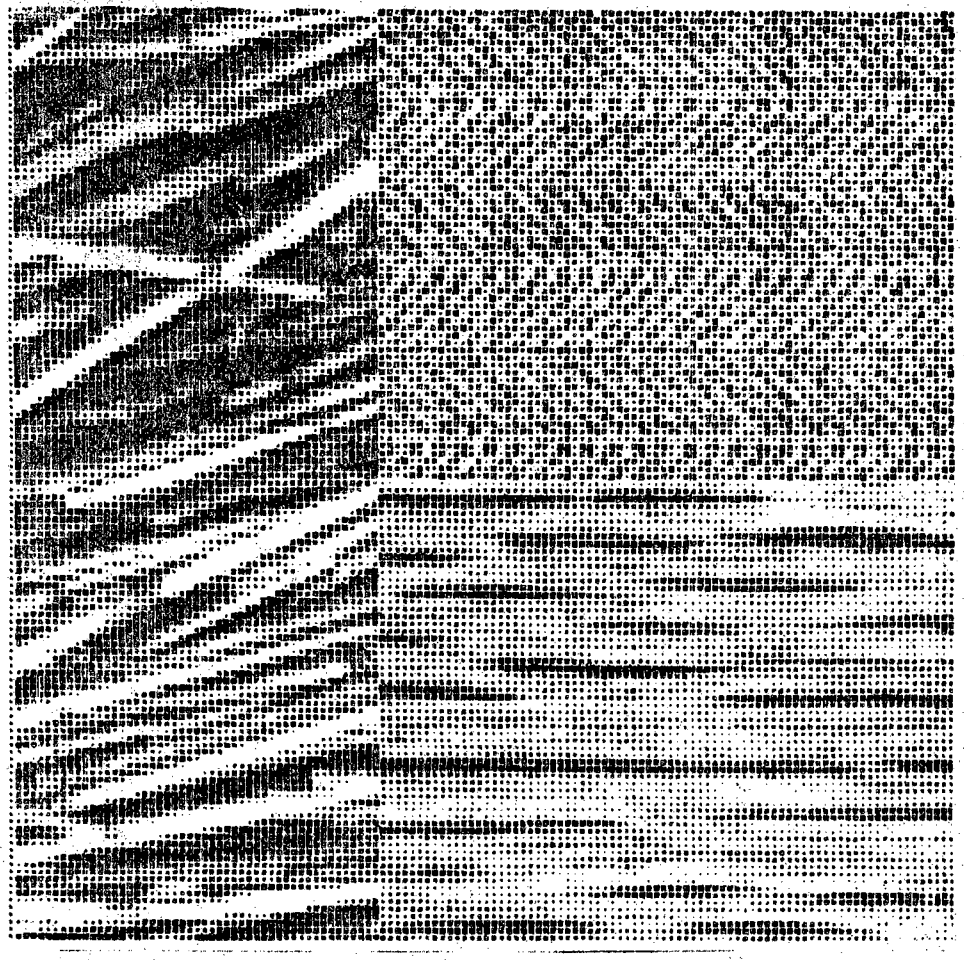


Figure 4. Third test image

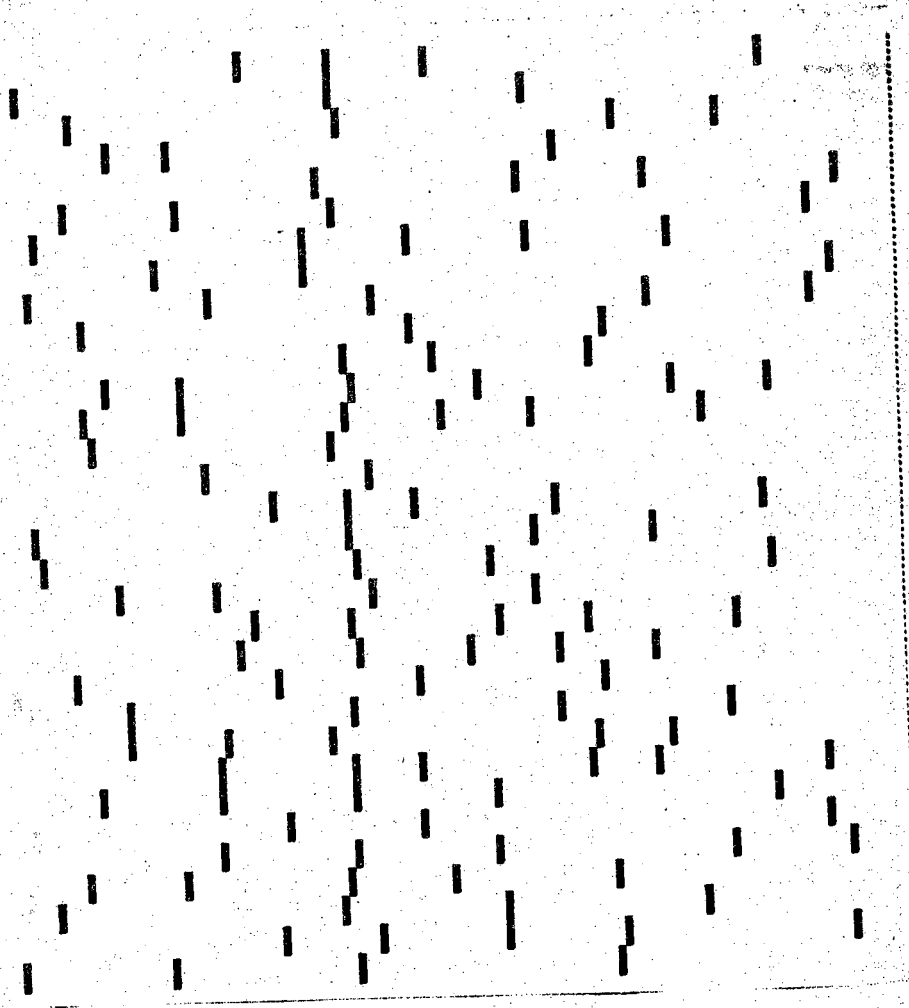


Figure 5. Detected edges for the first image in x direction

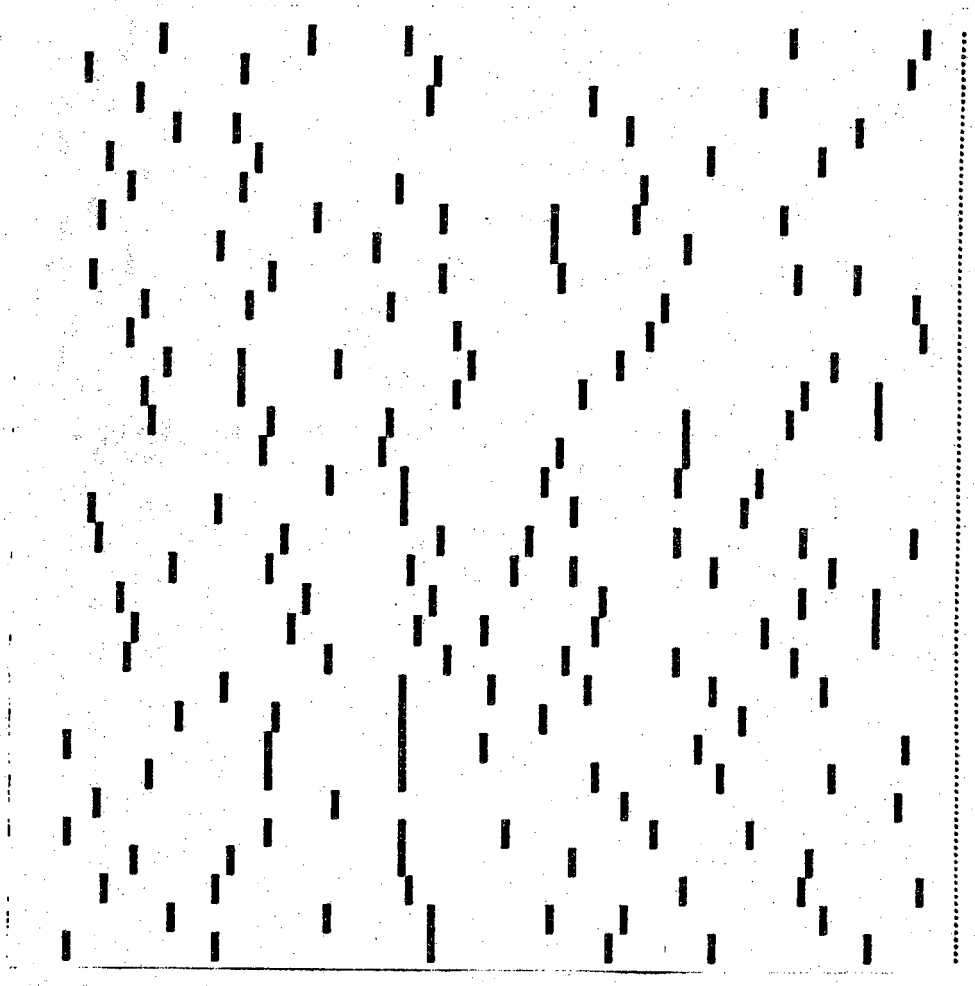


Figure 6. Detected edges for the second image in x direction

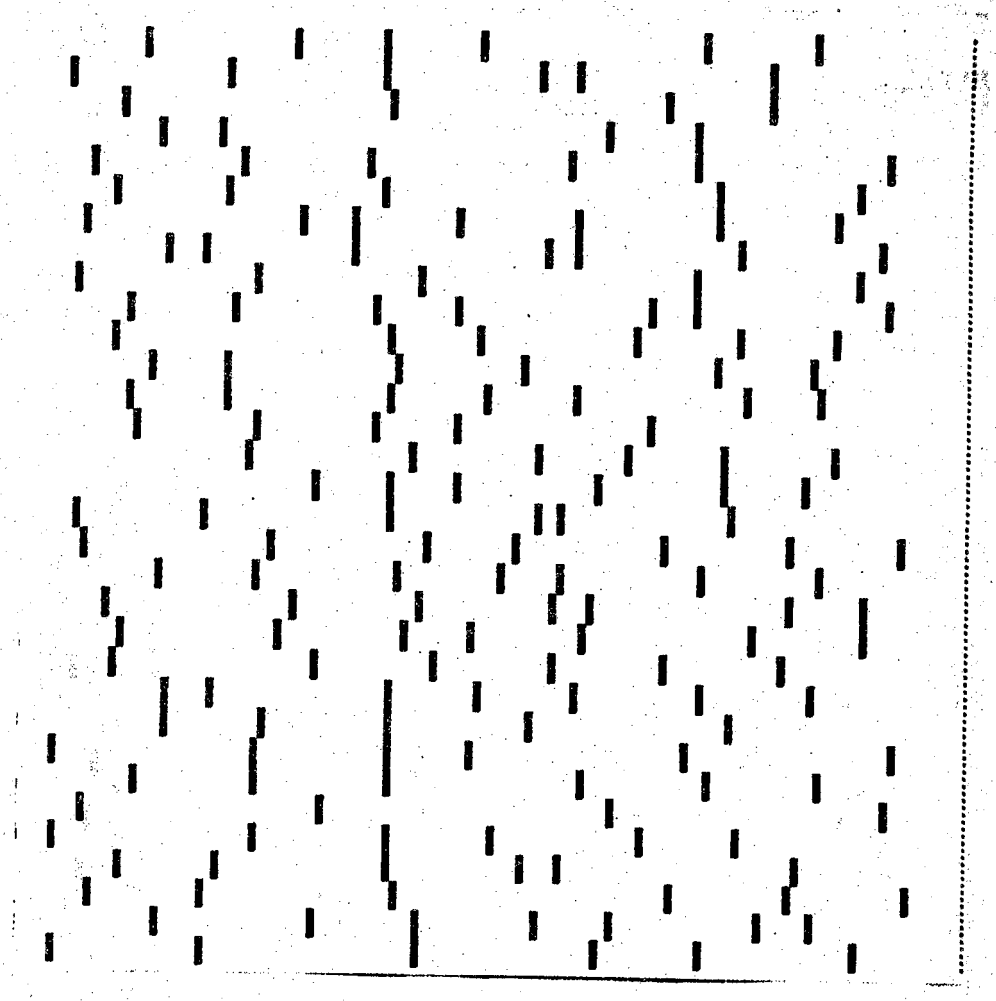


Figure 7. Detected edges for the second image in x direction

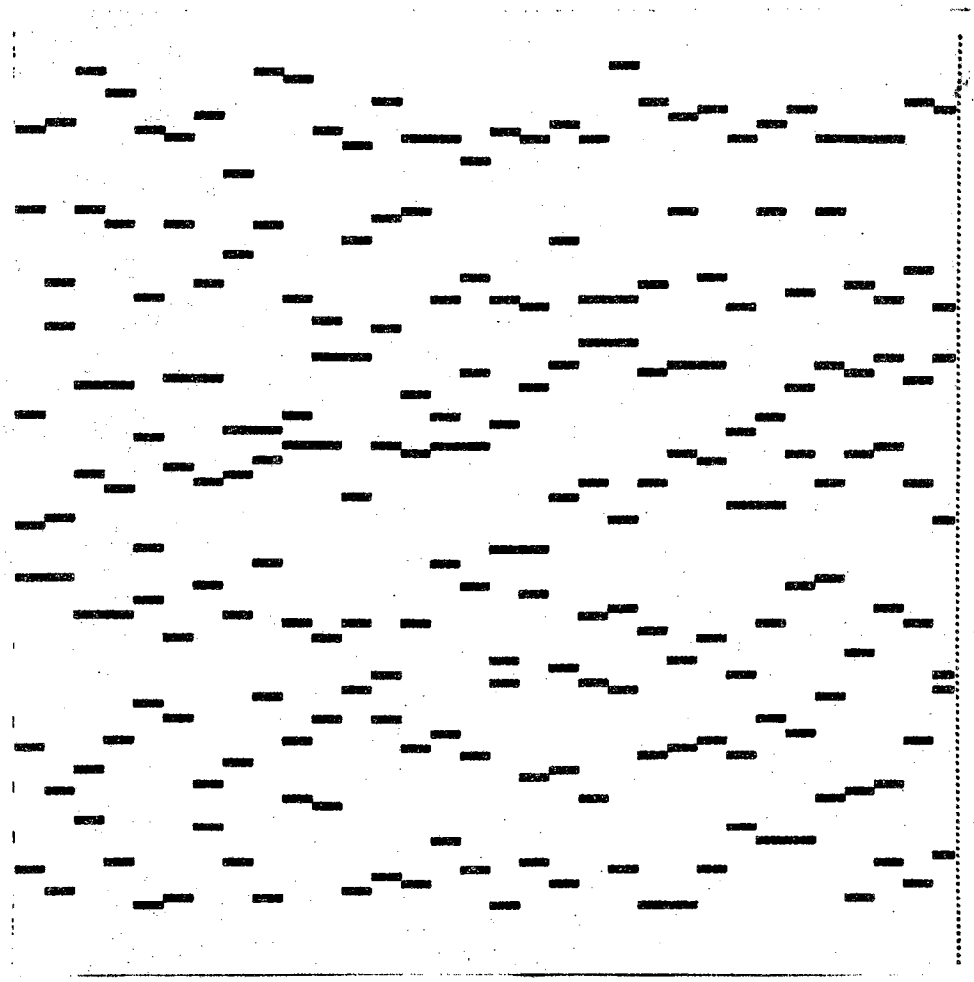


Figure 8. Detected edges for the third image in x direction

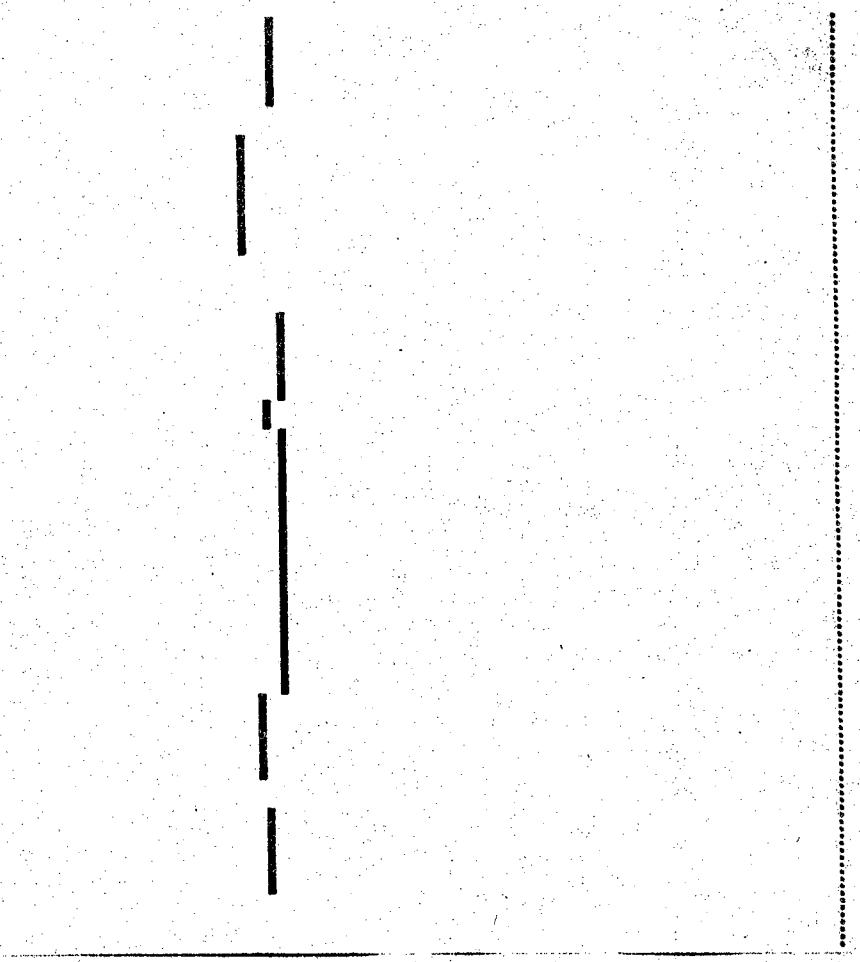


Figure 9. Cleaned edges by shrinking and expanding for figure 5

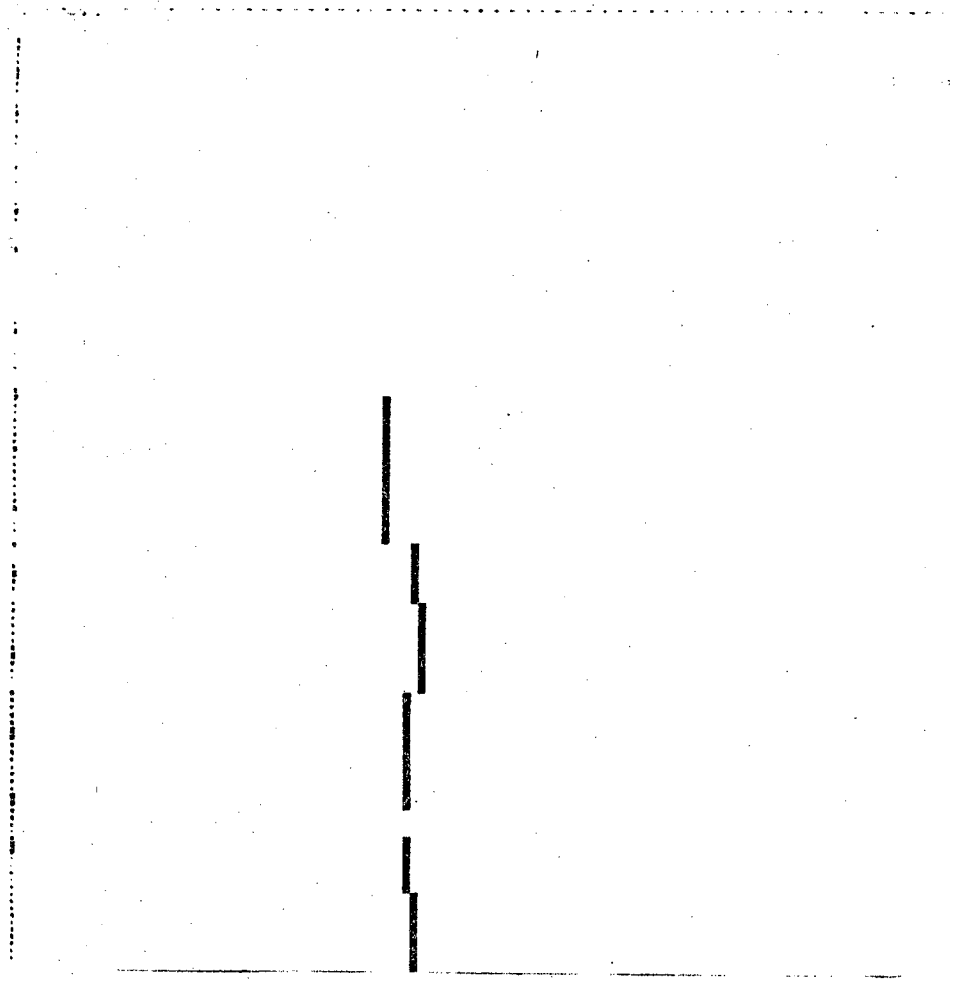


Figure 10. Cleaned edges by shrinking and expanding for figure 6

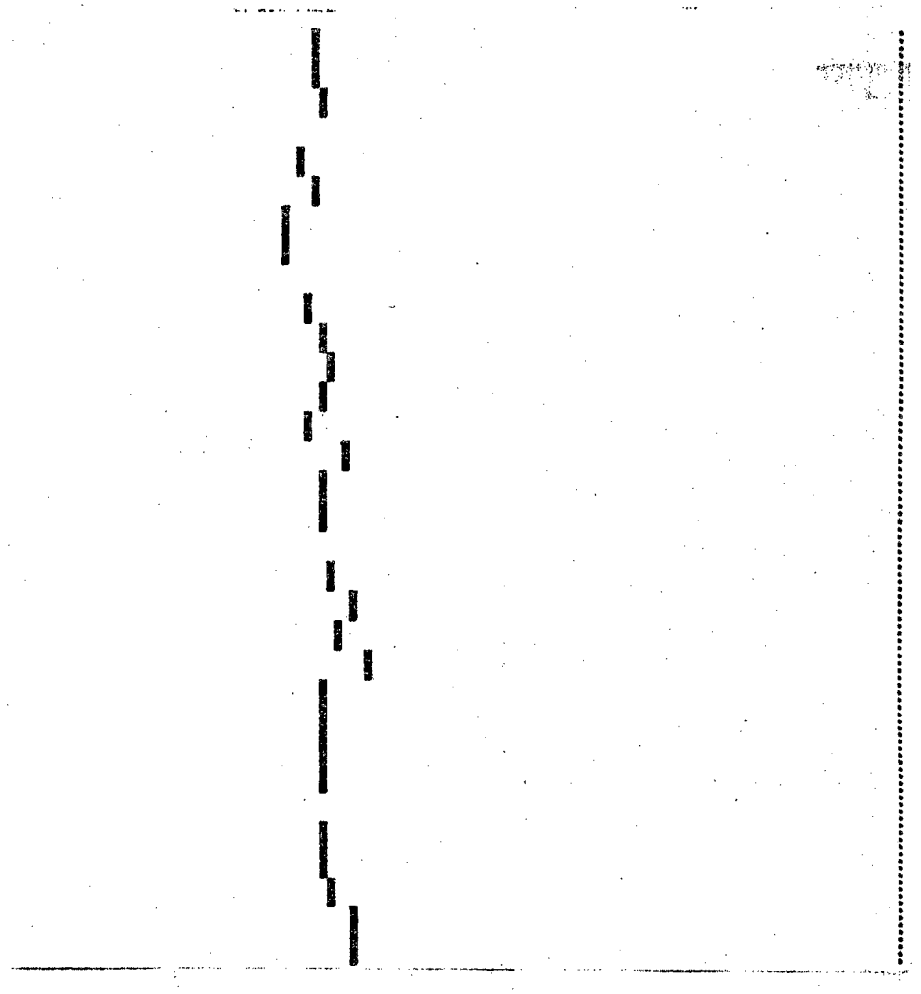


Figure 11. Cleaned edges by edge tracking for figure 7

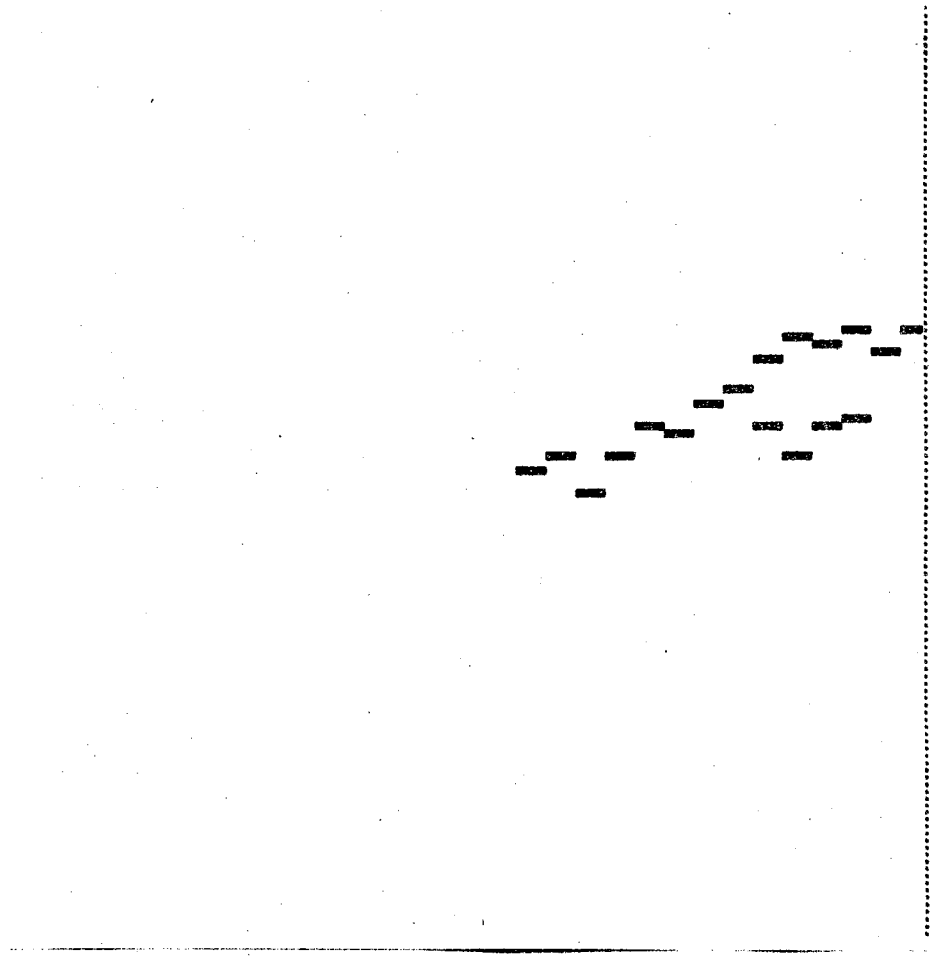


Figure 12. Cleaned edges by edge tracking for figure 8

6. Discussions and Conclusions

Statistical boundary detection method for textures detects boundary position very well but generates many unwanted edges. These noise like edges can be removed by conventional methods and an expert system approach. Conventional methods clean these noise-like edges well but still make errors and not so flexible to do for complex edges. Expert system approach gives very good results in overall performance but cost for computation is very high.

Appendices

A.1. Line Analysis Rules

Rule L1: IF the line end point is open
the distance to the line in front is short
THEN join the lines by forward expansion.

Rule L2: IF the line end point is open
the distance to the line in front is not long
the angle between two lines are small
THEN join the lines by forward expansion.

Rule L3: IF the line start point is open
the distance to the line behind is short
THEN join the lines by backward expansion.

Rule L4: IF the line start point is open
the distance to the line behind is not long
the angle between two lines are small
THEN join the lines by backward expansion.

Rule L5: IF the line end point is open
the distance to the line in front is not very long
the two lines have the same region to the left
the two lines have the same region to the right
THEN join the lines by forward expansion.

Rule L6: IF the line end point is open
the distance to the line in front is not very long
same region left of line 1 and right of line 2

the two lines have the same region to the right

THEN join the lines by forward expansion.

Rule L7: IF the line end point is open

the distance to the line in front is not very long

the two lines have the same region to the left

same region right of line 1 and left of line 2

THEN join the lines by forward expansion.

Rule L8: IF the line end point is open

the line length is short

the distance to the nearest line in front is large

THEN delete the line segment.

Rule L9: IF the line end point is open

the line length is short

the distance to the line in front is not short

the angle between two lines is large

THEN delete the line segment.

Rule L10 IF the line end point is open

the line length is short

the distance to the line in the back is not short

the angle between two lines is large

THEN delete the line segment.

Rule L11 IF the line end point is open

the line length is not short

THEN extend the line forward.

Rule L12 IF the line end point is open

the number of regions to the right is not small

the number of regions to the left is not small

THEN extend the line forward.

Rule L13 IF the line end point is open

the line length is not short

there is one region to the left of the line

there is one region to the right of the line

region to the left and right are not the same

THEN extend the line forward.

A.2. Proof of Theorem 1

Proof:

$$E[\hat{\vartheta}] = \Big(\sum_{k=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}} Z(k,l)Z^T(k,l)\Big)^{-1}\Big(\sum_{k=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}} Z(k,l)E[\log|X(k,l)|]\Big)\ , \textit{using (2.7)}$$

$$= \Big(\sum_{k=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}} Z(k,l)Z^T(k,l)\Big)^{-1}\left[\sum_{k=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}}\begin{bmatrix} d_1 \\ d_2 \\ -E[\log|W(k,l)|]\end{bmatrix}\right],\ \textit{using (2.3)}$$

$$= \begin{bmatrix} d_1 \\ d_2 \\ -E[\log|W(k,l)|]\end{bmatrix} \qquad \text{(A.1)}$$

So, $\hat{\vartheta}$ is an unbiased estimator.

By using (2.7) and (A.1),

$$\hat{\vartheta}-E[\hat{\vartheta}] = \Big(\sum_{k=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}} Z(k,l)Z^T(k,l)\Big)^{-1}\left[\sum_{k=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}}\begin{bmatrix} 0 \\ 0 \\ -\log|W(k,l)|+E[\log|W(k,l)|]\end{bmatrix}\right]$$

$$E[(\hat{\vartheta}-E[\hat{\vartheta}])(\hat{\vartheta}-E[\hat{\vartheta}])^T]$$

$$= E\left\{\Big(\sum_{k=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}} Z(k,l)Z^T(k,l)\Big)^{-1}\left[\sum_{k=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}} Z(k,l)Z^T(k,l)\Big)^{-1}\begin{bmatrix} 0 \\ 0 \\ -\log|W(k,l)|+E[\log|W(k,l)|]\end{bmatrix}\right]\right.$$

$$\left.\left[\sum_{m=1}^{\frac{N_1}{2}}\sum_{n=1}^{\frac{N_2}{2}}\begin{bmatrix} 0 \\ 0 \\ -\log|W(m,n)|+E[\log|W(m,n)|]\end{bmatrix}\right]^T Z(m,n)Z^T(m,n)\right)\left(\sum_{m=1}^{\frac{N_1}{2}}\sum_{n=1}^{\frac{N_2}{2}} Z(m,n)Z^T(m,n)\right)^{-1^T}\right\}$$

$$= \Big(\sum_{k=1}^{\frac{N_1}{2}}\Big(\sum_{l=1}^{\frac{N_2}{2}} Z(k,l)Z^T(k,l)\Big)^{-1}$$

$$\left[\sum_{k=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}}\sum_{m=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}} Z(k,l)Z^T(k,l)\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & E[(\log|W(k,l)|-E[\log|W(k,l)|])\\ & & (\log|W(m,n)-E[\log|W(m,n)|])^T]\end{bmatrix} Z(m,n)Z^T(m,n)\right]$$

$$(\sum_{m=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}} Z(k,l)Z^T(k,l))^{-1}$$

,using the whiteness of $\{\log|W(k,l)|\}$ which was proved in Lemma 2

$$= (\sum_{k=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}} Z(k,l)Z^T(k,l))^{-1}$$

$$\left[\sum_{k=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}} Z(k,l)Z^T(k,l)\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & Var[\log|W(k,l)|]\end{bmatrix} Z(k,l)Z^T(k,l)\right]$$

$$(\sum_{m=1}^{\frac{N_1}{2}}\sum_{n=1}^{\frac{N_2}{2}} Z(m,n)Z^T(m,n))^{-1} \qquad \text{(A.2)}$$

Now define $S(k) = \log|2\sin(\frac{\pi k}{N})|$.

Then

$$Z(k,l) = \begin{bmatrix}-S(k)\\ -S(l)\\ -1\end{bmatrix} \qquad \text{(A.3)}$$

and

$$Z(k,l)Z^T(k,l) = \begin{bmatrix}S^2(k) & S(k)S(l) & S(k)\\ S(k)S(l) & S^2(l) & S(l)\\ S(k) & S(l) & 1\end{bmatrix}$$

Rewrite the following by using (A.3).

$$\sum_{k=1}^{\frac{N_1}{2}} Z(k,l)Z^T(k,l)\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & Var[\log|W(k,l)|]\end{bmatrix} Z(k,l)Z^T(k,l)$$

$$= Var[\log|W(k,l)|]\sum_{k=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}} Z(k,l)Z^T(k,l) \qquad (A.4)$$

Substitute (A.4) into (A.2) and obtain the following equation.

$$E[(\hat{\vartheta}-E[\hat{\vartheta}])(\hat{\vartheta}-E[\hat{\vartheta}])^T] = \frac{\pi^2}{24}(\sum_{k=1}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{N_2}{2}} Z(k,l)Z^T(k,l))^{-1} \qquad (A.5)$$

For large N, the summations of the right hand side in (A.5) can be replaced by integrations.

i.e.,

$$\sum_{k=1}^{\frac{N}{2}}(\log|2\sin(\frac{\pi k}{N})|) \approx \frac{N}{2}\int_0^1(\log 2 + \log\sin(\frac{\pi t}{2}))dt$$

$$= \frac{N}{2}(\log 2) + \frac{N}{\pi}\int_0^{\frac{\pi}{2}}\log\sin u\,du$$

$$\approx \frac{N}{2}(\log 2 - \log 2) = 0 \qquad (A.6)$$

, using the result in the standard table(Gradshteyn 1980).

Similarly,

$$\sum_{k=1}^{\frac{N}{2}}(\log|2\sin(\frac{\pi k}{N})|)^2$$

$$= \sum_{k=1}^{\frac{N}{2}}(\log^2 2 + 2\log 2\log|\sin(\frac{\pi k}{N})| + (\log|\sin(\frac{\pi k}{N})|)^2)$$

$$\approx \frac{N}{2}\int_0^1(\log^2 2 + 2\log 2\log|\sin(\frac{\pi t}{2})| + (\log|\sin(\frac{\pi t}{2})|)^2)dt$$

$$= \frac{N}{2}\log^2 2 + \frac{2N}{\pi}\int_0^{\frac{\pi}{2}}\log\sin u\,du + \frac{N}{\pi}\int_0^{\frac{\pi}{2}}(\log\sin u)^2 du$$

$$= \frac{N}{2}\log^2 2 + \frac{2N}{\pi}(-\log 2) + \frac{\pi}{2}\left[(\log 2)^2 + \frac{\pi^2}{12}\right]$$

$$= \frac{\pi^2}{24}N \qquad \text{(A.7)}$$

Therefore by (A.5),(A.6) and (A.7),

$$\sum_{k=1}^{2}\sum_{l=1}^{2} Z(k,l)Z^T(k,l) = \frac{N_1N_2}{4}\begin{bmatrix} \frac{\pi^2}{12} & 0 & 0 \\ 0 & \frac{\pi^2}{12} & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{(A.8)}$$

Substitute (A.8) into (A.5) and obtain the following.

$$E[(\hat{\vartheta}-\vartheta)(\hat{\vartheta}-\vartheta)^T] = \begin{bmatrix} \frac{2}{N_1N_2} & 0 & 0 \\ 0 & \frac{2}{N_1N_2} & 0 \\ 0 & 0 & \frac{\pi^2}{6N_1N_2} \end{bmatrix} \qquad \text{(A.9)}$$

Therefore,

$$Var[\hat{\alpha}_1] = \frac{2}{N_1N_2}$$

$$Var[\hat{\alpha}_2] = \frac{2}{N_1N_2}$$

$$Var[\alpha] = \frac{\pi^2}{6N_1N_2}$$

■

References

Bickel, P.J. and Doksum, K.A. (1977) *Mathematical Statistics* Holden-Day, Inc.

Brillinger, D.R. (1981) *Time Series, Data Analysis and Theory*, Expanded Edition, Holden Day Inc.

Chen, P.C., and Pavlidis, T. (1979) "Segmentation of Textures by using Co-occurence Matrix and Split-and-Merge Algorithm," *Computer Graphics and Image Processing*, Vol. 10, pp 172-182

Granger, C.W.J. and Joyeux, R. (1980) "An Introduction to Long-Memory Time Series Models and Fractional Differencing," *Journal of Time Series Analysis*, 1 , 15-29.

Gradshteyn, I.S. and Ryzhik, I.M. (1980) *Table of Integrals, Series, and Products*, Academic Press.

Hosking, J.R.M. (1981) "Fractional Differencing," *Biometrika*, **68** , 165-176.

Kashyap, R.L. and Bauer, R.D., (1983) "A Statistical Model Based Approach to Edge Detection in Textures," *Proc. 21st Allerton Conf.*, Chicago, Illinois, October

Kashyap, R.L. and Eom, K.B. (1984) "A Consistent Estimation of Parameters in the Long-Memory Time Series Model," *Submitted for Publication.*

Kashyap, R.L. and Khotanzad, A., (1984) "A Stochastic Model Based Technique for Texture Segmentation," *Proc. the Seventh International Conference on Pattern Recognition*, July-August, Montreal.

Kashyap, R.L. and Rao, A.R. (1976) *Dynamic Stochastic Models from Empirical Data*, Academic Press.

Kashyap, R.L. and Lapsa, P.M. (1982) "Long-Correlation Models for Random Fields," *Proc. 1982 Conf. on Pattern Recognition and Image Processing*, Las

Vegas, June.

Lapsa, P.M. (1982) *New Models and Techniques for the Synthesis, Estimation, Identification and Segmentation of Random Fields*. Ph.D. Dissertation, School of Electrical Engineering, Purdue University, August.

Lawrance, A.J. and Kottegoda, N.T. (1977) "Stochastic Modelling of Riverflow Time Series," *Journal of Royal Statistical Society*, **A 140** , 1-47.

Mandelbrot, B.B. and van Ness, J.W. (1968) "Fractional Brownian Motions, Fractional Noises and Applications," *S.I.A.M. Review*, **10** , 422-437.

Mandelbrot, B.B. (1967) "Some Noises with $1/f$ Spectrum, A Bridge Between Direct Current and White Noise." *I.E.E.E. Trans. on Information Theory*, **IT-13** , 289 - 298.

Nazif, A.M. and Levine, M.D. (1984) "Low Level Image Segmentation: An Expert System," *I.E.E.E. Trans. on Pattern Analysis and Machine Intelligence*, September, Vol. PAMI-6, No. 5,

Rao. C.R. (1967) *Linear Statistical Inference and Its Applications.*, John Wiley & Sons.

Thompson, W.B., (1977) "Textural Boundary Analysis," *I.E.E.E. Trans. on Computers*, pp. 272-276

Winston, P.H. and Horn, B.K.H. (1981), *LISP*, Addison-Wesley Publishing Co.

PART II

I. DATA BASE

We are given an image consisting of a set of lines and points. This image is generated from an image of multiple textures by running an edge detection algorithm. There are two well known approaches to boundary-detection of textures. They are :

(1) finding features distinguishing one texture from another and

(2) using statistical rules like the maximum likelihood rules.

In the first approach, different methods have been attempted by eminent researchers including the fitting of random field models to the textures and using their parameters for classification. The second approach is based on dividing the image into a series of strips and the boundary, if any, between the textures in the strip is obtained by fitting random field models and using a likelihood rule. We, therefore, first execute a boundary detection algorithm on the original images and obtain the "extracted images" consisting of lines and points, some of which constitute the true boundaries and some of which do not.

Creation of Data Base

In order to create the knowledge base for the image we extract the important and necessary information out of the extracted image. The necessary information used to draw the lines in the image to segment different regions of textures are :

(1) continuous lines separating different parts of the image,

(2) isolated points in the image.

For an isolated point its coordinates in the image gives its exact location. For the line the following parameters have been stored in an array called "line" :

i_s = starting i coordinate of the line

j_s = starting j coordinate of the line

d_s = starting direction of the line

i_f = finishing i coordinate of the line

j_f = finishing j coordinate of the line

d_f = finishing direction of the line

l = length of the line measured as follows :

length of the line joining the following sets of points are all 1

$x\ x \qquad x$

$\qquad x$

length of the line joining the following sets of points are $\sqrt{2}$

$x \qquad x$

$x \qquad x$

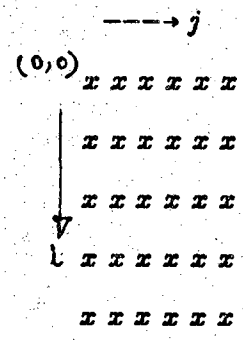
The above values are represented in the array "line" as follows for each line and point in the image.

$$i_s\ j_s\ d_s\ i_f\ j_f\ d_f\ l$$

Directions are represented as follows :

135	90	45
−1	x	1
−45	−90	−135

The i and j coordinates of the image are represented as follows :



The image is stored in array "im" with "1" for a point in the image and "0" for a blank. The program "draw-line" calls the subroutine "scan" which recursively scans the image row-wise starting from the origin (0,0) joining points along a line and returns the array "line". The flowcharts for the main program and the subroutine "scan" is shown in figures 1.2 and 1.3 respectively. The variable "prev" stands for the direction of the line joining the current point from the point before it and the variable "pres" stands for the direction from the current point to the next point.

The rules and metarules used to generate the database are as follows :

Meta Rule 1

IF there is a point in the image and

it has not been scanned

THEN scan it.

Rule -1

IF there is a point in the image within d_8 (defined in figure 1.1) distance of another point and

it has not been scanned

THEN these two points are "connected".

Rule –2

IF there are connected points

THEN find successive connected points and draw the complete line.

Rule –3

IF a point in the image has no other point in its d_8 distance

THEN it is an "isolated point".

Rule –4

IF a point has more than one connected points and

they have not been scanned before

THEN scan thru each one of them to find all branching lines.

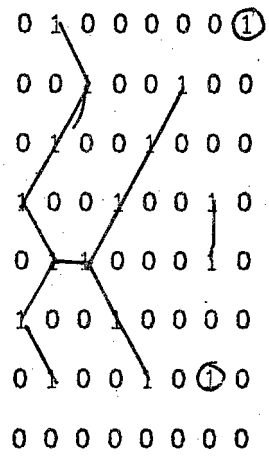
x x x

x o x

x x x

Figure-1.1 Figure shows points at d_8 distance from the point o

The above-mentioned algorithm is demonstrated on the following image :



This image has the following features :

(1) one line starting at (0,1) and branching into three lines ending at (6,1), (6,4), (1,5)

(2) one line of length 1 between (3,6) and (4,6)

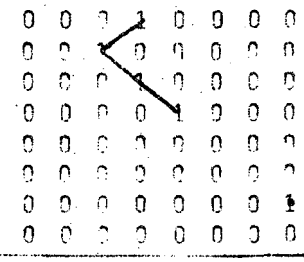
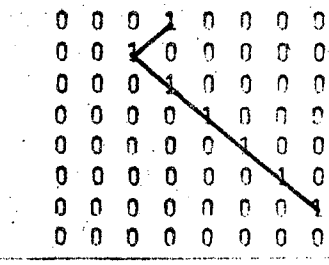
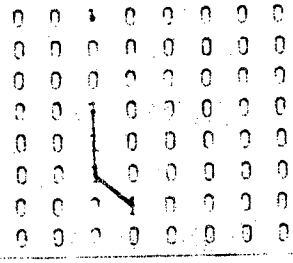
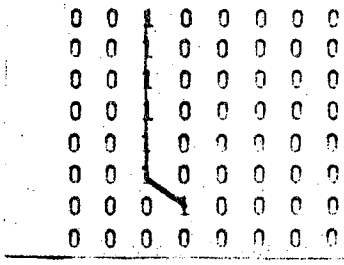
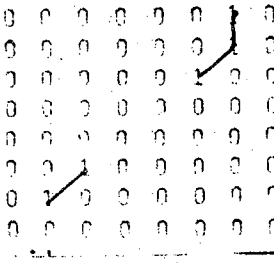
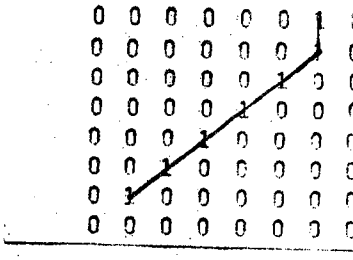
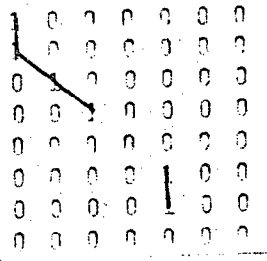
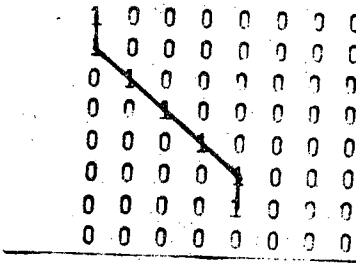
(3) two isolated points at (0,7) and (6,6)

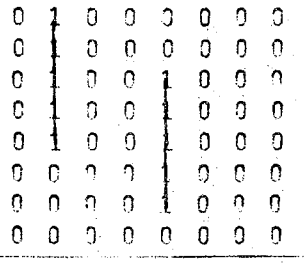
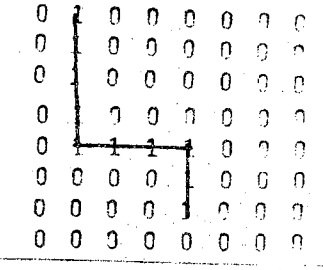
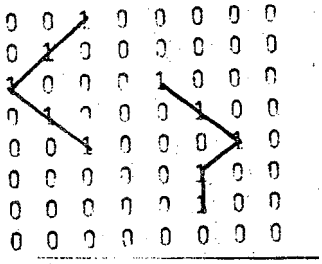
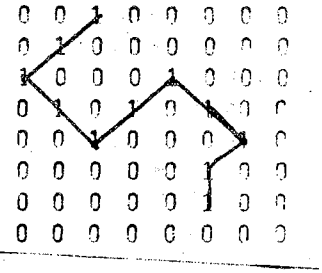
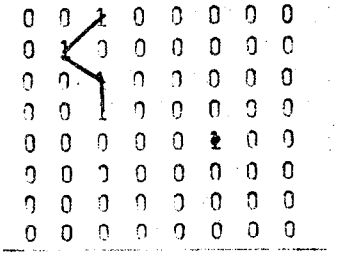
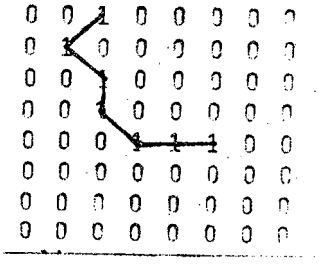
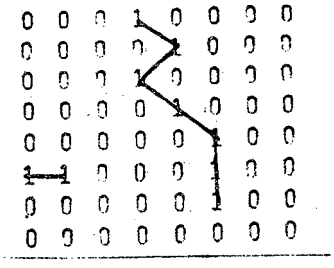
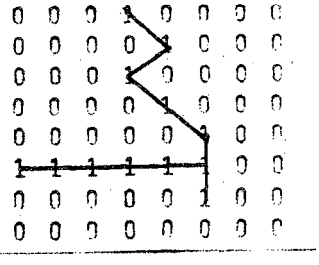
After running the program "draw-line" on this image we get the array "line" as follows :

i	j	d	i	j	d	l
0	1	-135	6	4	-135	9.4853
0	1	-135	1	5	45	10.8995
0	1	-135	6	1	-135	8.4853
0	7	0	0	7	0	0
3	6	-90	4	6	-90	1
6	6	0	6	6	0	0

The results show three lines starting from (0,1) at an angle of -135 degrees and ending at (6,4), (1,5), (6,1) with angles -135, 45 and -135 degrees respectively with respective line lengths 9.485, 10.899, 8.485 pixel units. It also shows another line starting at (3,6) and ending at (4,6) both with an angle of -90 deg. This line is of length 1. The program finds the isolated points at (0,7) and (6,6). Note that their lengths and directions are all 0.

Illustration of Some Rules

INPUT IMAGE	RULE	OUTPUT IMAGE
	RULE 5	
	RULE 6	
	RULE 11	
	RULE 16	

INPUT IMAGE	RULE	OUTPUT IMAGE
	RULE 26	
	RULE 31	
	RULE 33	
	RULE 34	

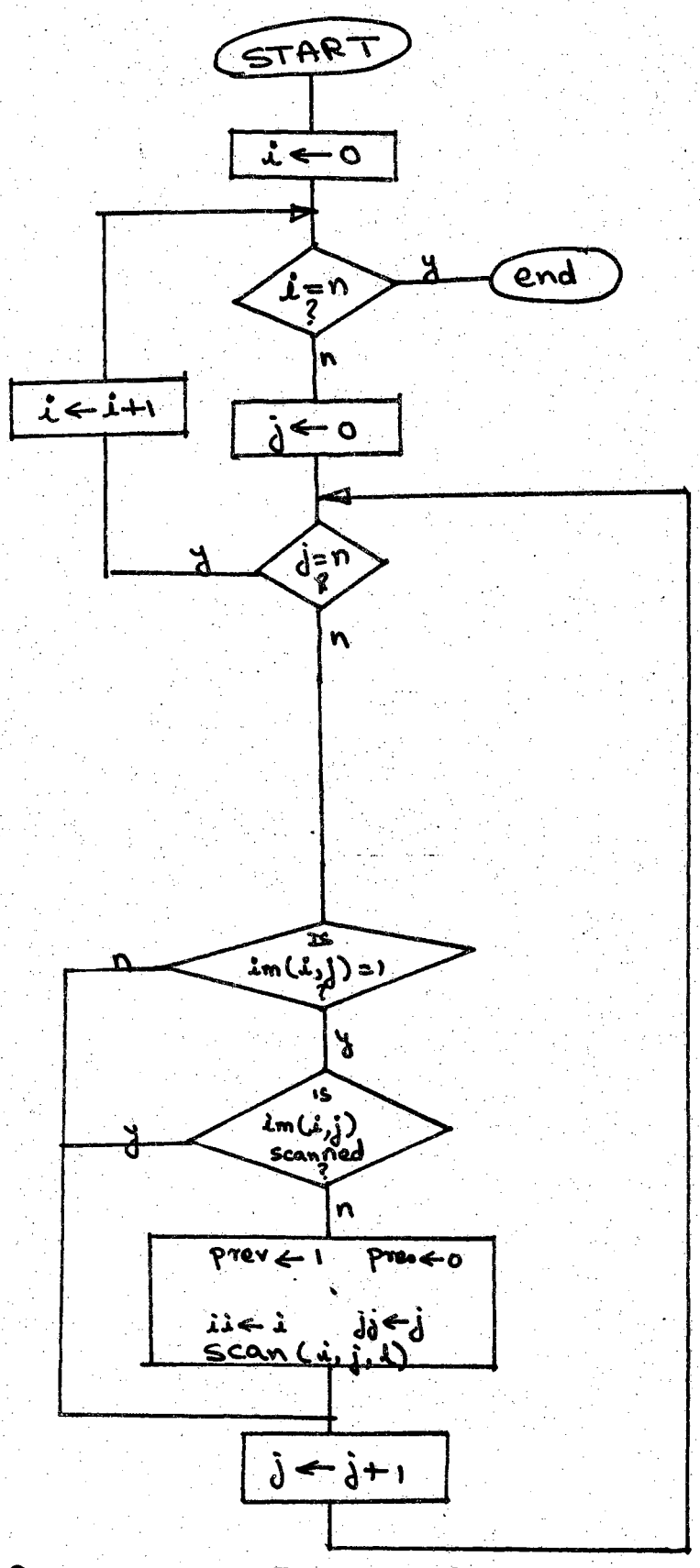


FIGURE 1.2 FLOWCHART FOR MAIN PROGRAM "DRAW-LINE" THAT CREATES THE DATABASE. THE PROGRAM IMPLEMENTS THE SUBROUTINE SCAN(i,j,d) SHOWN IN FIG. 1.3.

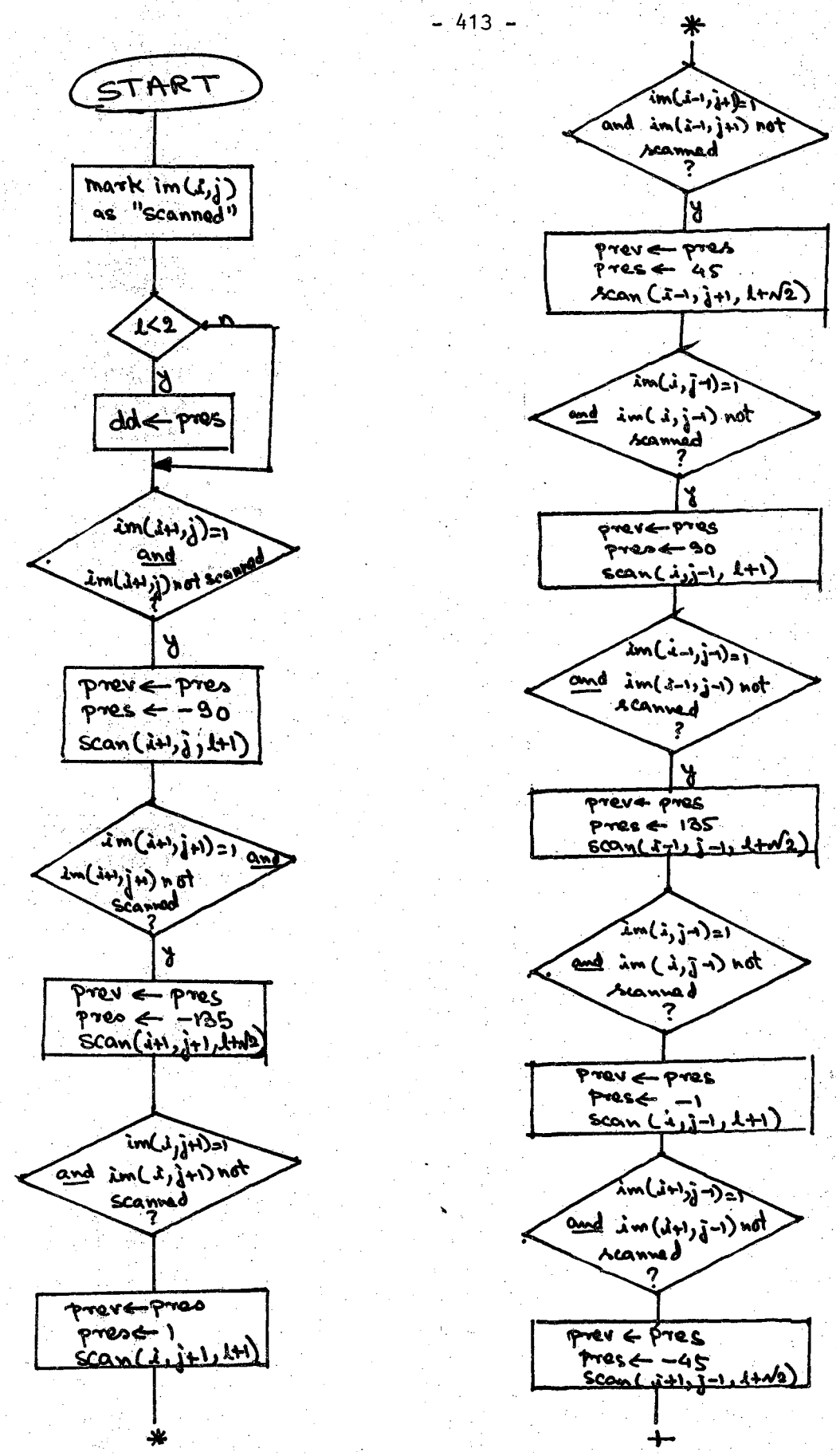


FIGURE 1.3. FLOWCHART FOR SUBROUTINE "SCAN(i, j, l)" WHICH FINDS THE LINE OR POINT STARTING FROM LOCATION (i, j) AND RETURNS THE ARRAY "LINE" WITH THE LENGTH OF LINE "l"

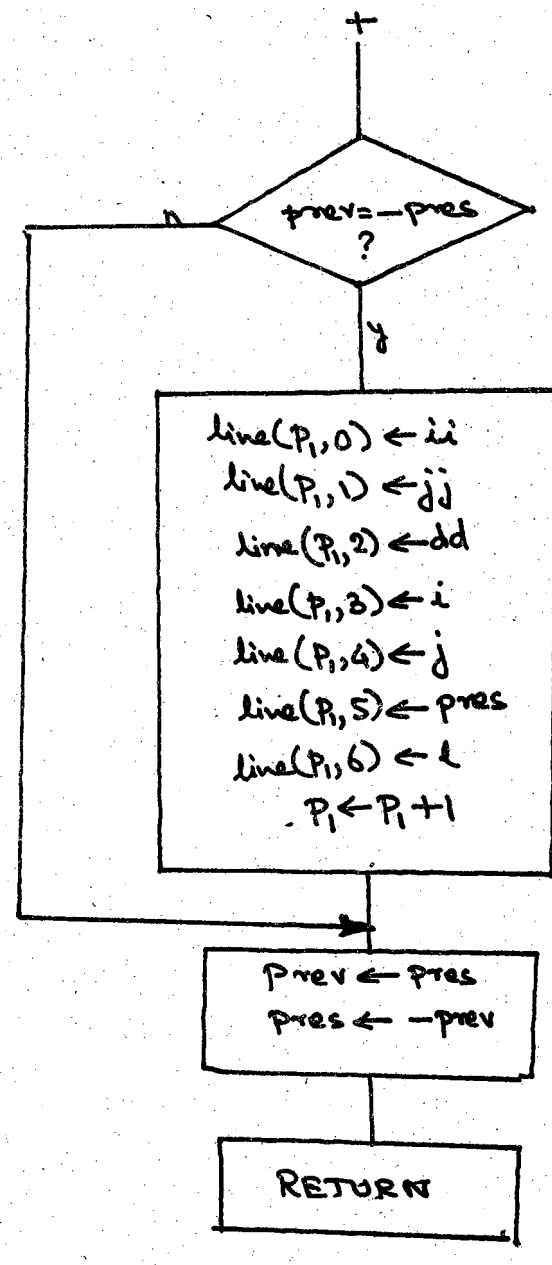


FIGURE 1.3 CONTINUED.

II. RULES FOR LINE/POINT JOINING

As mentioned before, there are two basic approaches to detect boundaries between texture images :

(1) using distinguishing features,

(2) using likelihood rules after dividing the image into a series of strips.

Both these methods give a set of lines and points along the true boundaries and a set of spurious lines and points. The rules are aimed at joining the discontinuous lines and points along the true edge and removing the spurious lines and points. A particular characteristic of the second method is that since it divides the image into strips we get a line across each strip separating one texture from another within the strip. The strips are usually chosen to be horizontal, vertical and diagonal. This results in a series of lines oriented along the eight directions mentioned before. Many of these lines are collinear to each other or are long straight lines. The rules take into account this fact and makes an effort at joining the lines along the true boundaries and eliminating the spurious ones. The rules are stated in the following sections and their working illustrated alongwith.

3.1 Rules Joining a Line and a Point in the Same Direction as the Line

Rule-5

IF there is a point P and

the distance of P from the end of a line L is short and

P is in the same direction as the end direction of L

THEN join L to P by extending end of L to P

Rule -6

IF there is a point P and

the distance of P from the beginning of line L is short and

P is in the same direction as the beginning direction of L

THEN join L to P by extending beginning of L to P.

Rule -7

IF there is a point P and

the distance of P from the end of L is medium and

P is in the same direction as end direction of L and

length of L is medium and

variance* of L is small

THEN join L to P by extending end of L to P

Rule -8

IF there is a point P and

the distance of P from the beginning of L is medium and

P is in the same direction as beginning direction of L and

length of L is medium and

* variance of a line is taken as the absolute value of the difference between the true distance of the line and the shortest distance between its starting and ending coordinates.

variance of L is small

THEN join L to P by extending beginning of L to P

Rule -9

IF there is a point P and

the distance of P from the end of L is medium and

P is in the same direction as end direction of L and

length of L is long

THEN join L to P by extending end of L to P

Rule -10

IF there is a point P and

the distance of P from the beginning of L is medium and

P is in the same direction as beginning direction of L and

length of L is long

THEN join L to P by extending beginning of L to P

3.2 Rules Joining Two Lines L1 and L2 with End Direction of L1 Same as Start Direction of L2 and End of L1 Collinear with Beginning of L2.

Rule -11

IF there are lines L1 and L2 and

distance between L1 and L2 is small and

end direction of L1 is same as Beginning direction of L2 and

starting point of L2 is in same direction as ending direction of L1

THEN join L1 and L2 by extending end of L1 to beginning of L2.

Rule –12

IF there are lines L1 and L2 and

distance between L1 and L2 is medium and

end direction of L1 is same as Beginning direction of L2 and

starting point of L2 is in same direction as ending direction of L1 and

length of L1 is medium or long and

variance of L1 and L2 are small

THEN join L1 and L2 by extending end of L1 to beginning of L2.

Rule –13

IF there are lines L1 and L2 and

distance between L1 and L2 is medium and

end direction of L1 is same as Beginning direction of L2 and

starting point of L2 is in same direction as ending direction of L1 and

length of L2 is medium or long and

variance of L1 and L2 are small

THEN join L1 and L2 by extending beginning of L2 to end of L1.

Rule –14

IF there are lines L1 and L2 and

distance between L1 and L2 is medium and

end direction of L1 is same as Beginning direction of L2 and

starting point of L2 is in same direction as ending direction of L1 and

lengths of L1 and L2 are medium or long

THEN join L1 and L2 by extending end of L1 to beginning of L2.

Rule – 15

IF there are lines L1 and L2 and

distance between L1 and L2 is long and

end direction of L1 is same as Beginning direction of L2 and

starting point of L2 is in same direction as ending direction of L1 and

lengths of L1 and L2 are long

THEN join L1 and L2 by extending end of L1 to beginning of L2.

3.3 Rules Joining Two Lines L1 and L2 with Mutual Directions Arbitrary but Beginning of L2 in the Same Direction as the End Direction of L1

Rule – 16

IF there are lines L1 and L2 and

distance between L1 and L2 is small and

starting point of L2 is in same direction as end direction of L1

THEN join L1 and L2 by extending end of L1 to beginning of L2.

Rule – 17

IF there are lines L1 and L2 and

distance between L1 and L2 is medium and

starting point of L2 is in same direction as end direction of L1 and

length of L1 is medium and

variance of L1 and L2 are small

THEN join L1 and L2 by extending end of L1 to beginning of L2.

Rule-18

IF there are lines L1 and L2 and

distance between L1 and L2 is medium and

starting point of L2 is in same direction as end direction of L1 and

lengths of L1 and L2 are medium or long

THEN join L1 and L2 by extending end of L1 to beginning of L2.

3.4 Rules Joining Two Lines with End Direction of L1 same as Starting Direction of L2 and Starting Point of L2 at a Mutual Distance d1 and Lateral distance d2 from L1.

The mutual distance (d1) and lateral distance (d2) between lines L1 and L2 are defined in figure 2.1.

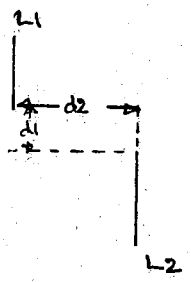


FIGURE-2.1 Figure shows mutual distance d1 and lateral distance d2 between L1 and L2.

Rule –19

IF there are lines L1 and L2 and

start direction of L2 same as end direction of L1 and

mutual distance d1 is small and

lateral distance d2 is small

THEN join the end of L1 to the beginning of L2.

Rule –20

IF there are lines L1 and L2 and

start direction of L2 same as end direction of L1 and

mutual distance d1 is medium and

lateral distance d2 is small and

length of L1 is medium or long and

length of L2 is small and

variance of L1 and L2 is small

THEN join the end of L1 to the beginning of L2.

Rule -21

IF there are lines L1 and L2 and

start direction of L2 same as end direction of L1 and

mutual distance d1 is medium and

lateral distance d2 is small and

length of L1 is small and

length of L2 is medium or long and

variance of L1 and L2 is small

THEN join the end of L1 to the beginning of L2.

Rule -22

IF there are lines L1 and L2 and

start direction of L2 same as end direction of L1 and

mutual distance d1 is medium and

lateral distance d2 is small and

lengths of L1 and L2 are medium or long

THEN join the end of L1 to the beginning of L2.

Rule-23

IF there are lines L1 and L2 and

start direction of L2 same as end direction of L1 and

mutual distance d1 is small and

lateral distance d2 is medium and

lengths of L1 and L2 are medium or long

THEN join the end of L1 to the beginning of L2.

Rule-24

IF there are lines L1 and L2 and

start direction of L2 same as end direction of L1 and

mutual distance d1 is long and

lateral distance d2 is medium and

lengths of L1 and L2 are long

THEN join the end of L1 to the beginning of L2.

Rule-25

IF there are lines L1 and L2 and

start direction of L2 same as end direction of L1 and

mutual distance d1 is long and

lateral distance d2 is small and

lengths of L1 and L2 are long or very long

THEN join the end of L1 to the beginning of L2.

3.5 Rules Joining Two Straight Lines L1 and L2 with Direction of L1 Same as Direction of L2 and Starting Point of L2 Adjacent to L1

An example of such lines is shown in figure 2.2. $\overline{BE}$ is the shortest distance between the lines $\overline{AB}$ and $\overline{CD}$. $\overline{CE}$ is the portion of the line $\overline{CD}$ that should be deleted after joining $\overline{BE}$.

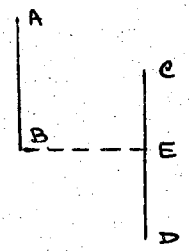


FIGURE-2.2Figure shows lines AB and CD where CE is deleted and BE is joined.

Rule –26

IF there lines L1 and L2 and

L1 and L2 are straight and

L1 and L2 are in the same direction and

start L2 is after start of L1 (i.e. A < C for L1 and L2 vertical) and

start of L2 is before end of L1 (i.e. C < B for L1 and L2 vertical) and

end of L2 is after end of L1 (i.e. B < D for L1 and L2 vertical) and

$\overline{CE}$ is small and

$\overline{BE}$ is small

THEN join $\overline{BE}$ and delete $\overline{CE}$.

Rule –27

IF there lines L1 and L2 and

L1 and L2 are straight and

L1 and L2 are in the same direction and

start L2 is after start of L1 (i.e. A < C for L1 and L2 vertical) and

start of L2 is before end of L1 (i.e. C < B for L1 and L2 vertical) and

end of L2 is after end of L1 (i.e. B < D for L1 and L2 vertical) and

$\overline{CE}$ is medium and

$\overline{BE}$ is small and

length of L2 is long

THEN join $\overline{BE}$ and delete $\overline{CE}$.

Rule –28

IF there lines L1 and L2 and

L1 and L2 are straight and

L1 and L2 are in the same direction and

start L2 is after start of L1 (i.e. A < C for L1 and L2 vertical) and

start of L2 is before end of L1 (i.e. C < B for L1 and L2 vertical) and

end of L2 is after end of L1 (i.e. B < D for L1 and L2 vertical) and

$\overline{CE}$ is small and

$\overline{BE}$ is medium and

length of L1 is medium or long ,

THEN join $\overline{BE}$ and delete $\overline{CE}$.

Rule –29

IF there lines L1 and L2 and

L1 and L2 are straight and

L1 and L2 are in the same direction and

start L2 is after start of L1 (i.e. A < C for L1 and L2 vertical) and

start of L2 is before end of L1 (i.e. C < B for L1 and L2 vertical) and

end of L2 is after end of L1 (i.e. B < D for L1 and L2 vertical) and

$\overline{CE}$ is small and

$\overline{BE}$ is medium and

length of L2 is medium or long

THEN join $\overline{BE}$ and delete $\overline{CE}$.

Rule –30

IF there lines L1 and L2 and

L1 and L2 are straight and

L1 and L2 are in the same direction and

start L2 is after start of L1 (i.e. A < C for L1 and L2 vertical) and

start of L2 is before end of L1 (i.e. C < B for L1 and L2 vertical) and

end of L2 is after end of L1 (i.e. B < D for L1 and L2 vertical) and

$\overline{CE}$ is medium and

$\overline{BE}$ is medium and

lengths of L1 and L2 are long

THEN join $\overline{BE}$ and delete $\overline{CE}$.

3.6 Rules Joining Lines L1 and L2 with Mutual Directions Arbitrary

Rule –31

IF there are lines L1 and L2 and

distance between end of L1 and beginning of L2 is small

THEN join end of L1 to the beginning of L2.

Rule –32

IF there are lines L1 and L2 and

distance between end of L1 and beginning of L2 is medium and

lengths of L1 and L2 are long

THEN join end of L1 to the beginning of L2.

3.7 Rules Joining Line L and Point P in Arbitrary Locations With Respect With to Each Other.

Rule –33

IF there is a line L and a point P and

distance between them is small

THEN join the line L to the point P

3.8 Rules Joining a Line L1 to a Point on Another Line L2 by Shortest Path

Rule –34

IF there are lines L1 and L2 and

shortest distance between end of L1 and line L2 is small and

variance of L1 is small

THEN join L1 and L2 by forward extension of L1.

Rule -35

IF there are lines L1 and L2 and

shortest distance between beginning of L1 and line L2 is small and

variance of L1 is small

THEN join L1 and L2 by backward extension of L1.

Rule -36

IF there are lines L1 and L2 and

shortest distance between end of L1 and line L2 is small and

length of L1 is medium or long and

length of L2 is medium or long

THEN join L1 and L2 by forward extension of L1.

Rule -37

IF there are lines L1 and L2 and

shortest distance between beginning of L1 and line L2 is small and

length of L1 is medium or long and

length of L2 is medium or long

THEN join L1 and L2 by backward extension of L1.

Rule -38

IF there are lines L1 and L2 and

shortest distance between end of L1 and line L2 is medium and

lengths of L1 and L2 are long

THEN join L1 and L2 by forward extension of L1.

Rule –39

IF there are lines L1 and L2 and

shortest distance between beginning of L1 and line L2 is medium and

lengths of L1 and L2 are long

THEN join L1 and L2 by backward extension of L1.

3.9 **Delete Rules**

Rule –40

IF there is an isolated point in the image

THEN delete it

Rule –41

IF there is a short line in the image

THEN delete it

Rule –42

IF there is a line of medium length and

there are one or more long and very long lines and

there are no lines within medium distance from it

THEN delete it.

III. INFERENCE SYSTEM

The "inference system" chooses the appropriate set of rules to apply on the given image. The set of rules applicable to the image is important to choose for the following reasons :

(1) it saves computing time in executing a limited number of rules on the image than running all the possible rules,

(2) it prevents confusion and possible errors due to application of inappropriate rules to the image.

I have divided the rules into the following five sets :

(1) Rules joining lines and points :

Rules 5, 6, 7, 8, 9, 10, 33.

Call this set S(1).

(2) Rules joining two lines L1 and L2 that are collinear i.e. the beginning direction of L2 is same as the ending direction of L1 and the beginning point of L2 is in the same direction as the ending direction of L1.

Rules 11, 12, 13, 14, 15.

Call this set S(2).

(3) Rules joining two lines L and L2 with beginning direction of L2 different from ending direction of L.

Rules 16, 17, 18, 31, 32, 34, 35, 36, 37, 38, 39.

Call this set S(3).

(4) Rules joining two lines L1 and L2 with beginning direction of L2 same as ending direction of L1 and L2 not collinear with L1 and L2 not adjacent to L1.

Rules 19, 20, 21, 22, 23, 24, 25.

Call this set S(4).

(5) Rules joining two lines L1 and L2 with beginning direction of L2 same as the ending direction of L1 and L2 not collinear to L1 and L2 adjacent to L1 within medium distance.

Rules 26, 27, 28, 29, 30.

Call this set S(5).

Besides these are the delete rules.

I assign five flags Flag(1),...,Flag(5) to each of the above sets of rules respectively. The following algorithm scans through the entire set of lines and points in the image and sets the appropriate flag to "1". I use the set of rules corresponding to the flags that are "1" and do not use those for which the flag is "0". The flowchart for the algorithm is given in figure 8.3.

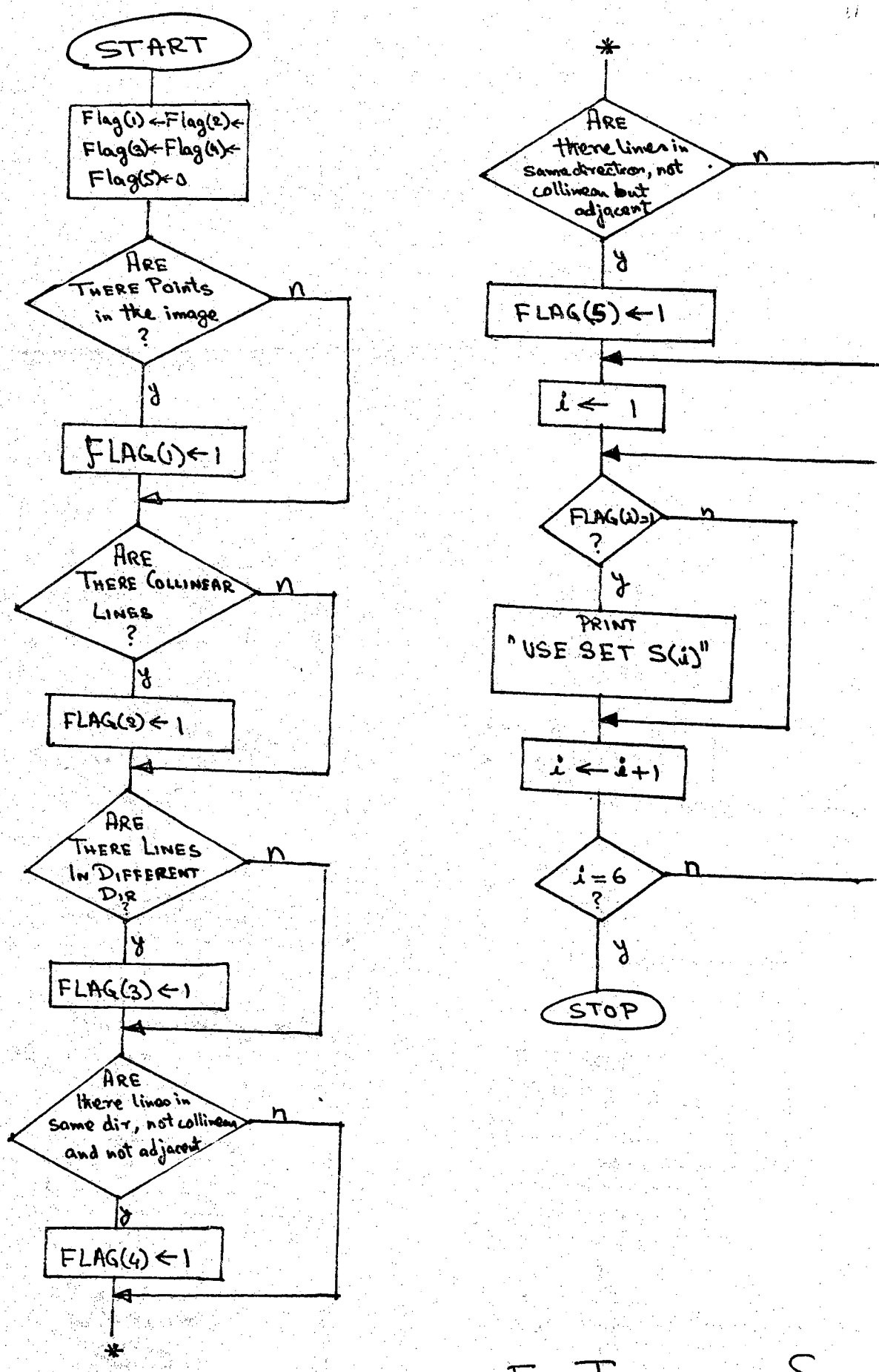


FIGURE #3. FLOWCHART FOR INFERENCE SYST

IV. EXPERIMENTAL RESULTS

The statistical likelihood method was implemented by Mr. K.B. Eom to generate the set of edges from 128 X 128 images of multiple textures. The rules discussed in section III were executed on them after selecting the appropriate rules by the inference system.

The original sets of lines and points obtained after running the edge detection algorithm on four different images are i figures 5.1, 6.1, 7.1 and 8.1. All these images show a set of lines parallel to each other. These lines are non-collinear and non-adjacent within medium distances. So the inference system chooses the set S(4) as the appropriate set of rules for these images. Thus the rules are :

S(4) : Rules 19, 20, 21, 22, 23, 24, 25 and

the delete rules for lines i.e. Rules 41, 42.

Even though a statistical approach can be taken to define short, medium and long lines and distances on the image, there is, however, o theoretical support to approach this problem. I, therefore, define distances heuristically based on the size of the image and the size of the lines in the image and *keep them fixed for all images in my experiments*. The distances are defined as follows :

(1) For distances between lines and points or between two lines the following units are chosen :

(ii) less than 4 pixel units is chosen as short distance,

(ii) between 4 to 6 pixels is chosen as medium distance,

(iii) between 7 to 10 pixels is chosen as long distance.

(2) For lengths of lines it may be seen in the images that the shortest line is 3 pixel units long and all lines are multiples of this. So the following units are chosen for lengths of lines.

(i) less than 4 pixel units is chosen as small line,

(ii) between 4 to 10 units is chosen as medium line,

(iii) between 11 to 14 units is chosen as long lines.

The sequence in which the rules are applied, their explanations and the output images are shown in table 1. It may be noted that figures 8.1,...,8.10 are for a 64 X 64 image for which the definitions of short, medium and long lines are modified correspondingly for this smaller image.

TABLE-1. Description of the sequence of rules, their explanations and the output images obtained by applying these rules.

Sequence	Rules	Explanation	Output images of Rules
1.	Rule-19	Join lines within short distances	Figs. 5.2, 6.2, 7.2, 8.2
2.	Rule-20	Join lines L1, L2 L1 medium or long L2 short mutual distance med lateral dist. small	Figs. 5.3, 6.3, 7.3, 8.3
3.	Rule-21	Join lines L1, L2 L1 short L2 medium or long mutual distance med. lateral dist. small	Figs. 5.4, 6.4, 7.4, 8.4
4.	Rule-22	Join lines L1, L2 L1,L2 med. or long mutual dist. medium lateral dist. small	Figs. 5.5, 6.5, 7.5, 8.5
5.	Rule-23	Join lines L1, L2 L1, L2 med. or long mutual dist. small lateral dist. med.	Figs. 5.6, 6.6, 7.6, 8.6
6.	Rule-41	delete small lines	Figs. 5.7, 6.7, 7.7, 8.7
7.	Rule-24	Join lines L1, L2 L1,L2 long mutual dist. long lateral dist. med.	Figs. 5.8, 6.8, 7.8, 8.8
8.	Rule-42	delete medium lines	Figs. 5.9, 6.9, 7.9, 8.9
9.	Rule-25	Join lines L1,L2 L1,L2 long or very long mutual dist. long lateral dist. small	Figs. 5.10, 6.10, 7.10, 8.10

Discussion

The results clearly indicate that these sets of rules work well for normal images of multiple textures. In figure 5.10 we see the true edge and no spurious edge. Here the rules completely pick up the true boundary without any

error. In figure 6.10 we see the true boundary and one spurious boundary which could not be deleted as according to our definitions it is a long line. In figure 7.10 we have lost a small upper part of the boundary as this region has fewer edges and they are disconnected by larger distances. This figure also shows one spurious edge which was not deleted as it is a long line by our definitions. The rules, however, pick up the major portion of the true boundary. All these results show that the suggested set of rules work well for normal images and appreciably detects the true boundary.

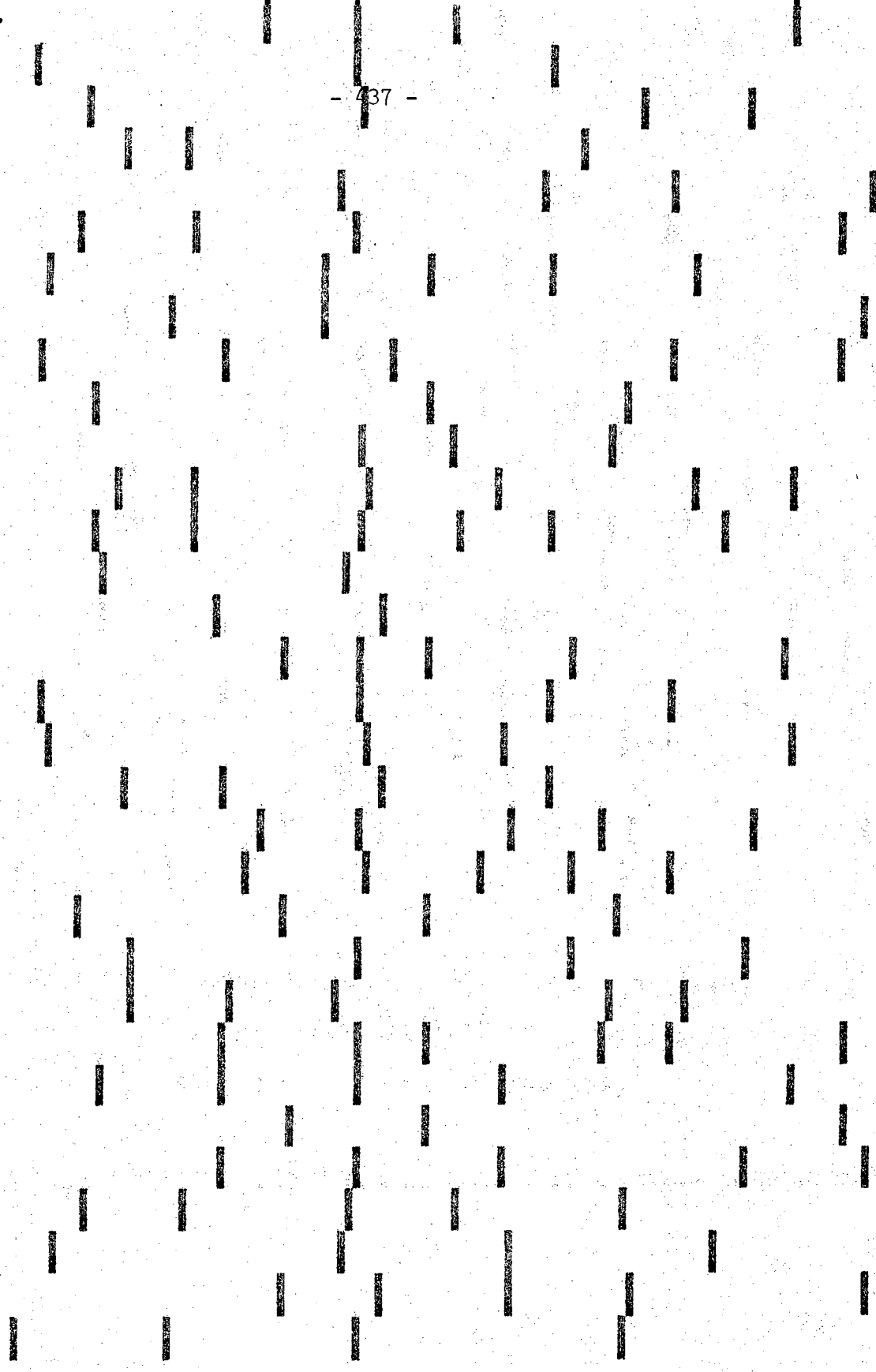


FIGURE-5.1 Figure shows extracted image consisting of a set of lines obtained by executing a boundary detection algorithm on the image.

FIGURE-5.2 Figure shows the result of running RULE 19 on image in figure 5.1.

FIGURE-5.3 Figure shows the result of running RULE 20 on image in figure 5.2.

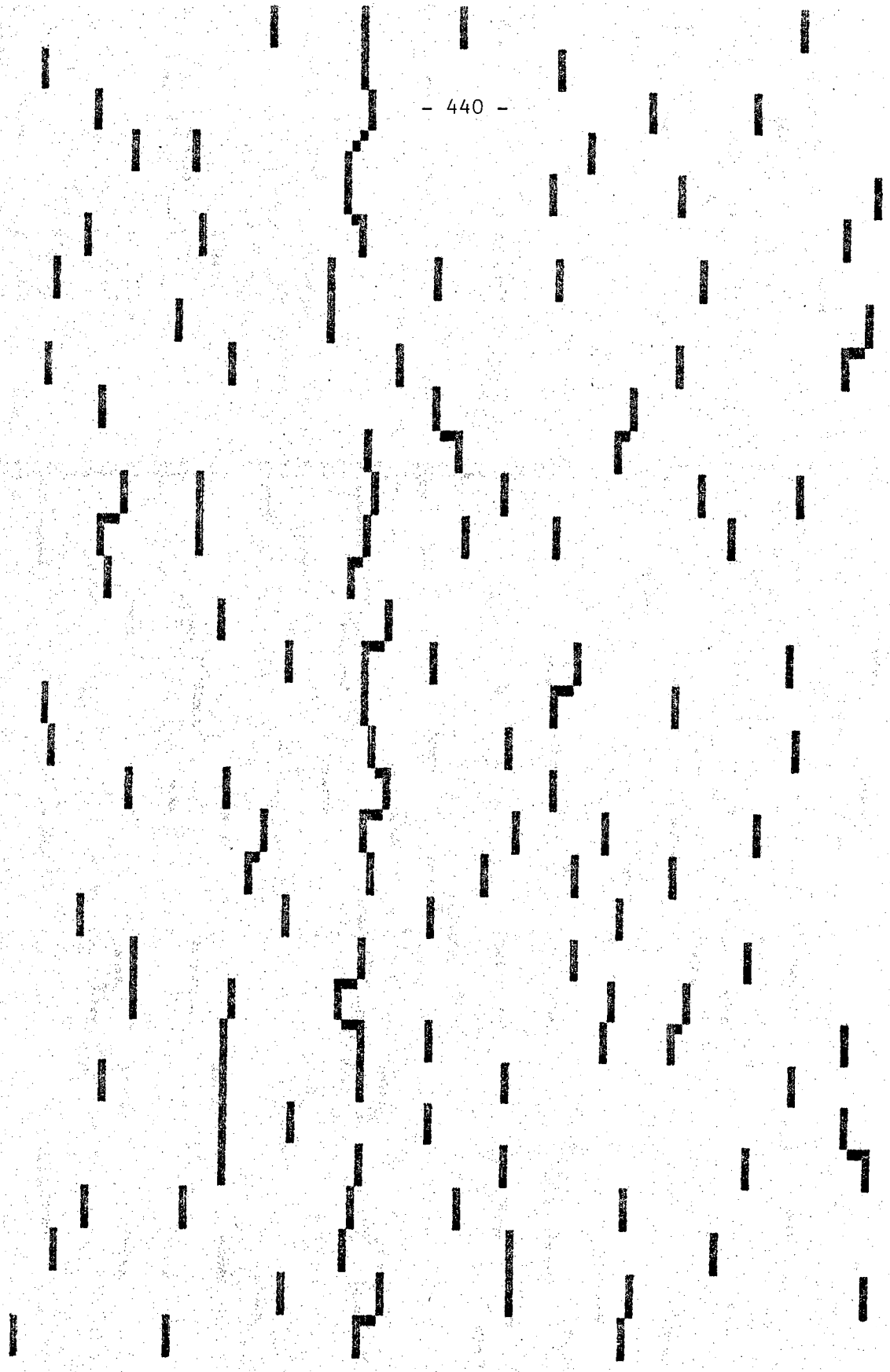


FIGURE-5.4 Figure shows the result of running RULE 21 on image in figure 5.3.

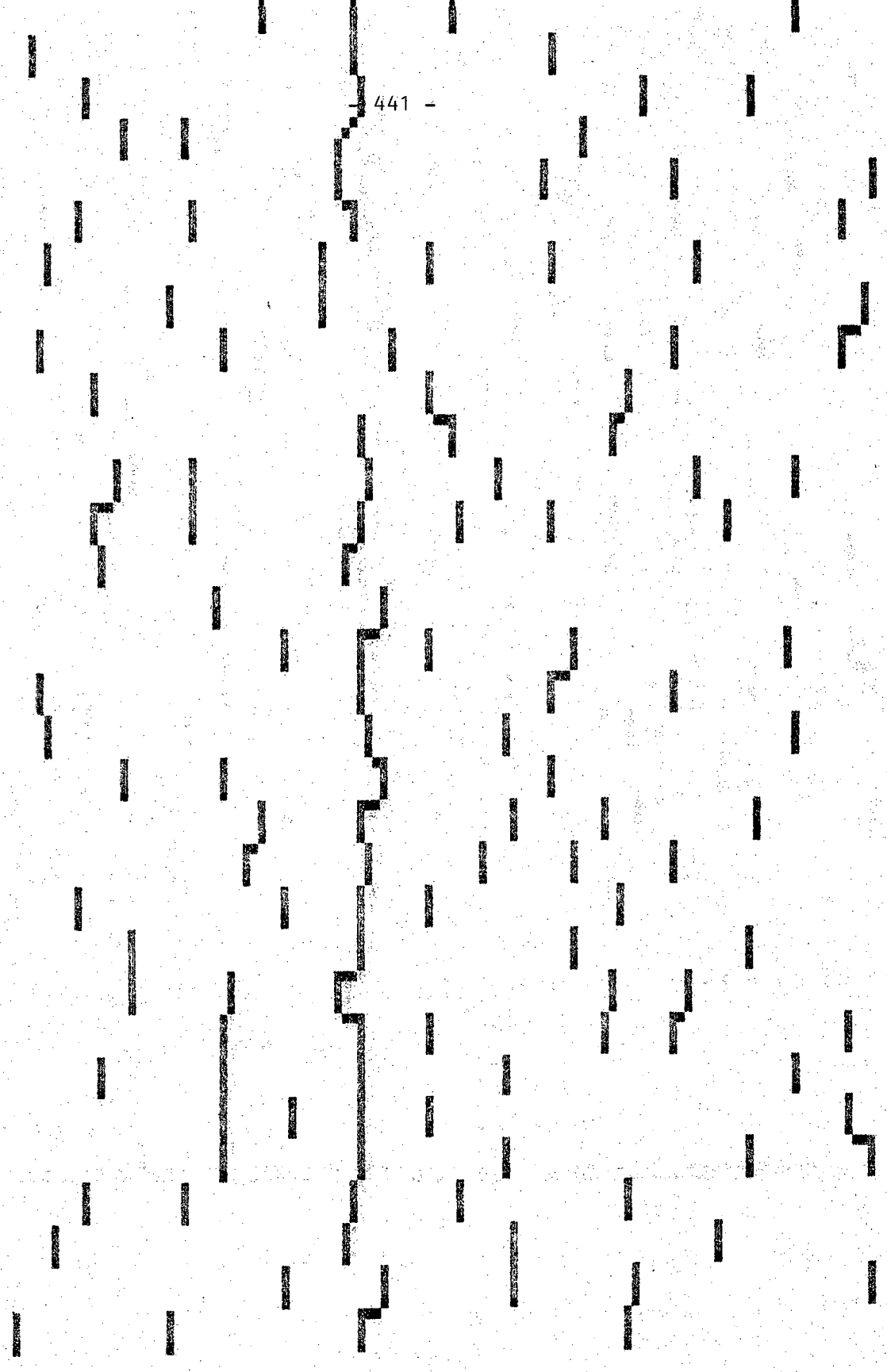


FIGURE-5.5 Figure shows the result of running RULE 22 on image in figure 5.4.

FIGURE-5.6 Figure shows the result of running RULE 23 on image in figure 5.5.

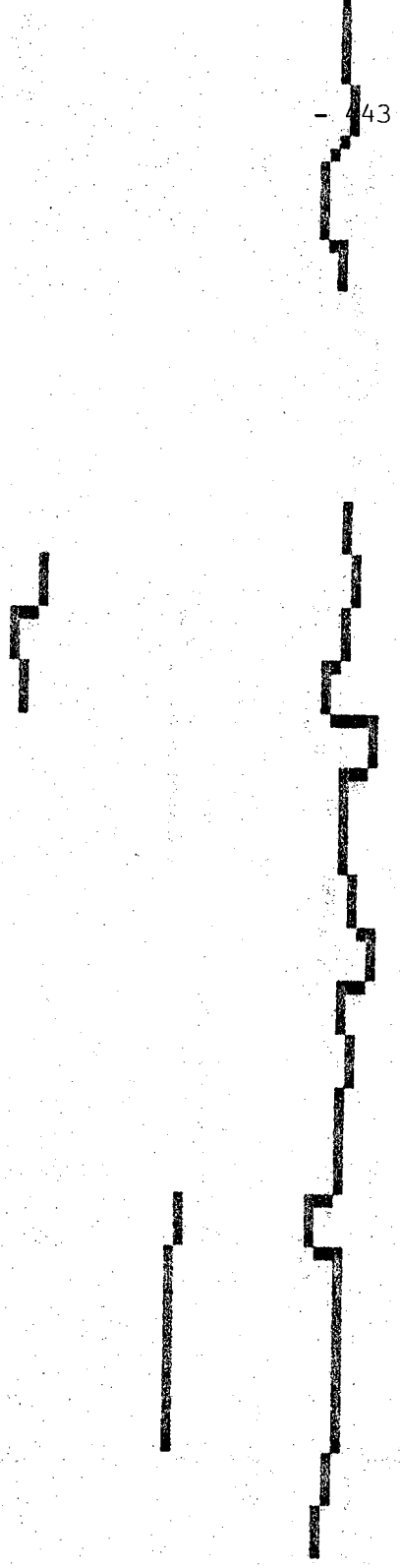


FIGURE-5.7 Figure shows the result of running RULE 41 on image in figure 5.6.

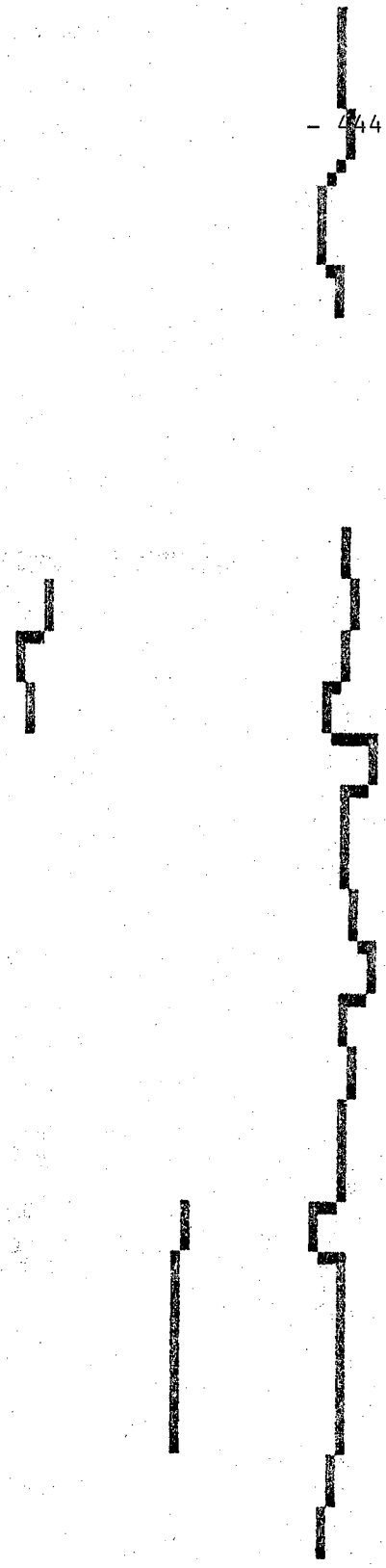


FIGURE-5.8 Figure shows the result of running RULE 24 on image in figure 5.7.

FIGURE-5.9 Figure shows the result of running RULE 42 on image in figure 5.8.

FIGURE-5.10 Figure shows the result of running RULE 25 on image in figure 5.9. This is the final output image.

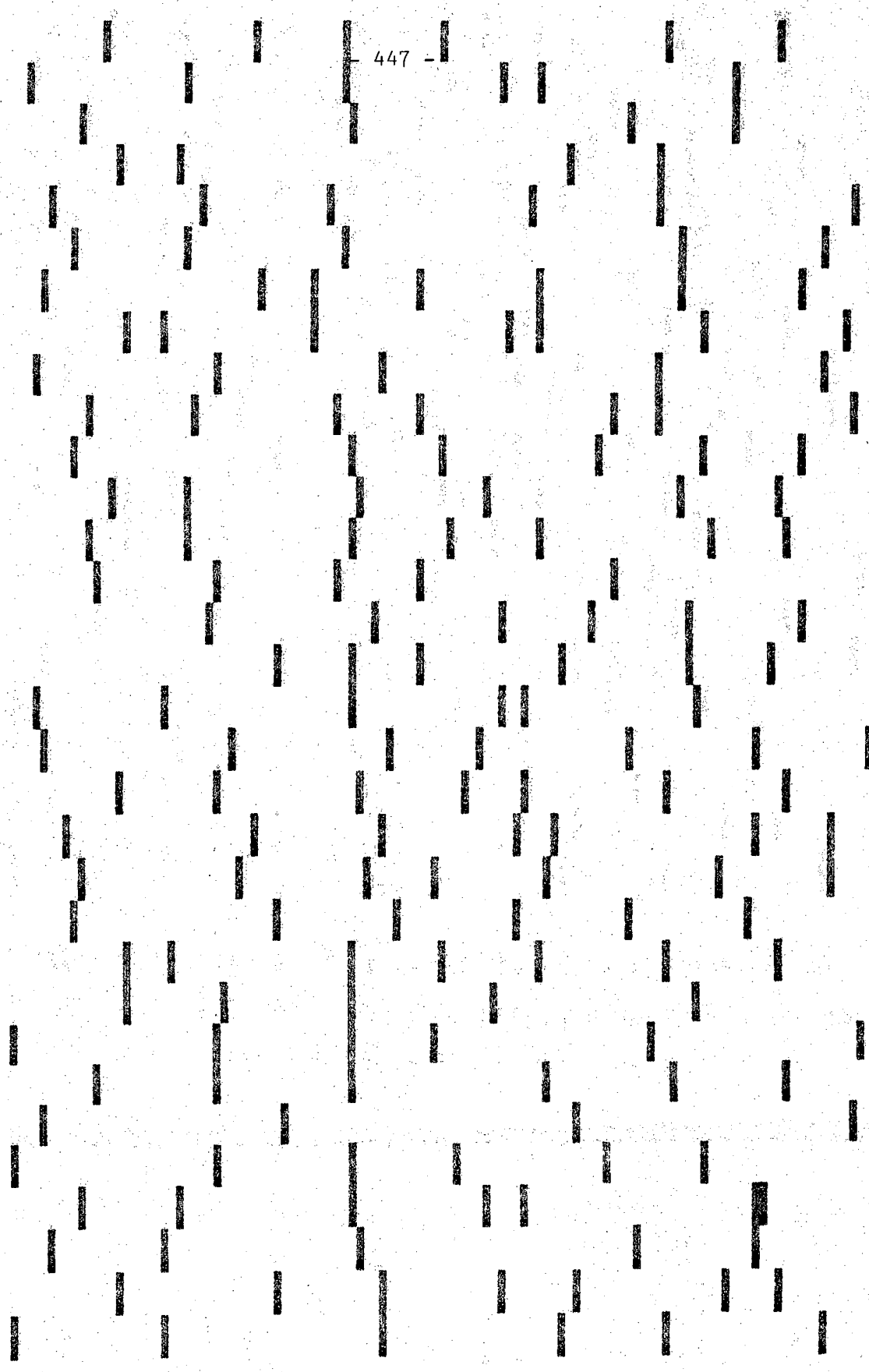


FIGURE-6.1 Figure shows extracted image consisting of a set of lines obtained by executing a boundary detection algorithm on the image.

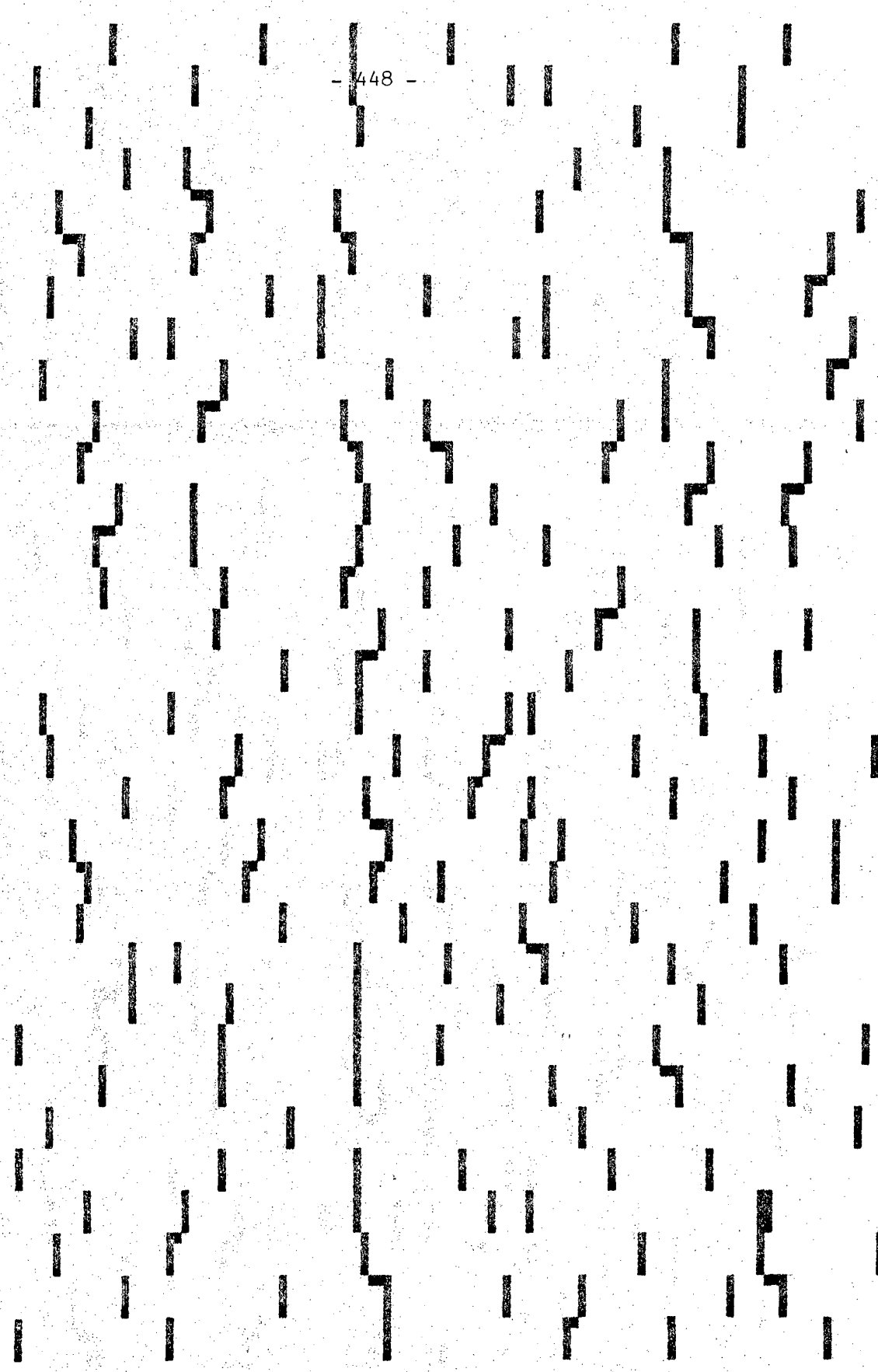


FIGURE-6.2 Figure shows the result of running RULE 19 on image in figure 6.1.

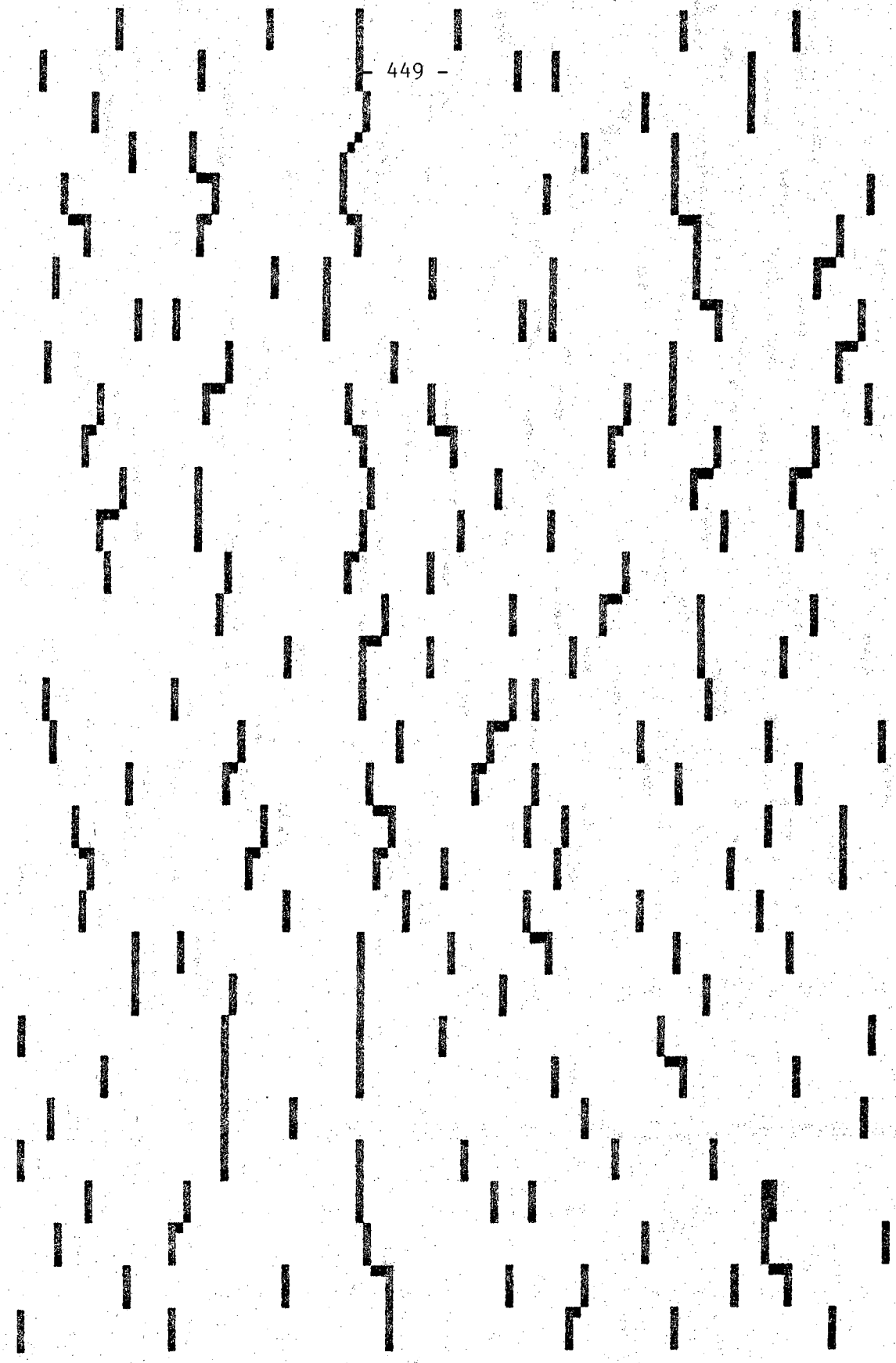


FIGURE-6.3 Figure shows the result of running RULE 20 on image in figure 6.2.

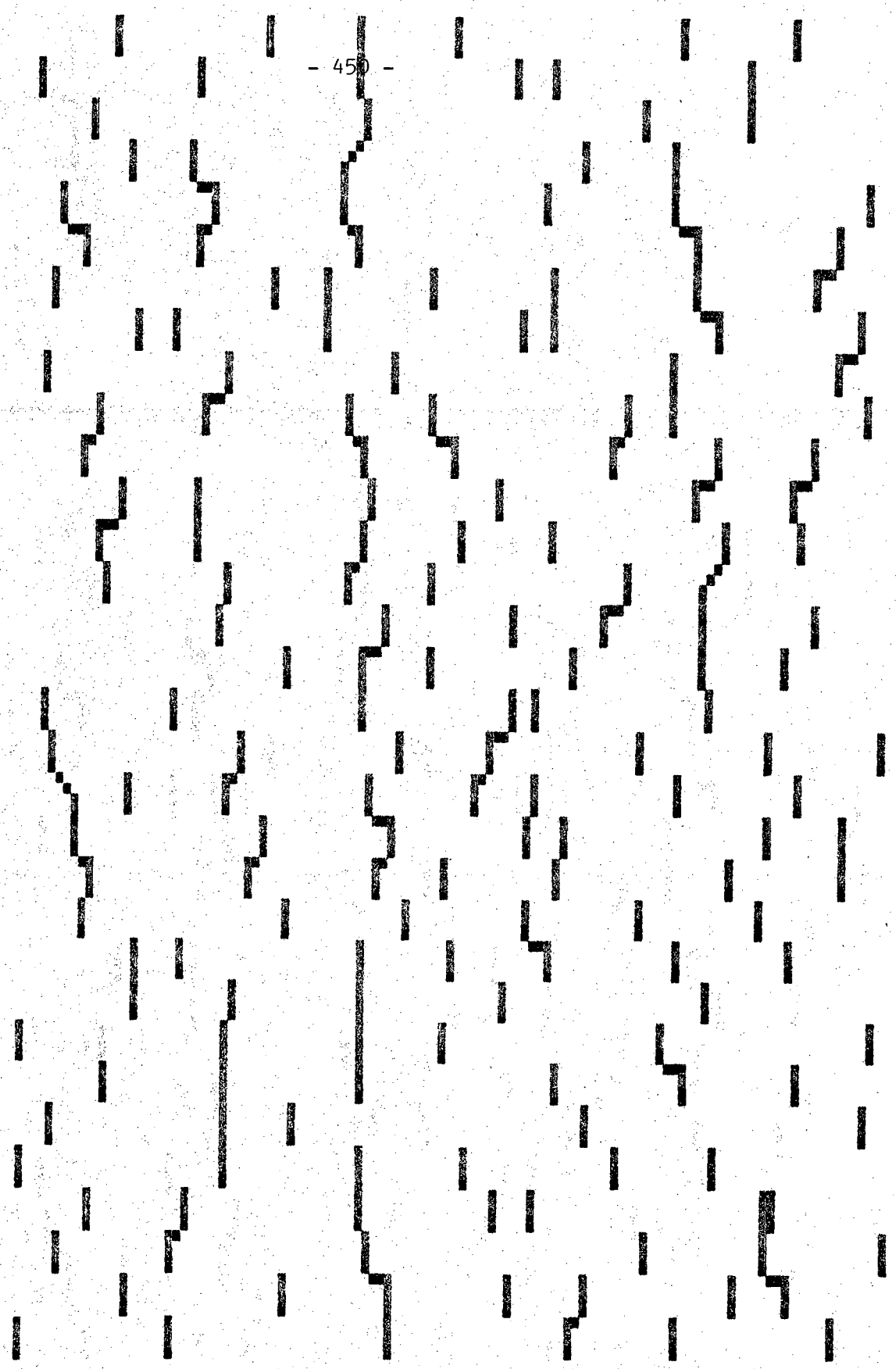


FIGURE-6.4 Figure shows the result of running RULE 21 on image in figure 6.3.

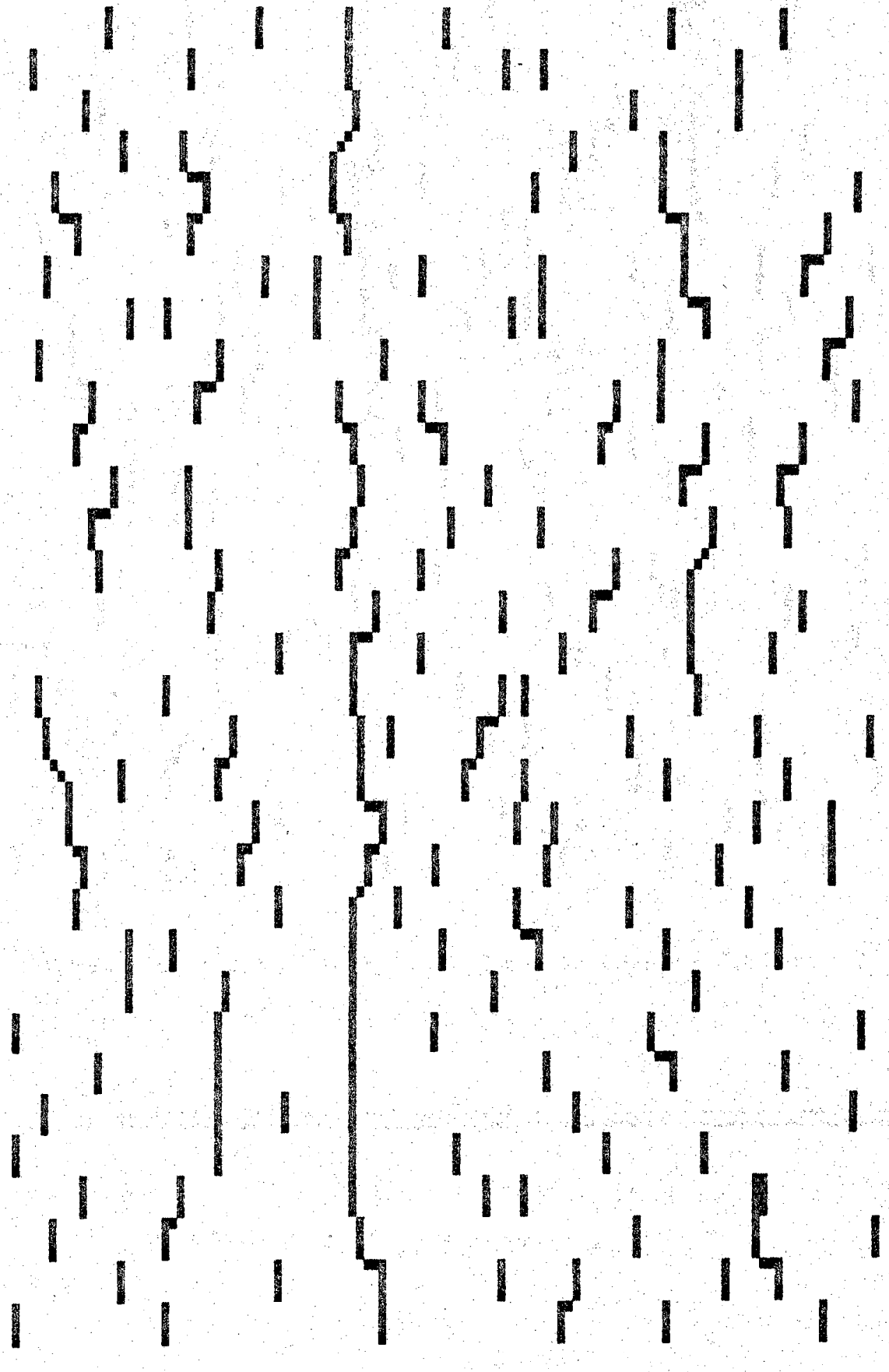


FIGURE-6.5 Figure shows the result of running RULE 22 on image in figure 6.4.

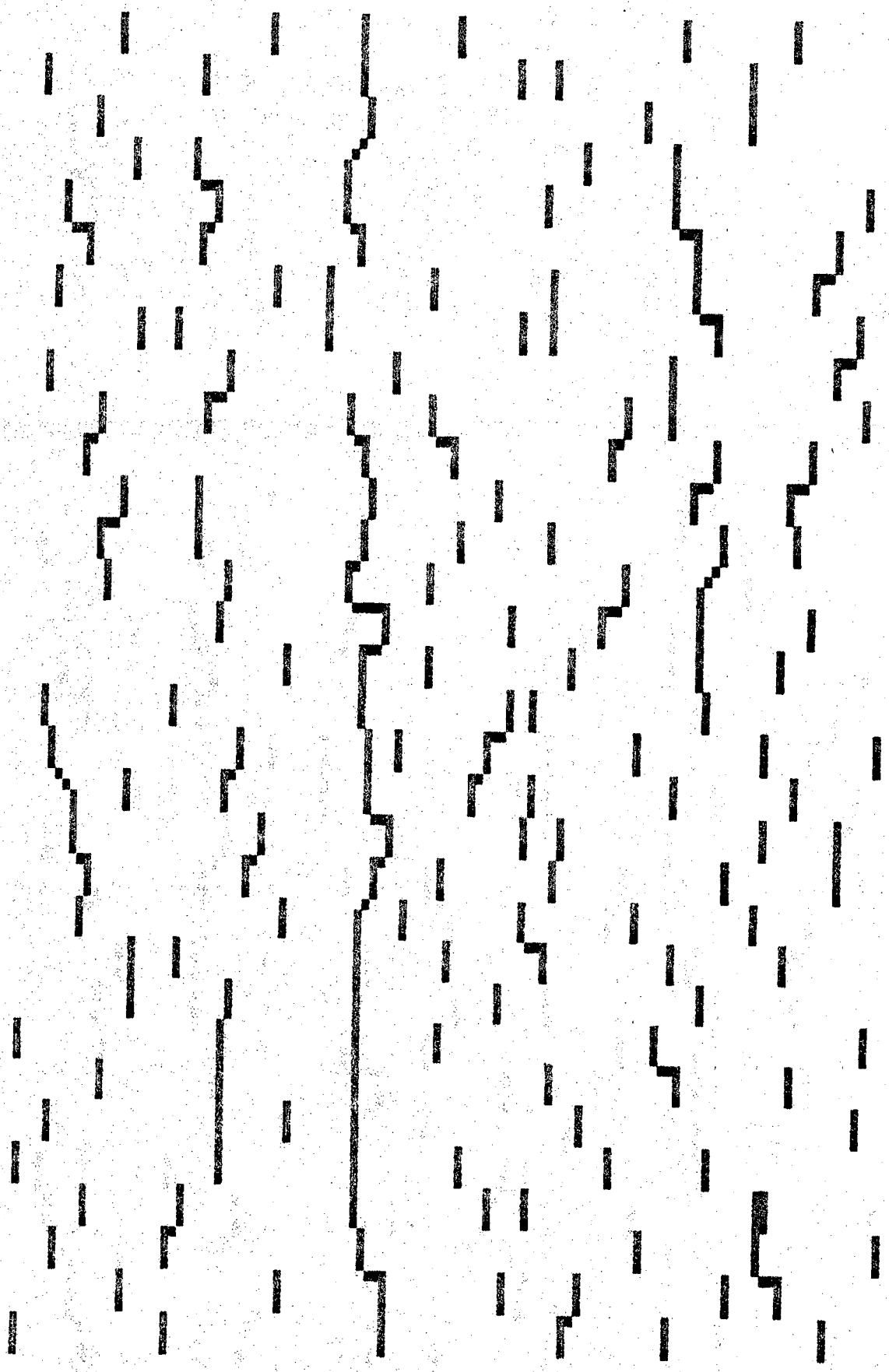


FIGURE-6.6 Figure shows the result of running RULE 23 on image in figure 6.5.

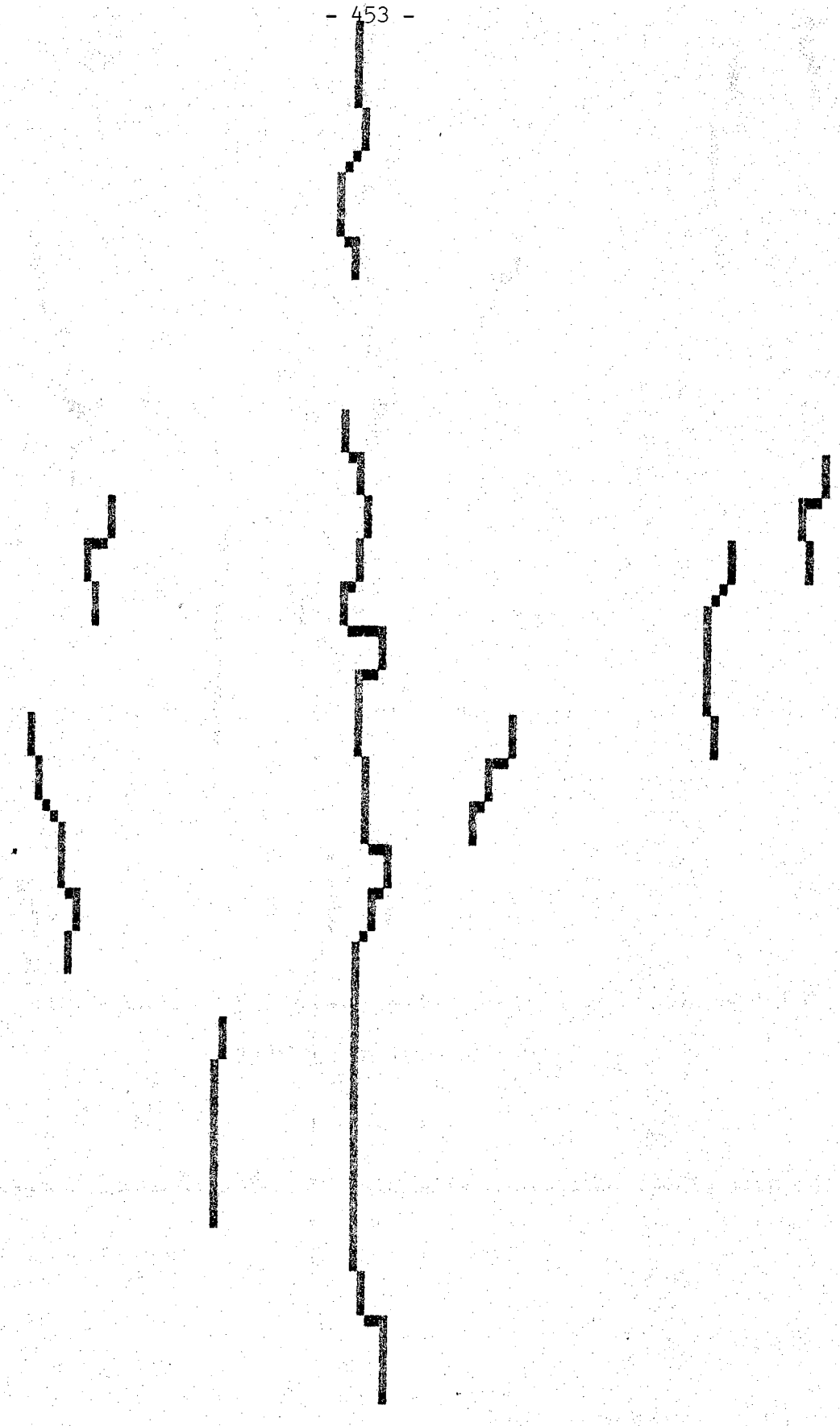


FIGURE-6.7 Figure shows the result of running RULE 41 on image in figure 6.6.

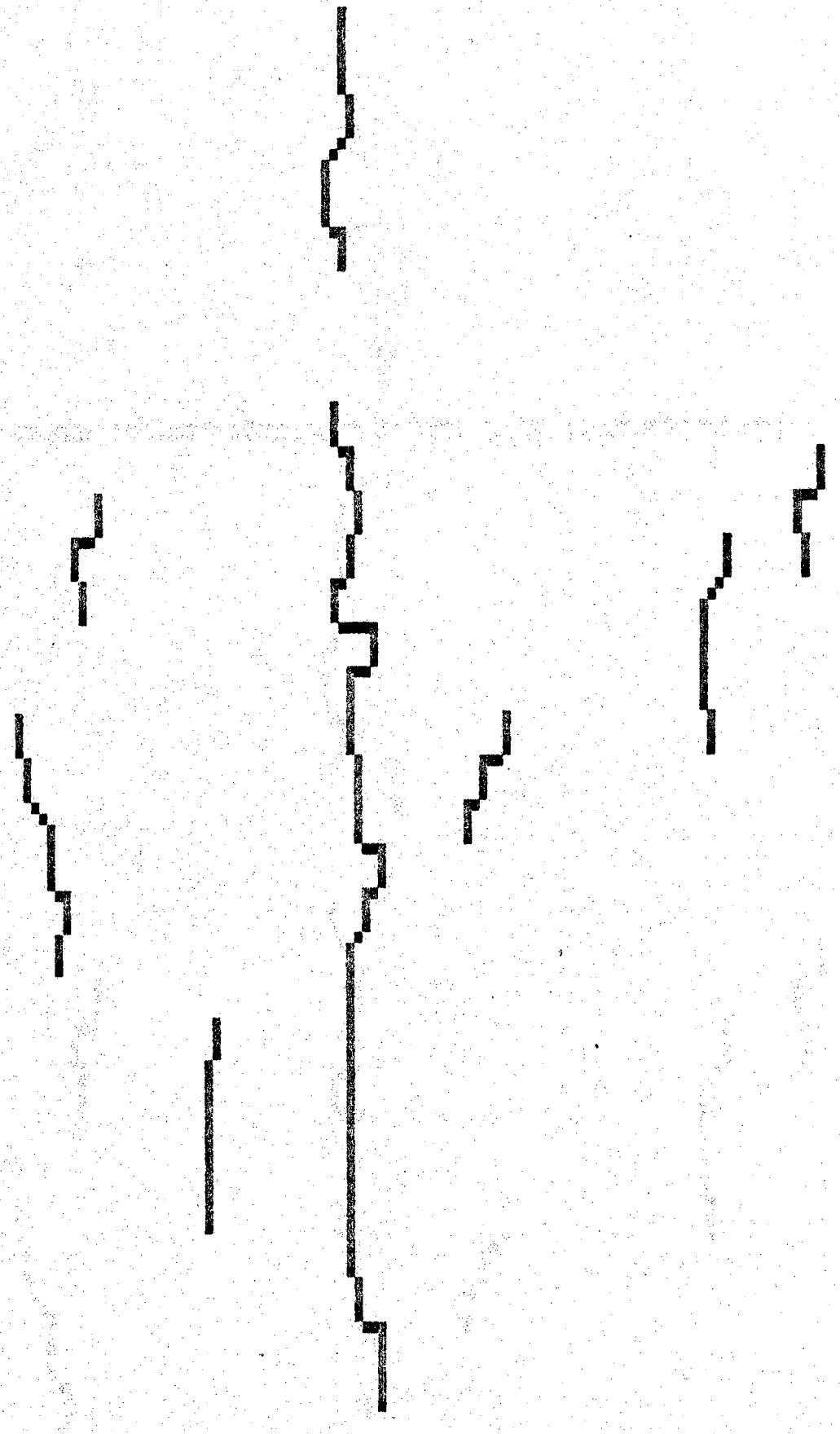


FIGURE-6.8 Figure shows the result of running RULE 24 on image in figure 6.7.

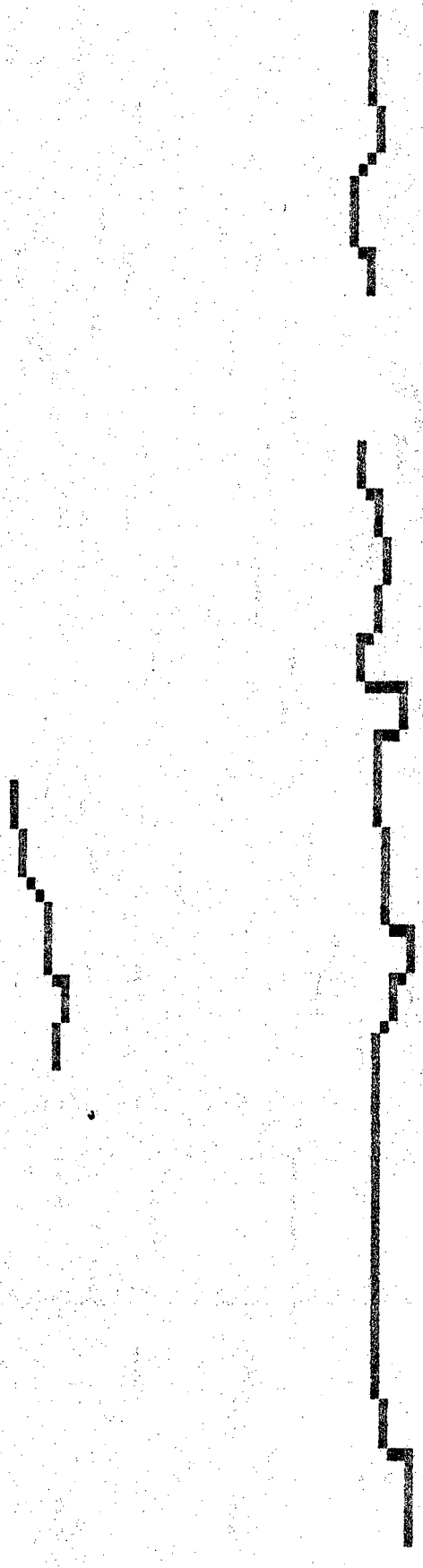


FIGURE-6.8 Figure shows the result of running RULE 42 on image in figure 6.8.

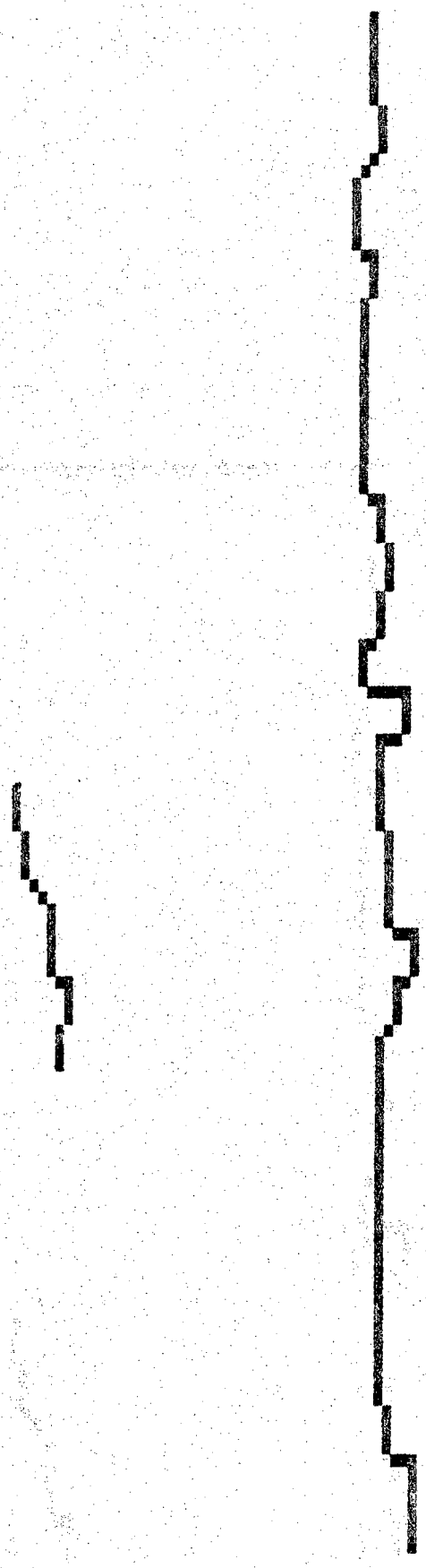


FIGURE-6.10 Figure shows the result of running RULE 25 on image in figure 6.9. This is the final output image.

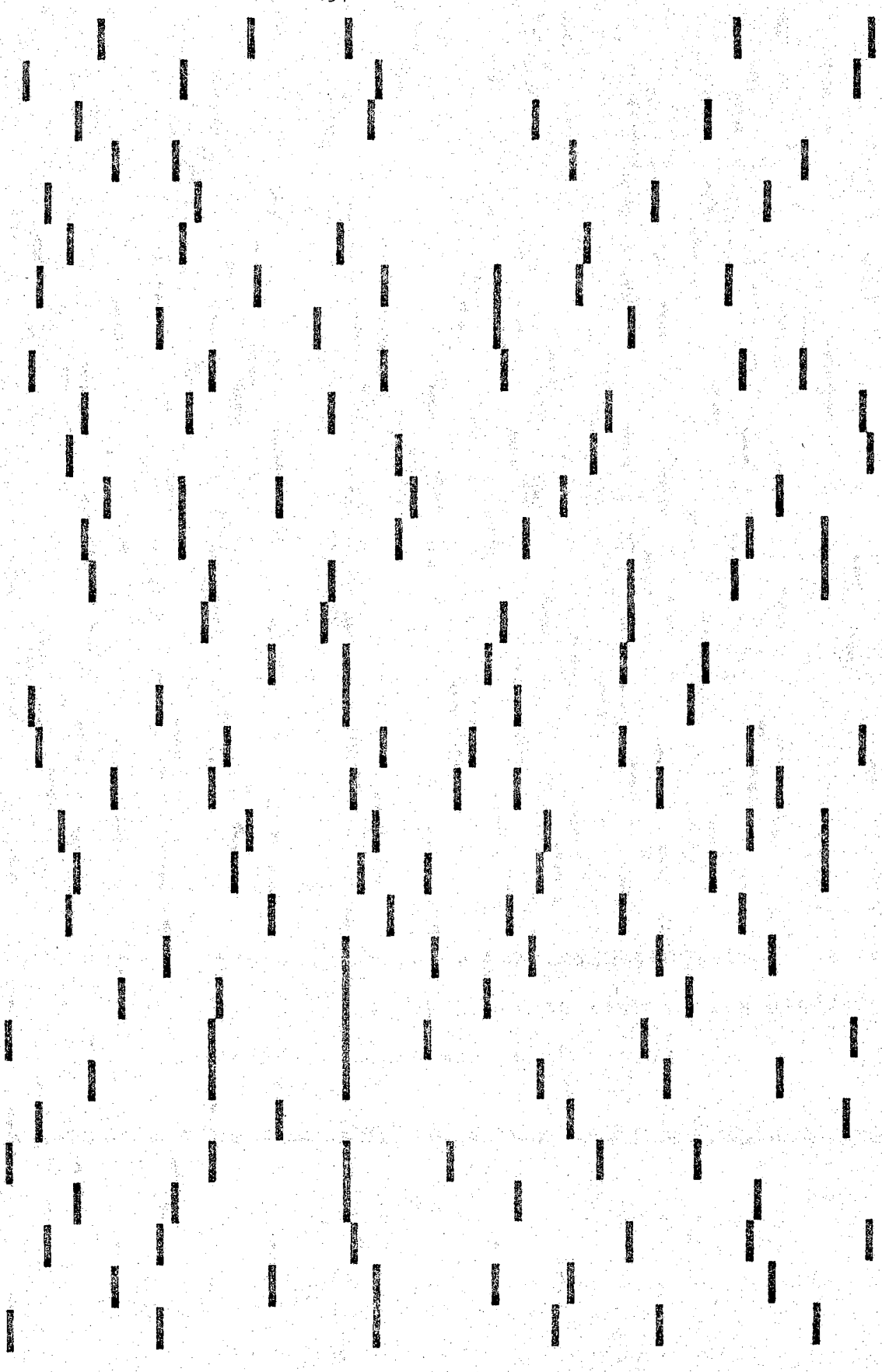


FIGURE-7.1 Figure shows extracted image consisting of a set of lines obtained by executing a boundary detection algorithm on the image.

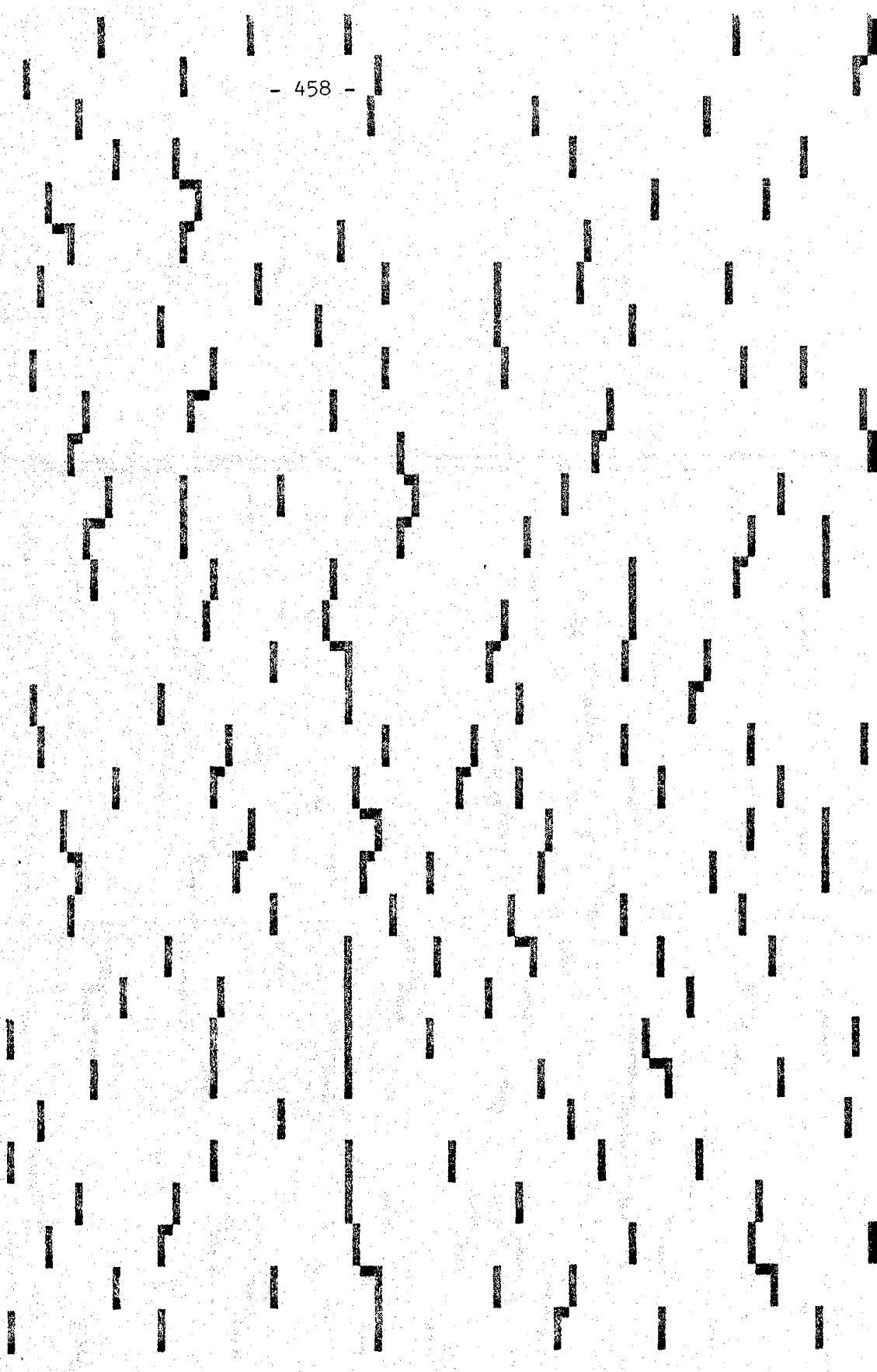


FIGURE-7.2 Figure shows the result of running RULE 19 on image in figure 7.1.

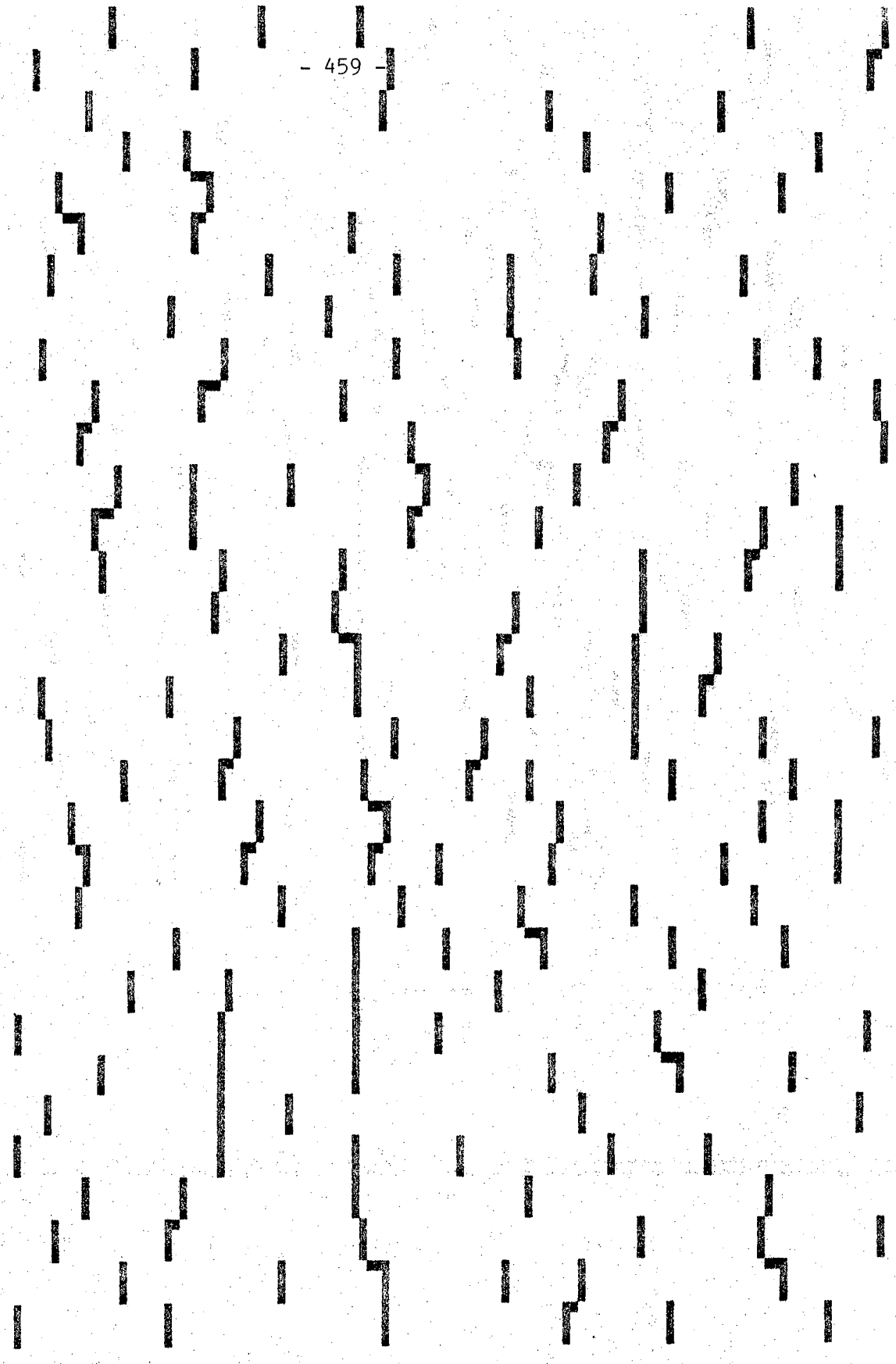


FIGURE-7.3 Figure shows the result of running RULE 20 on image in figure 7.2.

FIGURE-7.4 Figure shows the result of running RULE 21 image in figure 7.3.

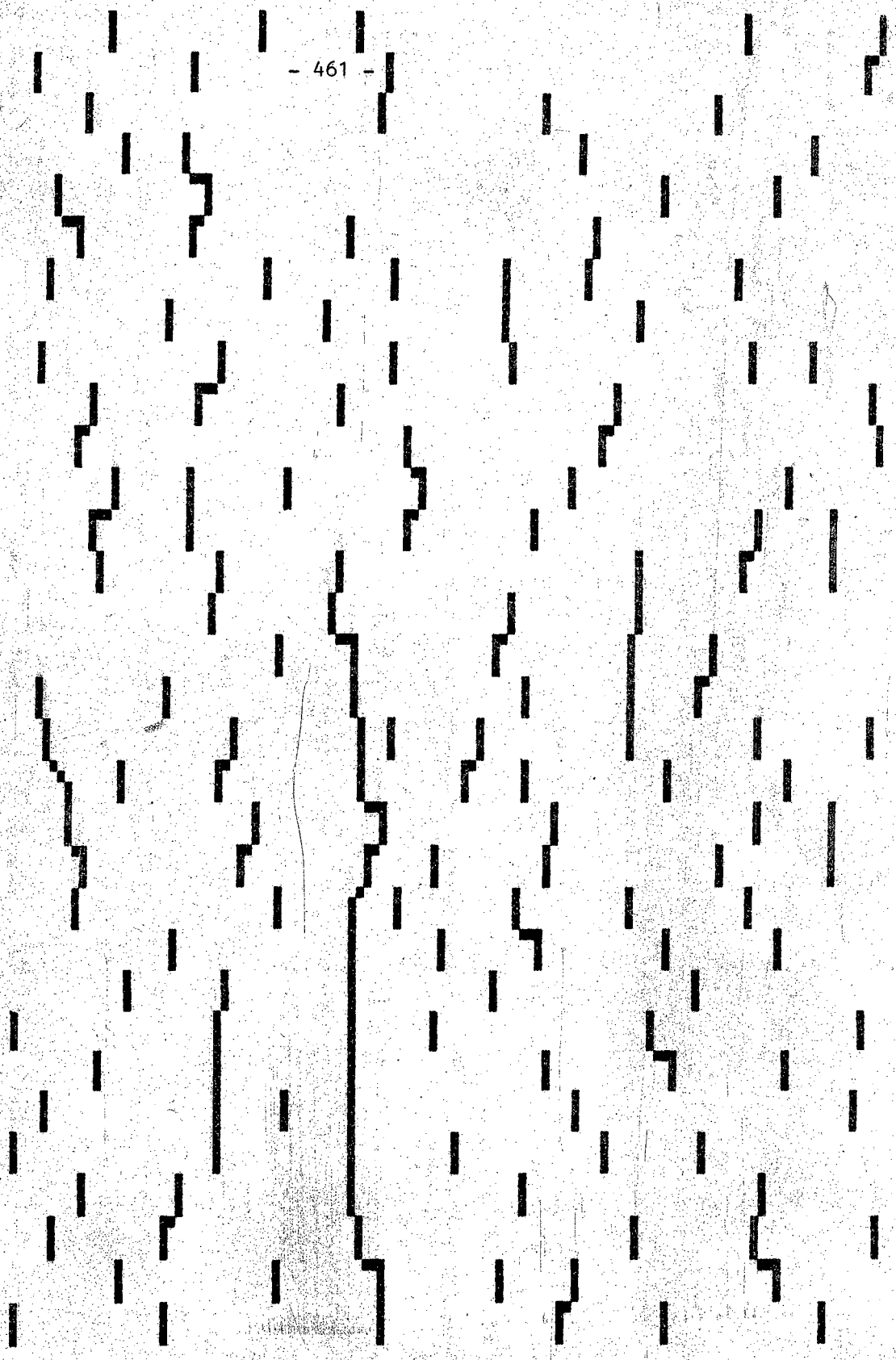


FIGURE-7.5 Figure shows the result of running RULE 22 on image in figure 7.4.

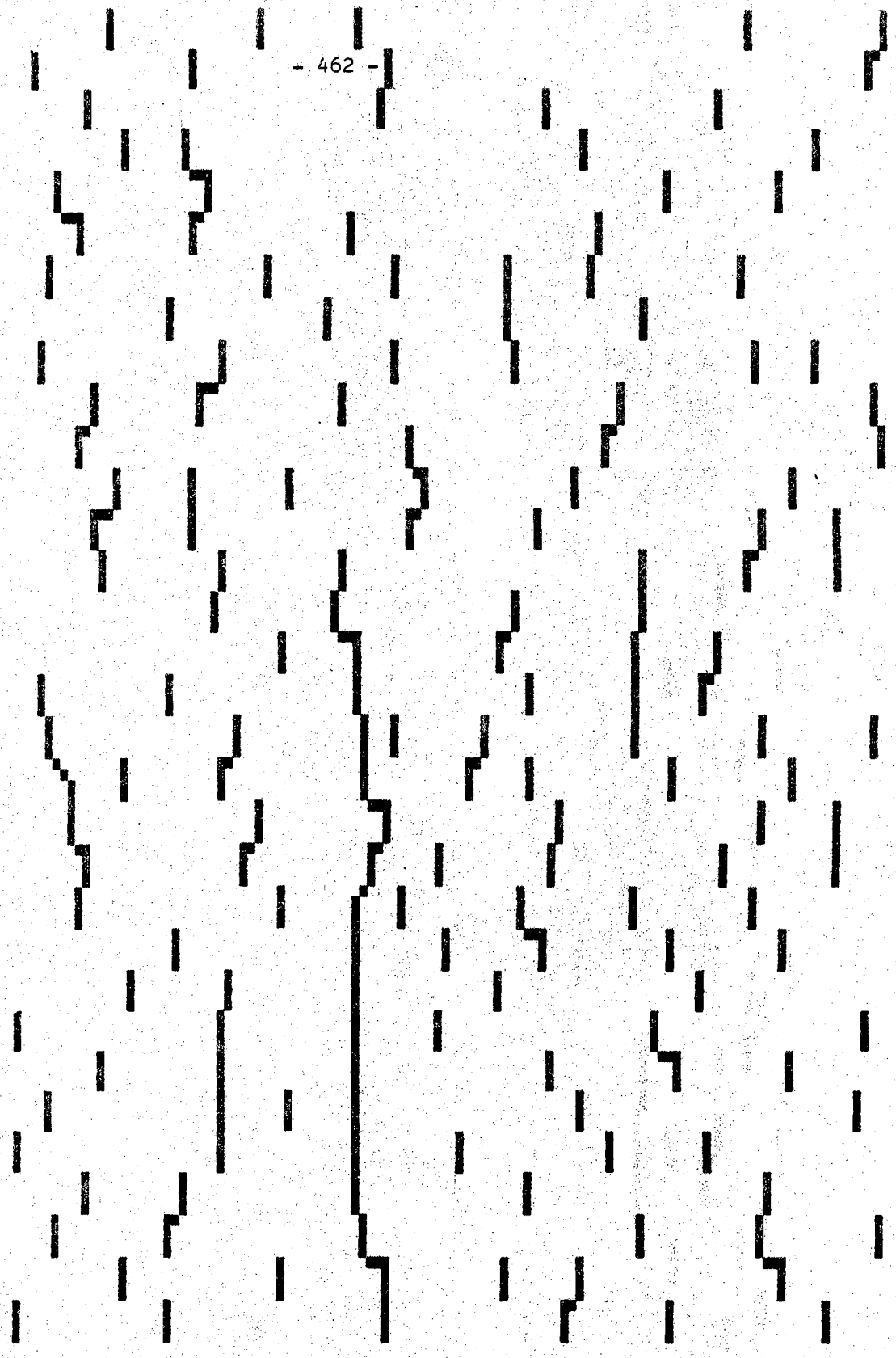


FIGURE-7.6 Figure shows the result of running RULE 23 on image in figure 7.5.

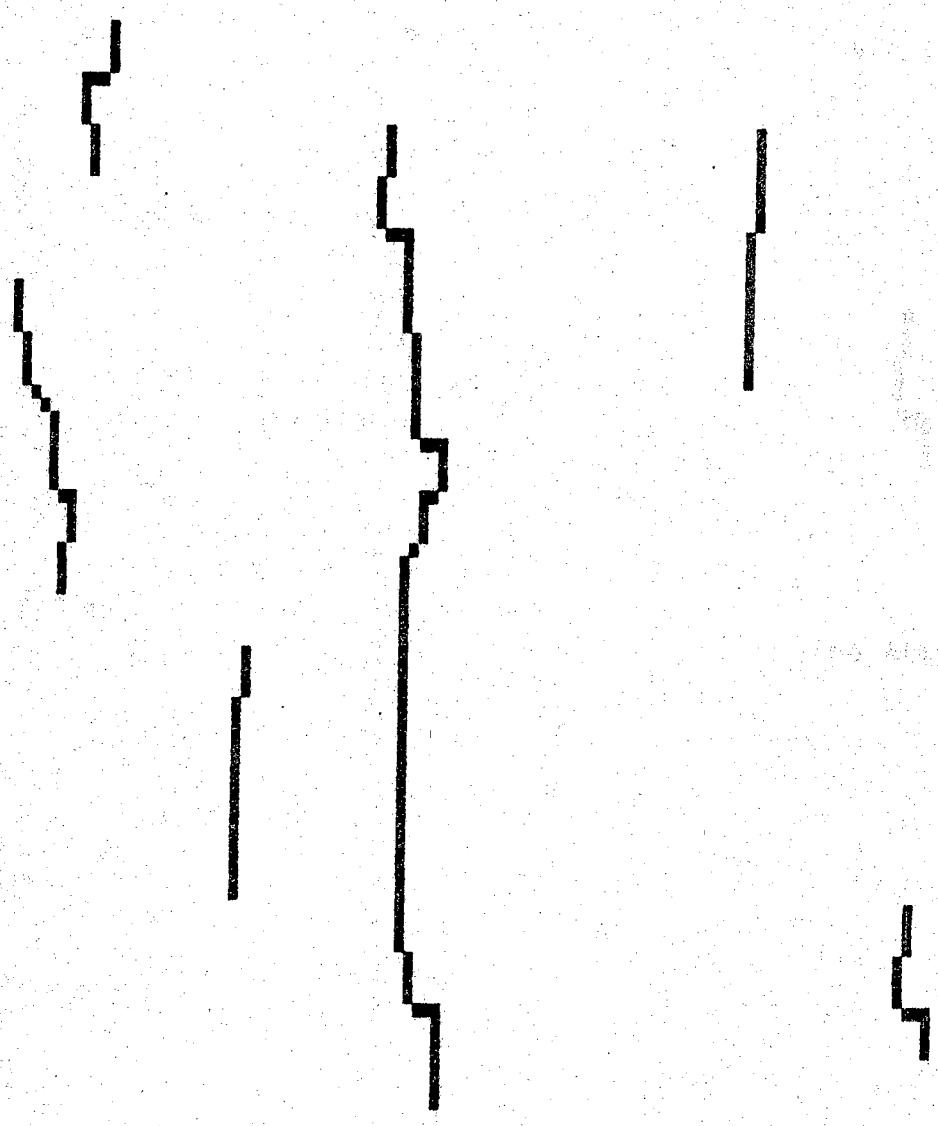


FIGURE-7.7 Figure shows the result of running RULE 41 on image in figure 7.6.

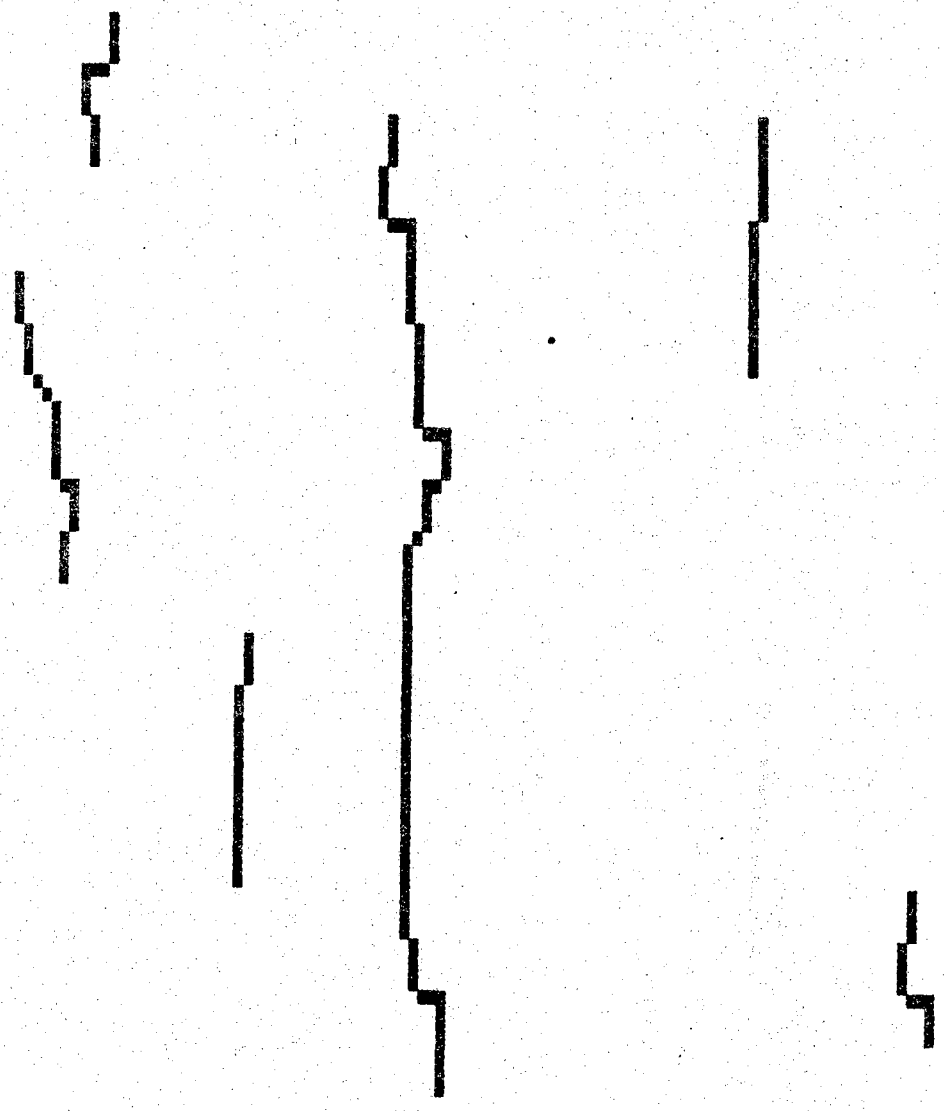


FIGURE-7.8 Figure shows the result of running RULE 24 on image in figure 7.7.

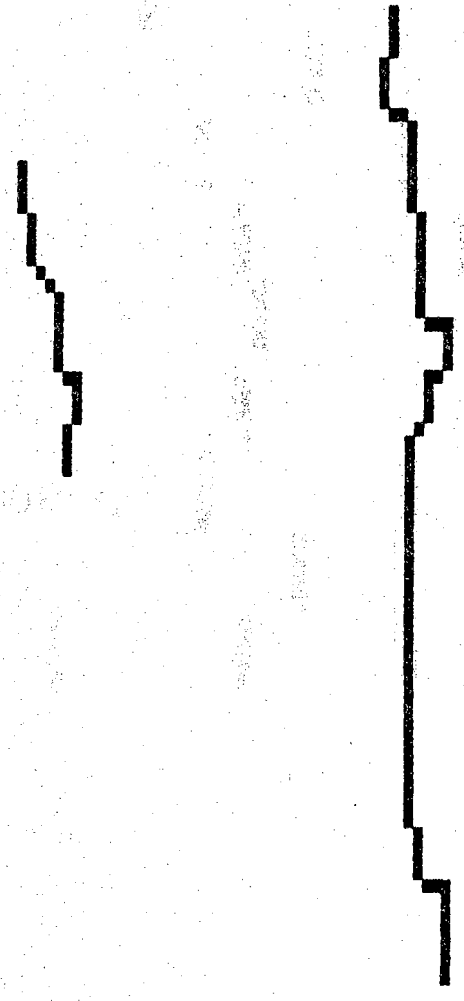


FIGURE-7.8 Figure shows the result of running RULE 42 on image in figure 7.8.

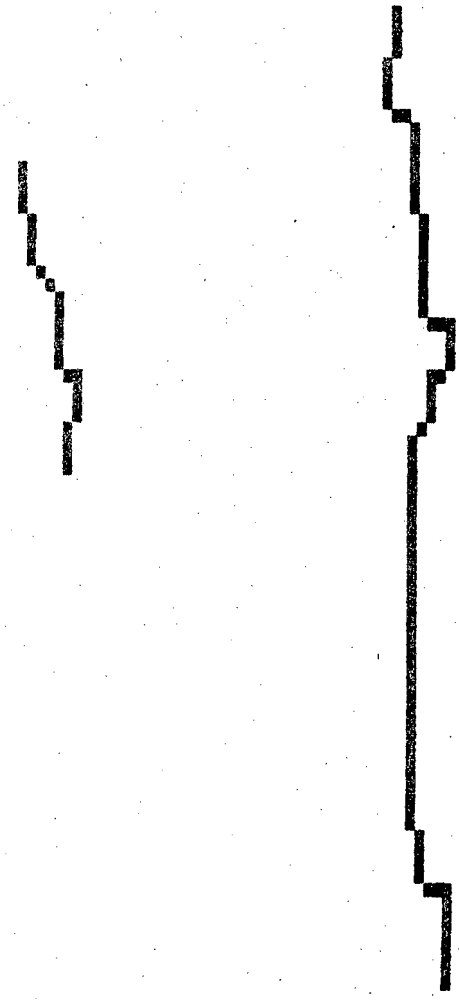


FIGURE-7.10 Figure shows the result of running RULE 25 on image in figure 7.9. This is the final output image.

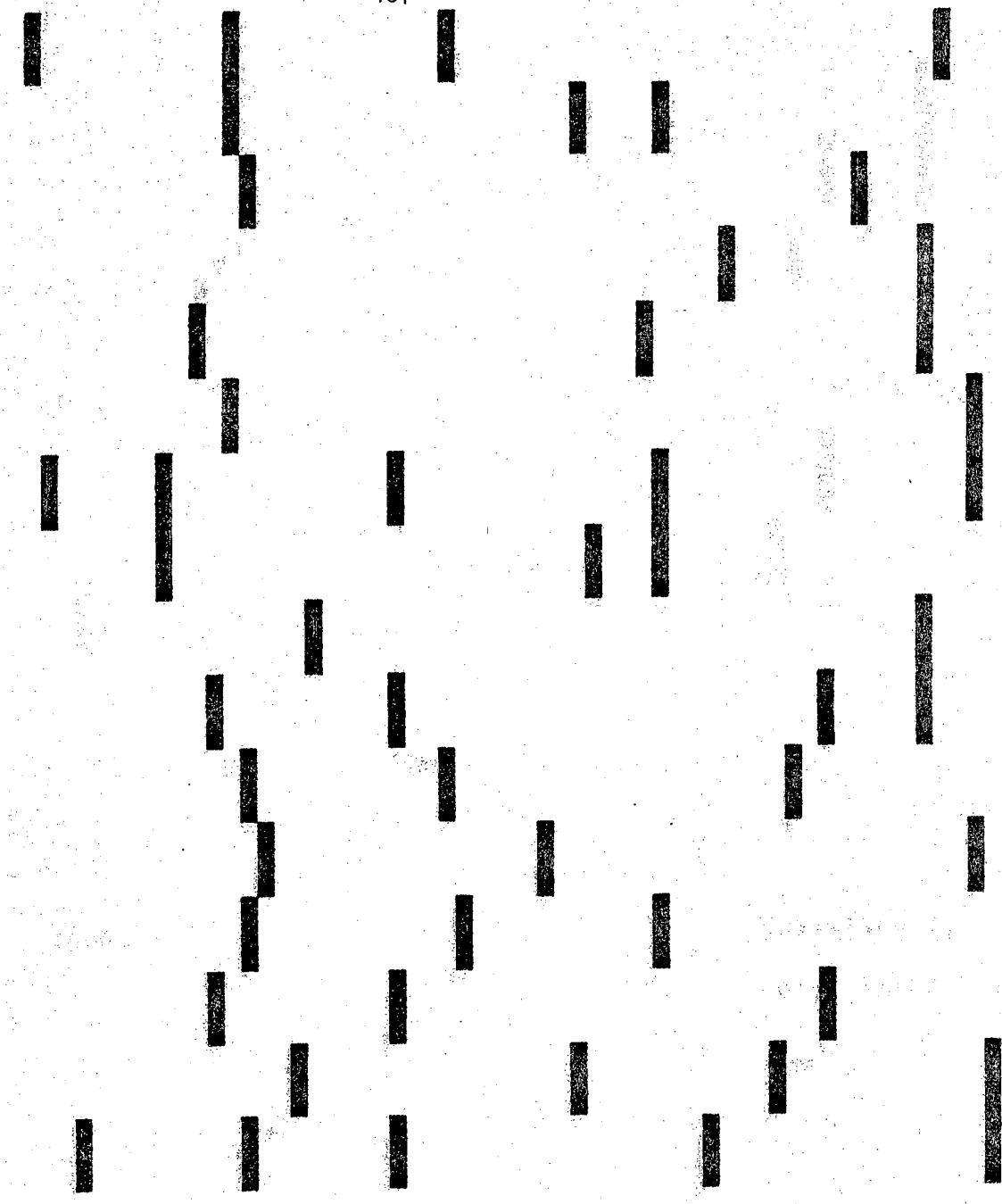


FIGURE-8.1 Figure shows extracted image consisting of a set of lines obtained by executing a boundary detection algorithm on the image.

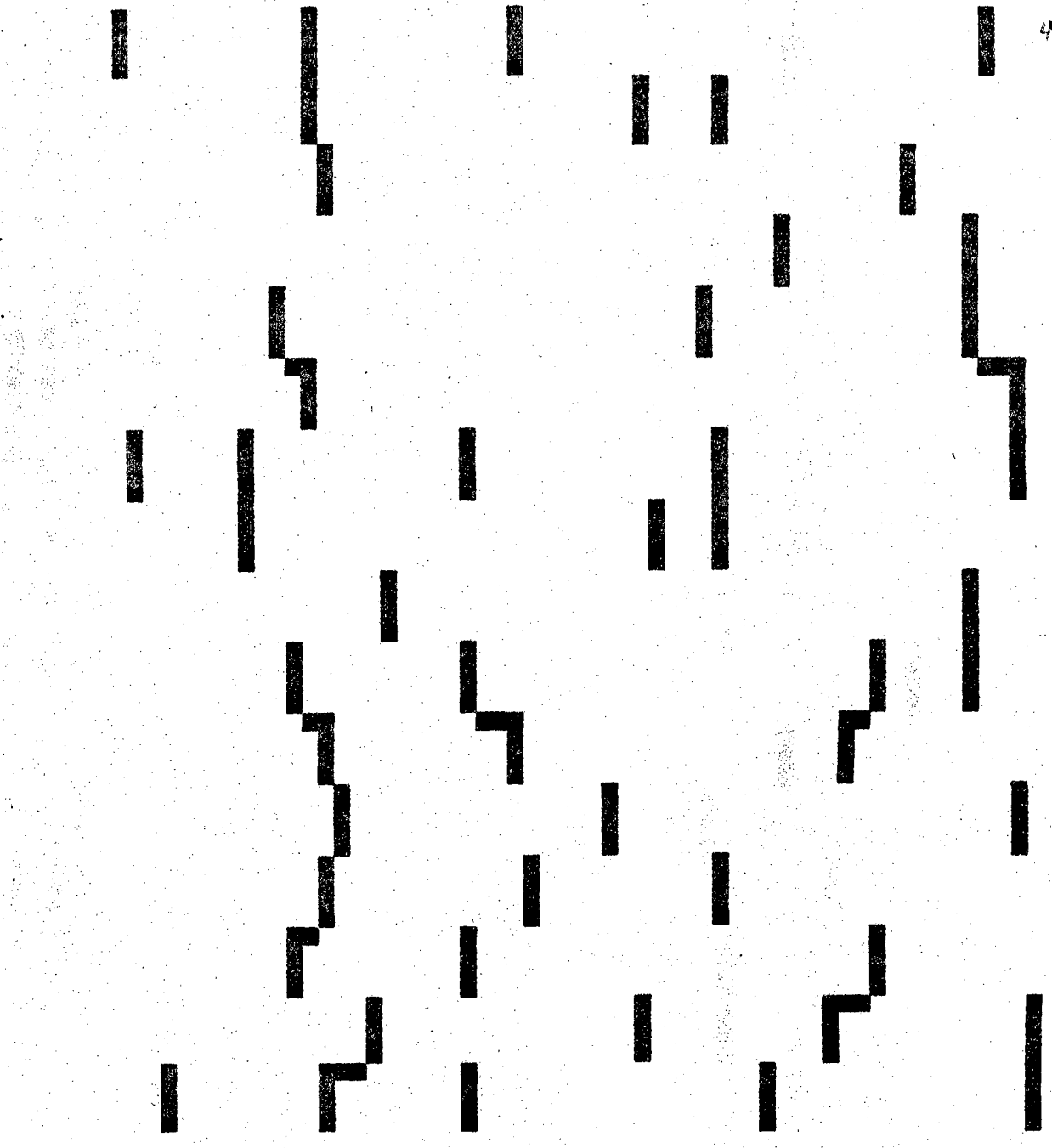


FIGURE-8.2 Figure shows the result of running RULE 19 on image in figure 8.1.

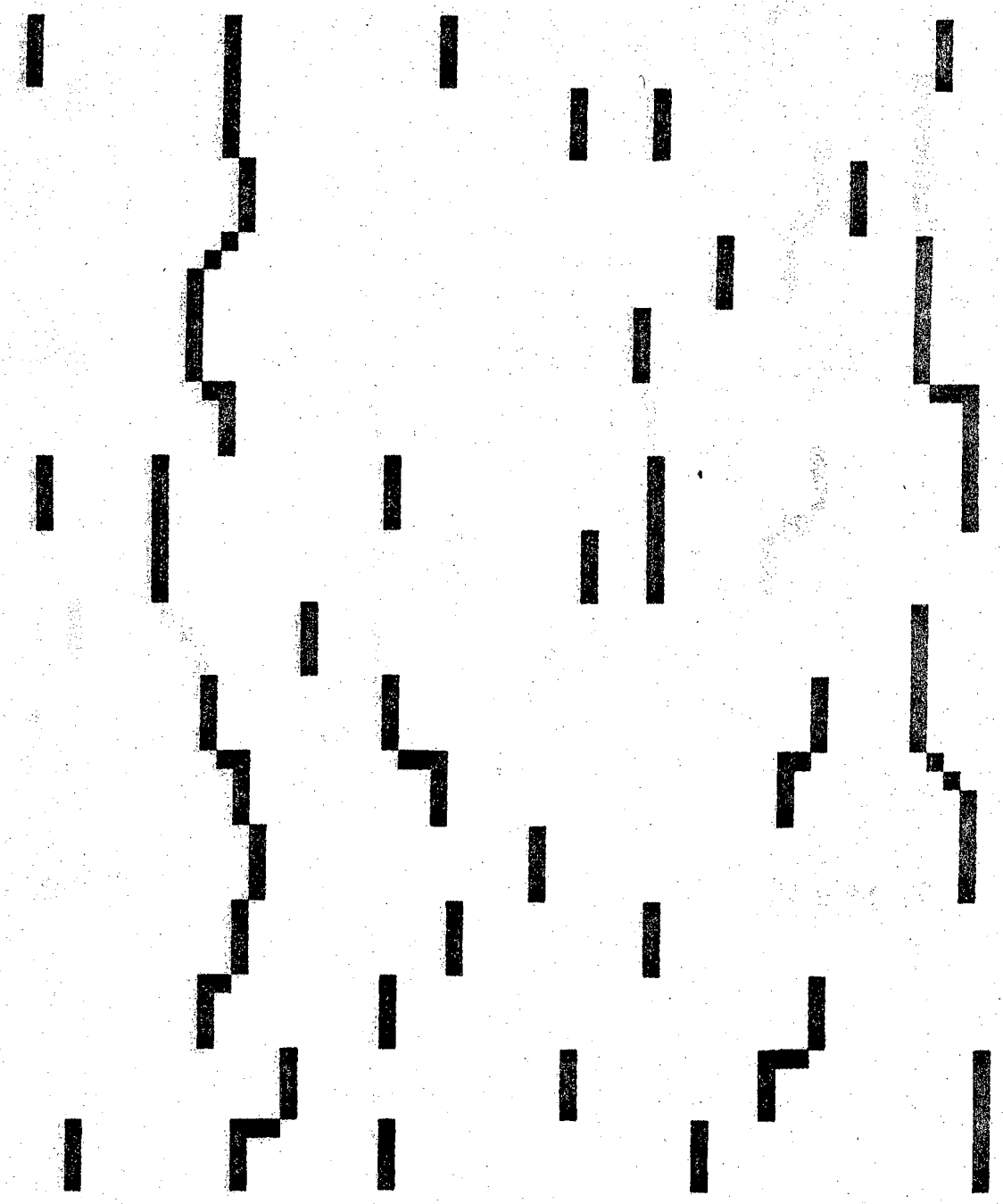


FIGURE-8.3 Figure shows the result of running RULE 20 on image in figure 8.2.

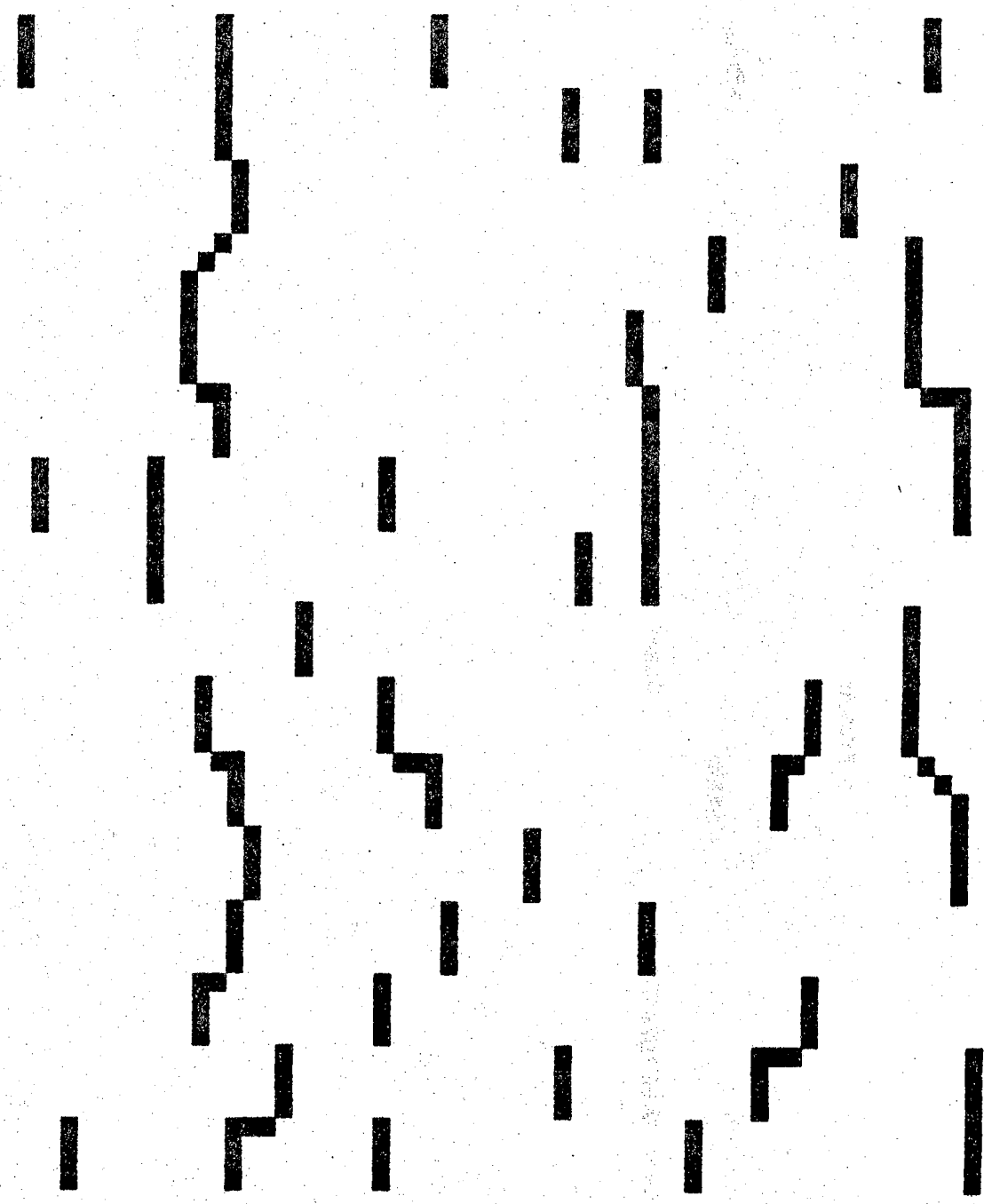


FIGURE-8.4 Figure shows the result of running RULE 21 on image in figure 8.3.

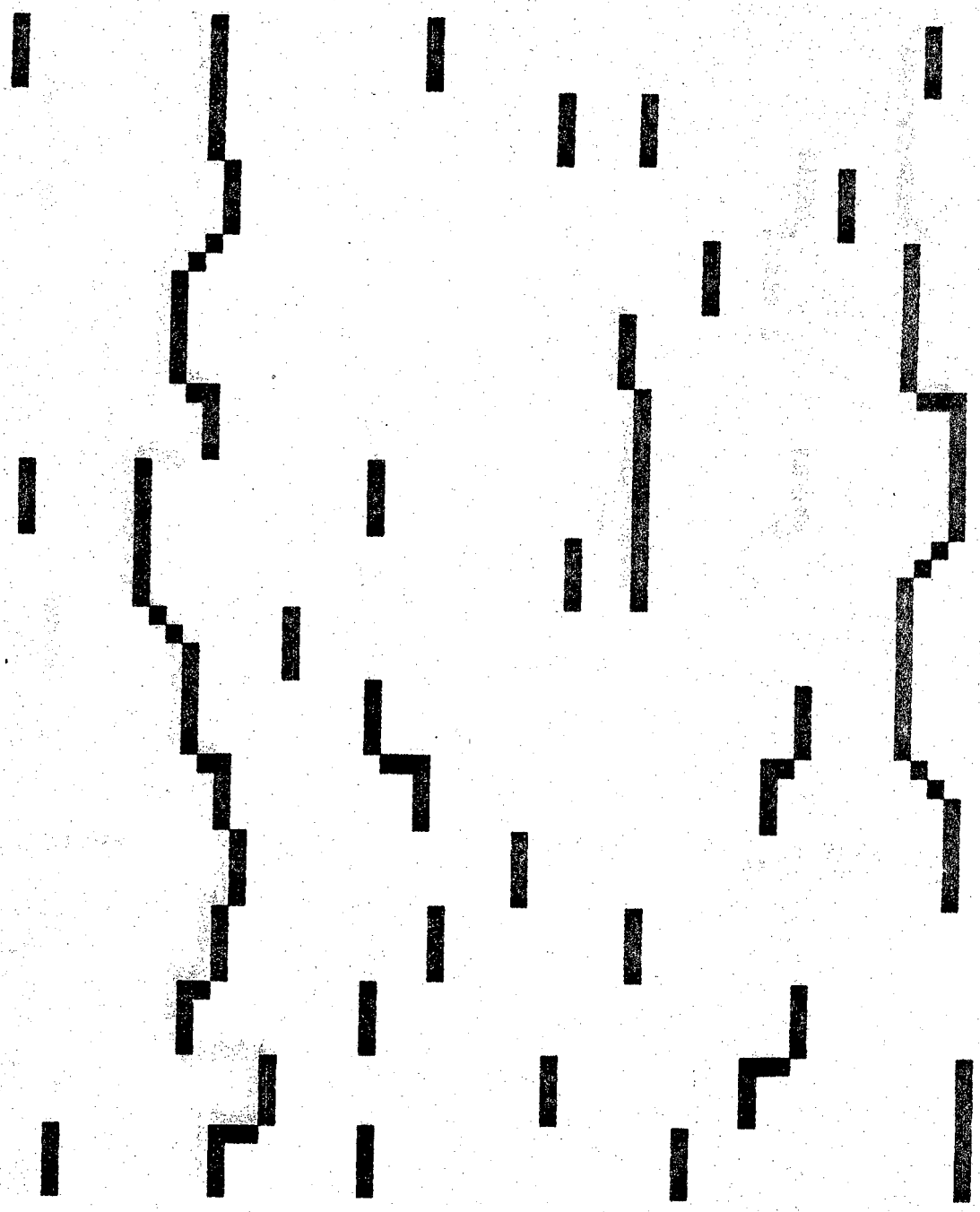


FIGURE-8.5 Figure shows the result of running RULE 22 on image in figure 8.4.

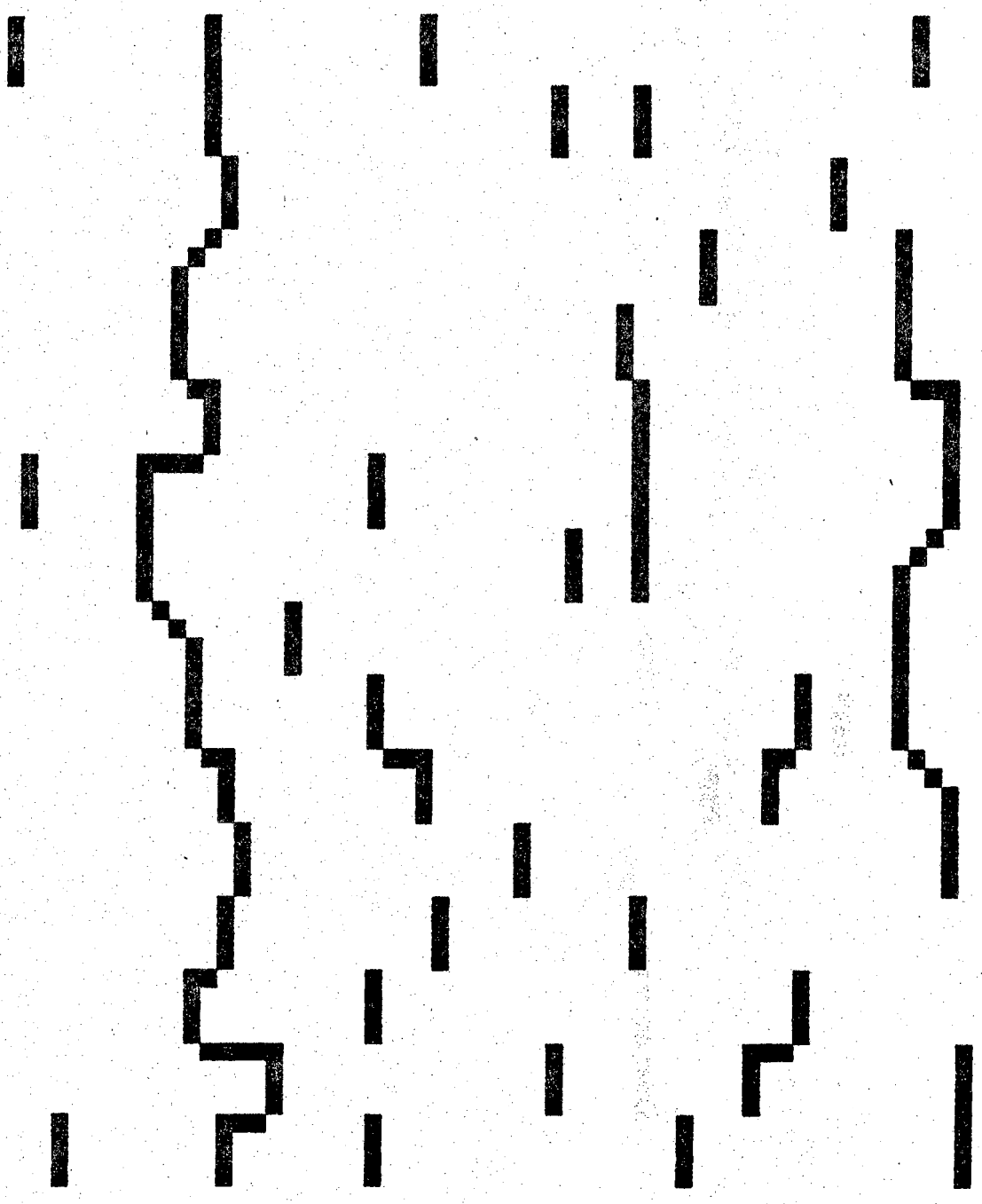


FIGURE-8.6 Figure shows the result of running RULE 23 on image in figure 8.5.

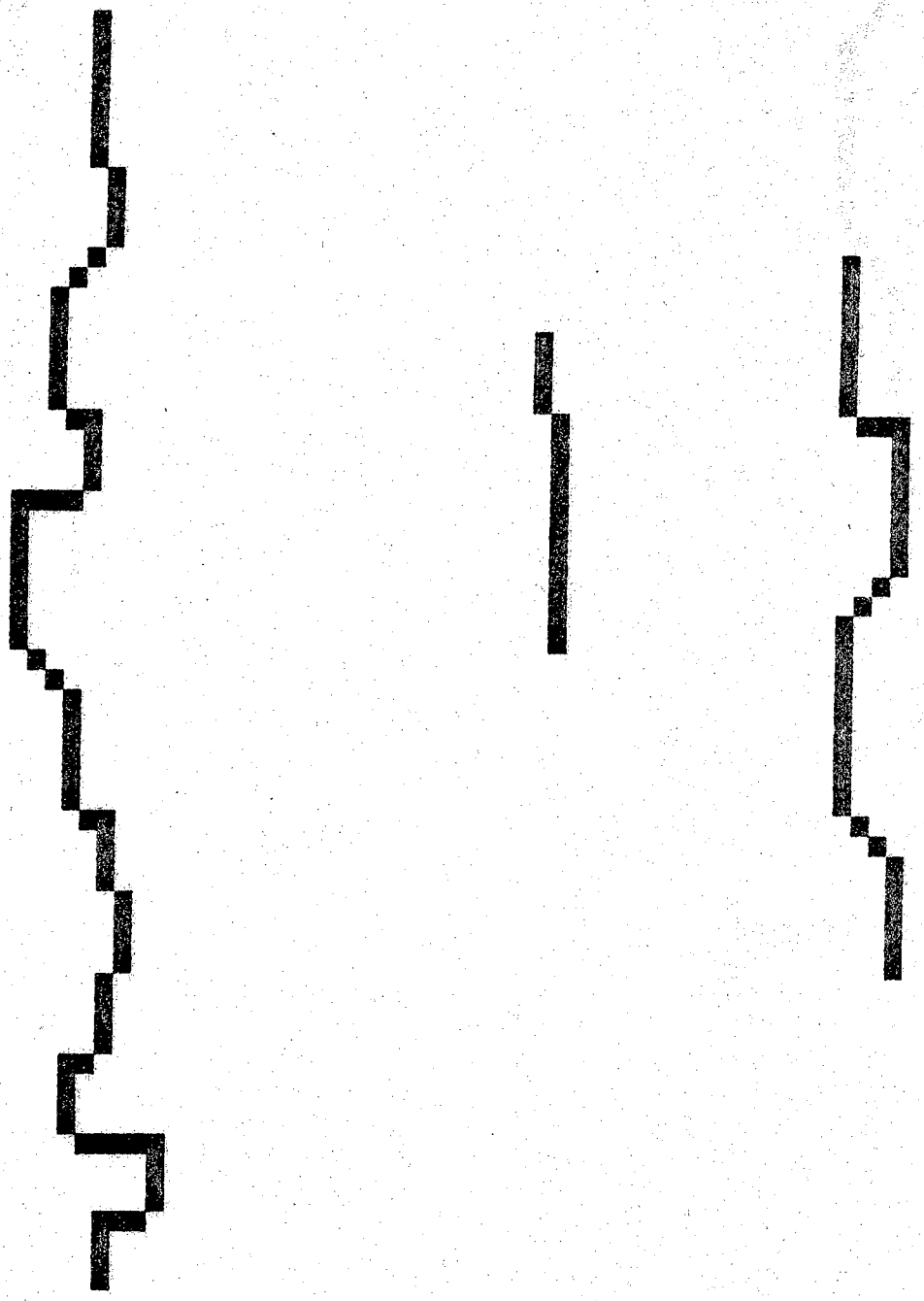


FIGURE-8.7 Figure shows the result of running RULE 41 on image in figure 8.6.

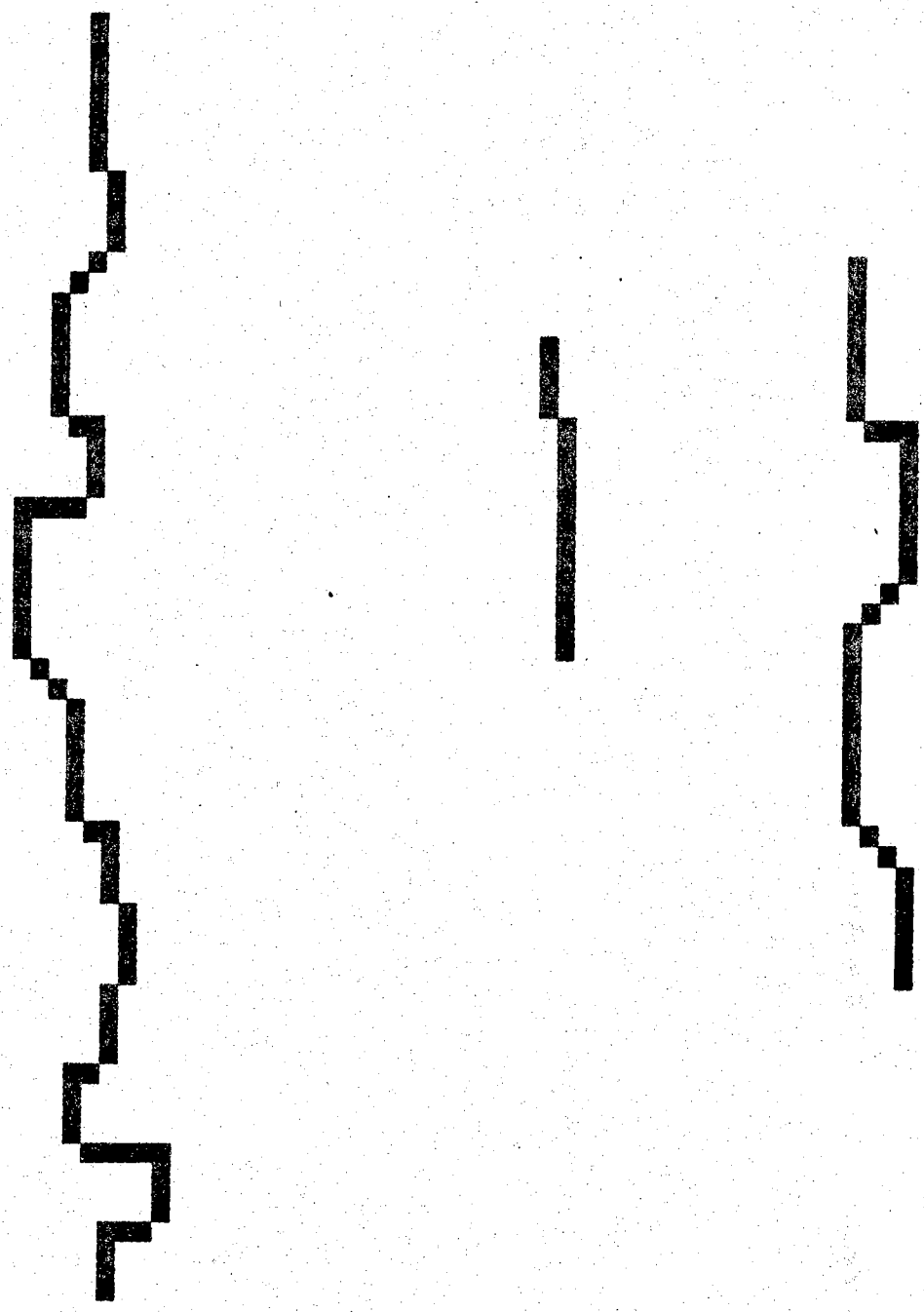


FIGURE-8.8 Figure shows the result of running RULE 24 on image in figure 8.7.

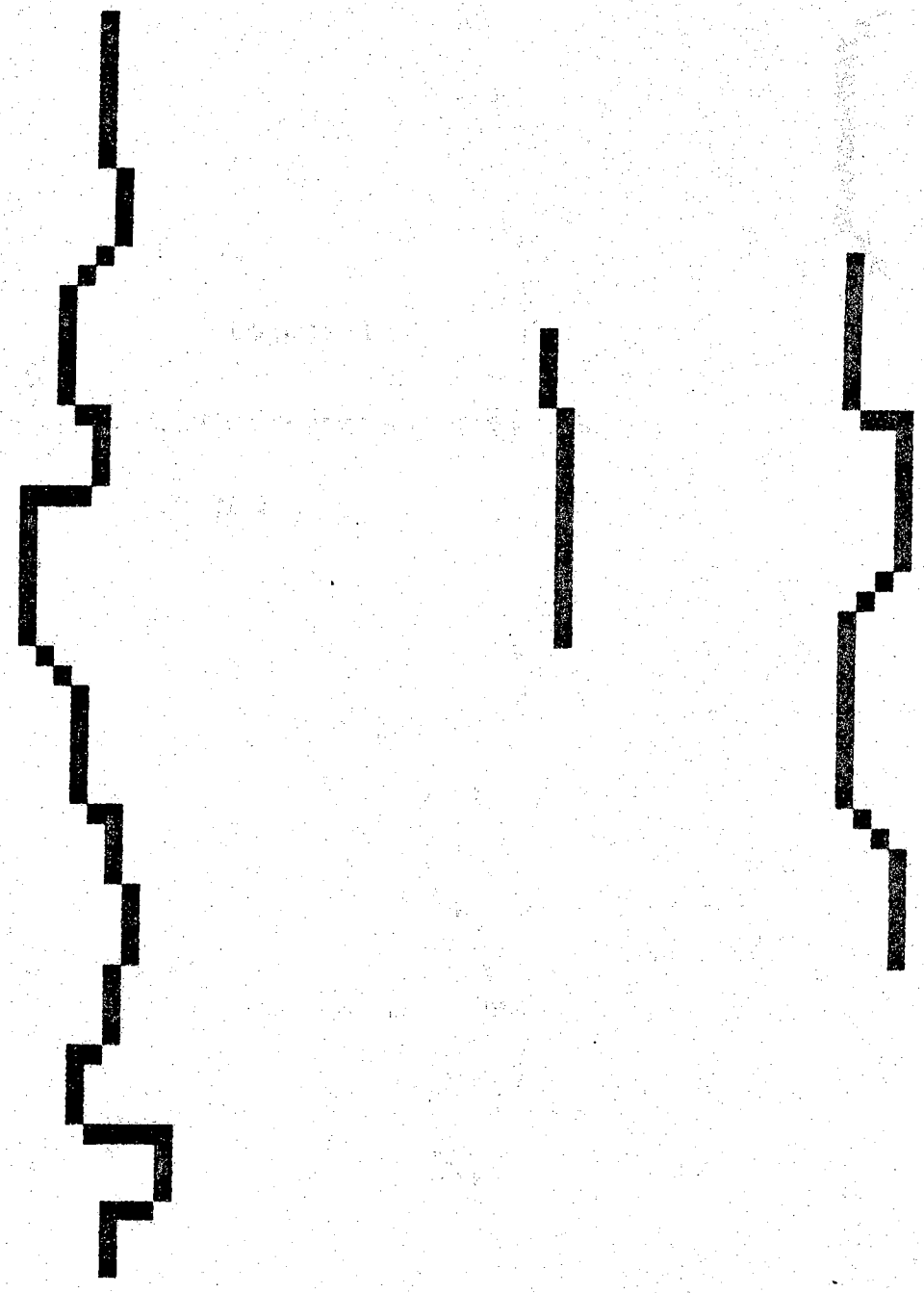


Fig. 8.9 Figure Shows the Result of Running RULE 42 on Image in Figure 8.8.

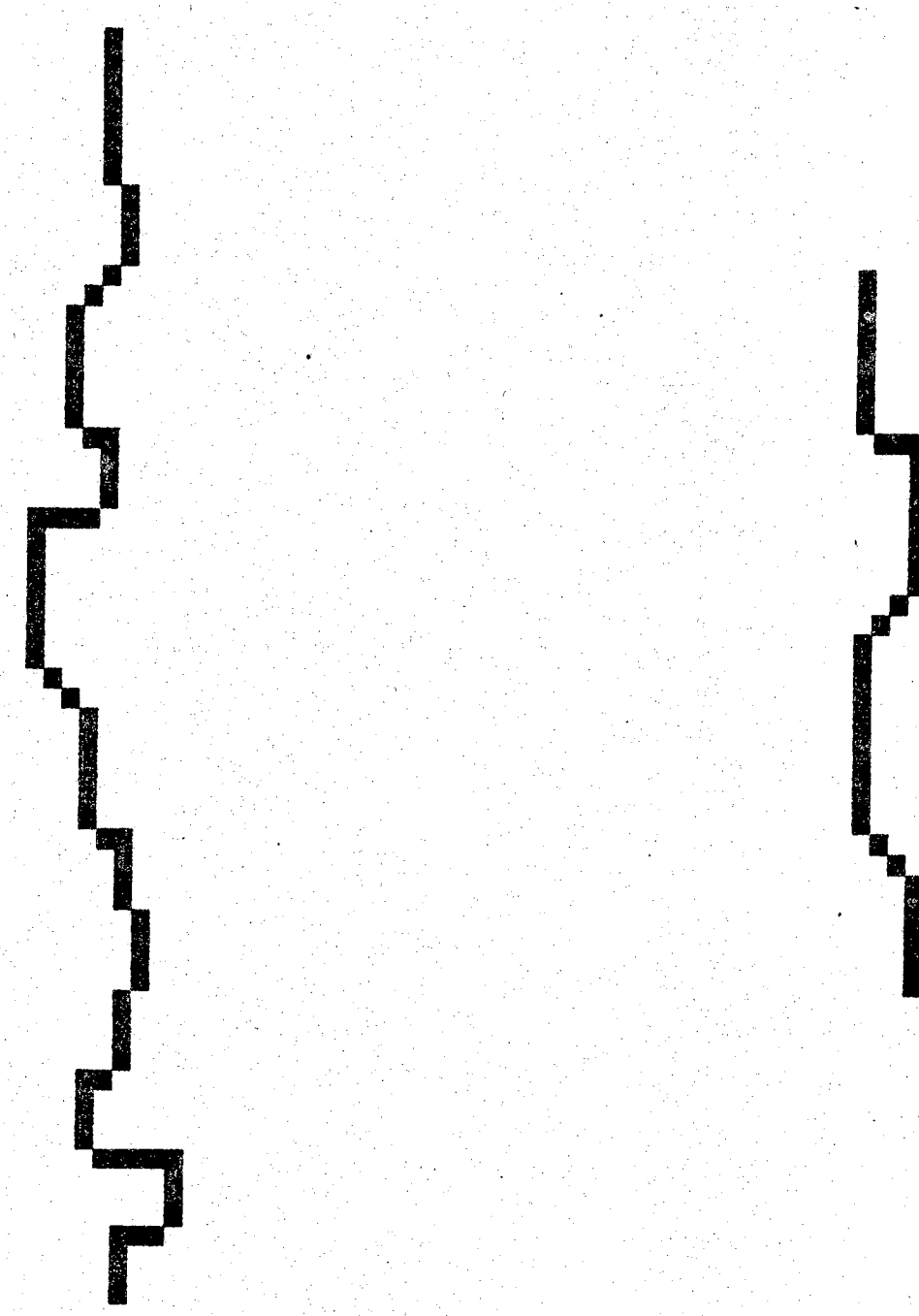


FIGURE-8.10 Figure shows the result of running RULE 25 on image in figure 8.9. This is the final output image.

Chapter 11

Stripe Pattern Interpreter and Stacker (SPIS): Expert System

H. S. Yang

STRIPE PATTERN INTERPRETER AND STACKER (SPIS) : EXPERT SYSTEM

EE 695G PROJECT

Hyun S. Yang

ABSTRACT

An expert system called SPIS(Stripe Pattern Interpreter and Stacker) is established for the Robot to identify 3-D objects, find optimum grip-point and stack those objects in the proper place(e.g. a box for the bricks, a post for the toroids etc.) in the order of size. The SPIS is hiring two experts such as SPI(Stripe Pattern Interpreter) and STACKER. In fact, the final goal of this project is to let the Robot Arm in the Robot Vision Lab to accomplish these goals. However, I am willing to put more

importance on the implementation of the SPI for the temporary goal. Since the segmentation of the complex scene is not my primary concern in this project, I will assume that the objects can occlude but not touch. The input to the SPI is the slit-projected image which contains 3-D coordinates information. The outputs are the name and size of the object, 3-D coordinates of the optimum grip-point, the type of container where the object will be stacked and the order of stacking. I will use low level information such as type of stripe segment, surface, axis, cross-section and hole obtained by 2-D and 3-D data. Rule base will be developed based on these low level information and exploited for deducing the augmented features of target object before matching in order to ameliorate the recognition accuracy. Forward chaining is adopted for the inference of desired outputs and meta rule based on the specificity ordering resolves the confliction when more than one rule are triggered while determining the optimal gripping point and the proper container.

1. INTRODUCTION

Recently, much effort has been made for the robot to recognize 3-D objects and grasp them for accomplishing some predetermined tasks. First of all, the robot must be able to recognize objects in the scene. As yet, most of the scene consist of objects with rather simple geometrical shapes. Furthermore, they usually do not overlap so highly that segmentation can be done with ease. To deal with more practical situation, it is stipulated that the vision system must be able to segment and recognize the occluded objects in the complex scene. Object recognition from the partial view cannot guarantee the accuracy. Even human cognition system might not recognize objects precisely if the overlapping is over threshold. However, relatively saying, human cognition system collaborating with the vision system, performs much better than the machine. In fact, human cognition system uses an assortment of low level information and reasoning mechanism to derive the conclusion. I intend to simulate the reasoning mechanism of human cognition system.

Before I begin the segmentation and recognition of 3-D objects, I intend to review the 3-D object representation. Certainly, there is no perfect object representation and recognition scheme, as performs well to any kind of objects. There is a wide gap between raw image seen by the camera and an understanding of what is seen. It is very difficult to bridge this gap. Three representation schemes seem to promote the exploitation of natural constraints. Two of these three representations are the primal sketch and 2-1/2 dimensional sketch, which describe

images from a *viewer-centered perspective*. The third one is the *world model* , which describes objects from a *viewer-independent* perspective. Information in the primal sketch and 2-1/2 dimensional sketch makes explicit what is going on at each point in the original image. Hence the primal sketch and the 2-1/2 dimensional sketch often are called to be *viewer centered*. Instead, the information in a world model often is expressed in terms of coordinate systems attached to objects. Such world models are called to be *object centered*. One approach for world model is to use *generalized cylinders* [9]. A generalized cylinder is a 2-D shape moved along a line called the *axis*. The shape is kept at a constant angle with respect to the line. The shape may be any shape. The axis need not be straight. Some objects have varying cross-sections. *Gaussian sphere* is a world model possessing 3-D surface orientation information [10]. To represent the object, object centered representation is prefered to the viewer centered one since describing the objects from the viewer centered perspective is ambiguous. By ambiguous I mean descrpition cannot be complete. Instead, object centered representation is complete in the sense that it contains viewer independent information for the objects.

In regard to the recognition of objects, one only can have partial, viewer centered information. Therefore recognition of object is matching viewer centered information about an object with object centered descriptions in the data base.

Human vision system is an expert system in which a lot of informations as to objects such as intensity, texture, color, depth, etc. are controlled heteroarchically and parallely to reach the correct recognition of objects in 3-D space. It seems to me that the human vision system

employs an expert reasoning mechanism deducing the conclusion as to object shapes by following plausible lines of reasoning starting from the partial or insufficient informations captured through eyes. Especially, for the purpose of recognizing object in the highly complex scene in which the amount of obtainable information about the object is quite restricted, this deduction mechanism might be adopted more frequently. To mimic the deduction mechanism of human cognition system, one might collect as much raw features as he can from the vision system and produce augmented features through rule based deduction system from the raw features. To ameliorate the recognition accuracy, this augmented feature base is compared with descritions in the data base to select the most feasible object class or name.

The reason why I am employing ranging device is that I finally will let the robot to locate and grasp the target object and move it to a certain place. I hesitate to adopt ranging scheme based on the stereoscopic binocular vision since the so called corresponding problem has not yet been solved completely and moreover the information I can get from the structured light based ranging device is much more than that from the stereoscopic depth perception.

Optimal gripping point for the robot has not been considered much , as yet. The objects handled by the vision-based robot have not been very complicated enough to require the optimal gripping point. But, for the more intelligent robot to achieve the rather sophisticated goal, it might be necessary for it to find out the best gripping point for the target object. For instance, trying to grasp a box without knowing its size in advance might enable for the robot not to be able to grip the box in case the width(or length) of the box is greater than the gripper size. The

robot must measure the size of the box first and then let the gripper to grasp the box so that it can handle. Or if the size of the box is out of the capability of gripping, simply give up gripping or push it away to make scene less complex. Usually, it is safer to grip the object center which possibly corresponds to the center of mass of the object. But if the task of robot is gripping the screw driver to insert the screws into the wall, the robot must grip the handle part of the screw driver. Moreover, the center of the handle might be preferred. For picking the curved object such as C-curved cylinder or S-curved cylinder, the best gripping point could be the point where the curvature changes maximally since the touching area is the maximum around that point. Conclusively, the optimal gripping point must be determined by the shape of object, the usage of object, and the goal of the task. For this project, I assume the goal of the task is stacking the objects in the scene in the proper container by the order of size. However, the class of objects and the goal of the task can be generalized more.

2. SYSTEM ORGANIZATION

Knowledge base is comprised of data base and rule base. Data base is constructed by frames which store object names, slots representing object class (AKO: a kind of), object features such as number of

primitives, type of stripe pattern, axis, cross-section, surface, angle between adjacent surfaces, edge, hole, and size. Number of primitives provides the global attribute of object. Axis, cross-section, surface, angle between adjacent surfaces, edge, hole, and size are all extracted based on 3-D data. Type of stripe pattern gives 2-D information about object surface , cross-section, and global attribute like the number of simple parts. However, type of stripe is not a redundant feature since surface orientation andcross-section information based on range information might not always be correct. Credibility as to those information might be improved by using type of stripe pattern information.

Name of the frame is the name of the object, slot is the name of the feature and the filler is the value of the feature. The objects include 3-D primitives such as block, plate, stick, hexagonal prism, octagonal prism, a dodecahedron, icosahedron, cylinder, sphere, cone, toroid and more complex objects comprising more than one 3-D primitives. Features used for representation of objects are all object centered. For instance, a cube has only one 3-D primitive, no axis, square as a cross-section, six square surfaces, 12 straight edges, same width, length,and height, etc. Those features are absolutely viewer-independent. Description of complex objects is different from that of the simple primitive. Complex objects are represented by the number of simple parts, the name of each simple part, and connection relations between adjacent simple parts.

Rule base consists of knowledge rule and meta rule. For generating augmented features, I prefer to fire all the rules satisfying the facts, instead of triggering the most specified one, for the purpose of having as many features as possible before matching. Matching is done by

searching for the prototype in the data base which has maximum similarity with the target object. I do not develop the rule base for object recognition so that the rule base precisely deduce the name of the object. The reason is that it would require too many rules fired until finally the rule determines the name of object. The rule base should be unnecessarily large to recognize complex object in that case. Instead, I would rather develop the rule base firing maximum features starting from the extracted features through low and mid level process and confining the object class so that matching the features of candidate object with every prototype in the data base is not necessary. For example, if the candidate object is known as simple (containing only one 3-D primitive), only the simple object descriptions in the data base are matched. By this, one can reduce the matching time very much.

Once the name, size and class of the object are determined, rule base is activated to infer the usage of the object based on those information and then the optimal gripping point and the proper container is determined based on the object shape and usage. But since there exists only one optimal gripping point or container, we must exploit meta rule to solve conflict resolution in case more than one rule is fired. I adopt the specificity ordering for the conflict resolution. Namely, the rule that is more specialized to the current situation will have the precedence.

Fig.1 describes the procedures of the system. The input to the SPIS is the slit projected image containing 3-D coordinates. The output of the SPI is the name(or class) of the object, optimal gripping point as 3-D coordinates, type of container. STACKER will be activated then to grasp and move the object into the container specified. This part of the

process will not be taken into account in this project. Since objects do not touch, we might assume the scene will not be disturbed by moving an object. However, it is safer to move the object which is at the outer-most (front-most, back-most, left-most, right-most) position in the scene since there is less chance to touch the rest of the objects while the gripper is moving. Therefore the object at the outer-most position will have the priority as far as the robot picks up the object in the order of safety. I give the precedence to the front-most object since it might be the nearest one to the robot. As far as we may assume the scene is not disturbed while gripping and moving, we may use the old data to deal with new object. That is a great advantage because, otherwise, the scene should be projected again to update the new information as to the changed scene.

The Scheduler determines the best target after the Segmenter isolates every object in the scene successfully and load only this object in the STM (Short Term Memory) for the purpose of focus of attention. The local experts such as STA (Stripe Type Analyzer), SA (Surface Analyzer), ACA(Axis Curvature Analyzer), CA(Cross-section Analyzer), HA(Hole Analyzer) are then activated to extract features of the target object in the STM. Features extracted from local experts will be interacted with rule base to generate more features. For the object recognition,the augmented features are exploited as mentioned above. The Scheduler terminates the process when no more object is left in the scene.

The programming for accomplishing this task contains three parts, low level, mid level, and high level. Low level consists of determination of stripe type, calculation of local surface normals, axis computation, cross-section extraction, 3-D hole extraction.

Mid level exploits low level information to produce the fact list as the starting point for the high level forward chaining. STA (Stripe Type Analyzer) makes use of 2-D stripe information to check the surface property and decompose the complex object into the simple parts. NOHA (Needle Orientation Histogram Analyzer) and NMHA (Needle Magnitude Histogram Analyzer) are activated to determine the type of surface. ACA (Axis Curvature Analyzer) examines the object curvature to determine the type and number of axis, the number of primitives, and the point of inflection. CA (Cross-section Analyzer) is used for separating simple parts and determination of cross-section type.

High level is composed of data base, rule base, and inference engine written by LISP.

3. KNOWLEDGE ACQUISITION

It is rather easy to acquisite knowledge about object shapes and usages. One may develop object centered representation of object by almost common sense as to the 3-D shape information of the 3-D objects. No expertise is necessary for object representation. But, feature extraction and analysis from the slit projected image requires an expert having experience with structured-light based ranging device and 3-D feature extraction based on range data. The author has been

working on those areas for a couple of years and could be considered as an expert. For the range data extraction by structured light system and feature extraction by the range data, [11] is referred. Developing rule base is also based on the common sense and expertise. For example, it is trivial to conclude by common sense that

if an object is comprised of only linear stripes, then it is a planar object.

If an object has only same length of stripes, then it has only one simple part.

If an object has curved cross-section, then it is a curved object.

Instead, it needs some expert knowledge about the related expertise to deduce that

if an object has discontinuous distribution in the needle orientation histogram, then it is a planar object.

if an object has triangular distribution in the needle magnitude histogram, it has a spherical surface.

if an object has the slope polarity change in the axis curvature, it is a bended object.

if an object has a group of stripes containing two range discontinuities each, it has a hole.

4. EXPERIMENT

For the experiment, the scene consists of cubes (large, small), toroids (large, small), a sphere, a cylinder, a C-curved cylinder, a dodecahedron, a cup and a motor (Fig.2). Objects do not touch, albeit occlude. Since objects do not touch, objects can be segmented by component labelling followed by range discontinuity detection. After segmentation is successfully accomplished, the minimum z-coordinate of each object is calculated to select the nearest object to the robot. The object which has the minimum z-coordinate is selected as the best target. If more than one object have the same z-coordinates, the largest object among the candidates is selected as the best one. The reason I give the priority to the larger object is that removing the larger object makes the scene less complex. Based on this rationale, the C-curved cylinder has been selected as the first target for the robot since it has the minimum distance from the robot. In fact, the distances from the robot to the motor and to the C-curved cylinder are almost same, but C-curved cylinder has precedence over the motor because it is much larger. Fig.3 describes the segmented C-curved cylinder by component labelling and range discontinuity detection. For the feature extraction of the target object, I used algorithms such as local surface normal calculation by least squares method, needle diagram generation, 3-D hole detection, and axis curvature estimation. They have been written by FORTRAN. Data base, rule base, and inference engine are written by LISP. The trouble with interfacing low level data with high level is the time spent by LISP when one tries to do the number crunching work with LISP. It is almost impossible to do the feature extraction by time sharing LISP. We only can do each part separately. Segmentation and feature extraction are done by FORTRAN and the extracted features are

stored in the property array. LISP reads and interprets the property array, and generates a fact-list. Fact-list is then a starting point for the deduction of augmented features and the object recognition. As far as the algorithms are concerned, I have done all the low level parts including segmentation and feature extraction, and programmed inference engine based on forward chaining, database based on frames. But mid level comprising an assortment of analyzers which are supposed to anayze low level information and convert the extracted object features into abstract property forms has not been done yet.

For the C-curved cylinder, features are extracted from the 3-D information. STA(Stripe Type Analyzer) provides the curved stripe patterns whose lengths are almost same. ACA(Axis Curvature Analyzer) analyzes the curvature (Fig.4) of the C-curved cylinder, and concludes the object is bended with one inflection point. CA (Cross-section Analyzer) detect curved cross-sections whose length are almost identical. HA (Hole Analyzer) fails to detect 3-D hole. Extracted features are put into the fact list and inference engine is activated to deduce more features until no rule is fired. The augmented fact list is then used to find out the most similar object in the database by matching only the descriptions in the same class as the target object.

This project is too big to be done in a semester by one researcher. I am going to continue to implement SPIS completely. As a EE695G project, I like to turn in partial result and programming I got. Those include: segmentation by component labelling and range discontinuity, needle orientation and magnitude histogram, 3D hole detection, axis curvature estimation, cross-section analysis, forward-chainning(deduction), frame-based database, IF-THEN rulebase.

5. CONCLUSIONS AND DISCUSSIONS

As the scene becomes more complex due to occlusion or occluding, object recognition becomes more difficult and the recognition accuracy decreases. Deduction mechanism simulating human cognition system could produce more features starting from the limited features obtained through 3D vision system. Since the mid level which analyzes low level information to extract features does not guarantee to reach perfect answer, certainty factor should be added to the rules used for object recognition. In other words, IF-THEN reasoning rules must adopt FH (Finding-to- Hypothesis) rule or HH (Hypothesis-to-Hypothesis) rule. To reduce the object recognition time and the space of rule base, we had better not develop rule base which employs rules deducing the precise name of object. For the simple objects used as an example, this rationale may not sound legitimate. But I believe it does when we deal with more complex objects.

As the robot becomes more and more intelligent, optimal gripping point reasoning based on object shape and usage will be crucial. If the goal of the task is stacking, it would be necessary to select the proper container for a certain object and stack objects in the order of size.

REFERENCE

1. F. Hayes-Roth, et al., Building Expert Systems, Addison-Wesley, 1984.

2. S. Weiss and C. Kulikowski, A Practical Guide to Designing Expert Systems, Rowman and Allanheld, 1984.

3. N. J. Nilson, Problem Solving Methods in AI, McGraw-Hill, 1971.

4. N. J. Nilson, Principles of Artificial Intelligence, McGraw-Hill.

5. A. Barr and E. A. Feigenbaum eds.,The Handbook of AI, Vol. 1,2,3, Kaufmann Inc., 1981,82.

6. P. H. Winston, Artificial Intelligence, 2nd ed., Addison Wesley, 1984.

7. P. H. Winston and B. K. P. Horn, LISP, Addison Wesley, 1981.

8. P. Kinnucan, "COMPUTERS THAT THINK LIKE EXPERTS," High Technology, Jan. 1984, pp. 30-43.

9. G. Agin and T. Binford, "Computer Description of Curved Objects," 3rd IJCAI, 1973, pp. 629-640.

10. K. Ikeuchi and B. Horn, "Numerical Shape from Shading and Occluding Boundaries," Artificial Intelligence 17, 1981, pp. 141-184.

11. H. S. Yang, K. L. Boyer and A. C. Kak, "Range Data Extraction and Interpretation by Structured Light," the 1st Conference on Artificial Intelleigence Application.

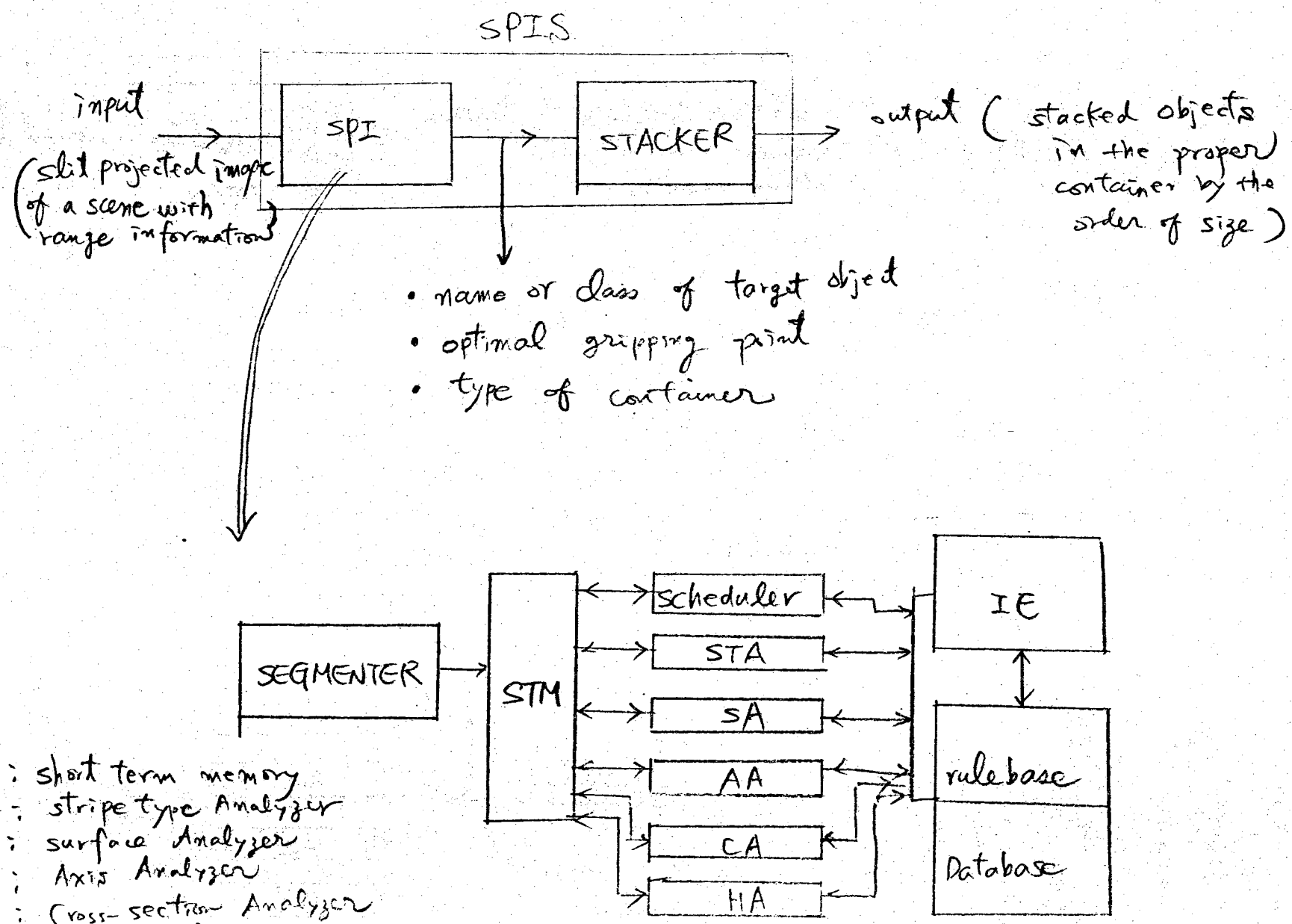


Fig. 1 BLOCK DIAGRAM FOR SPIS

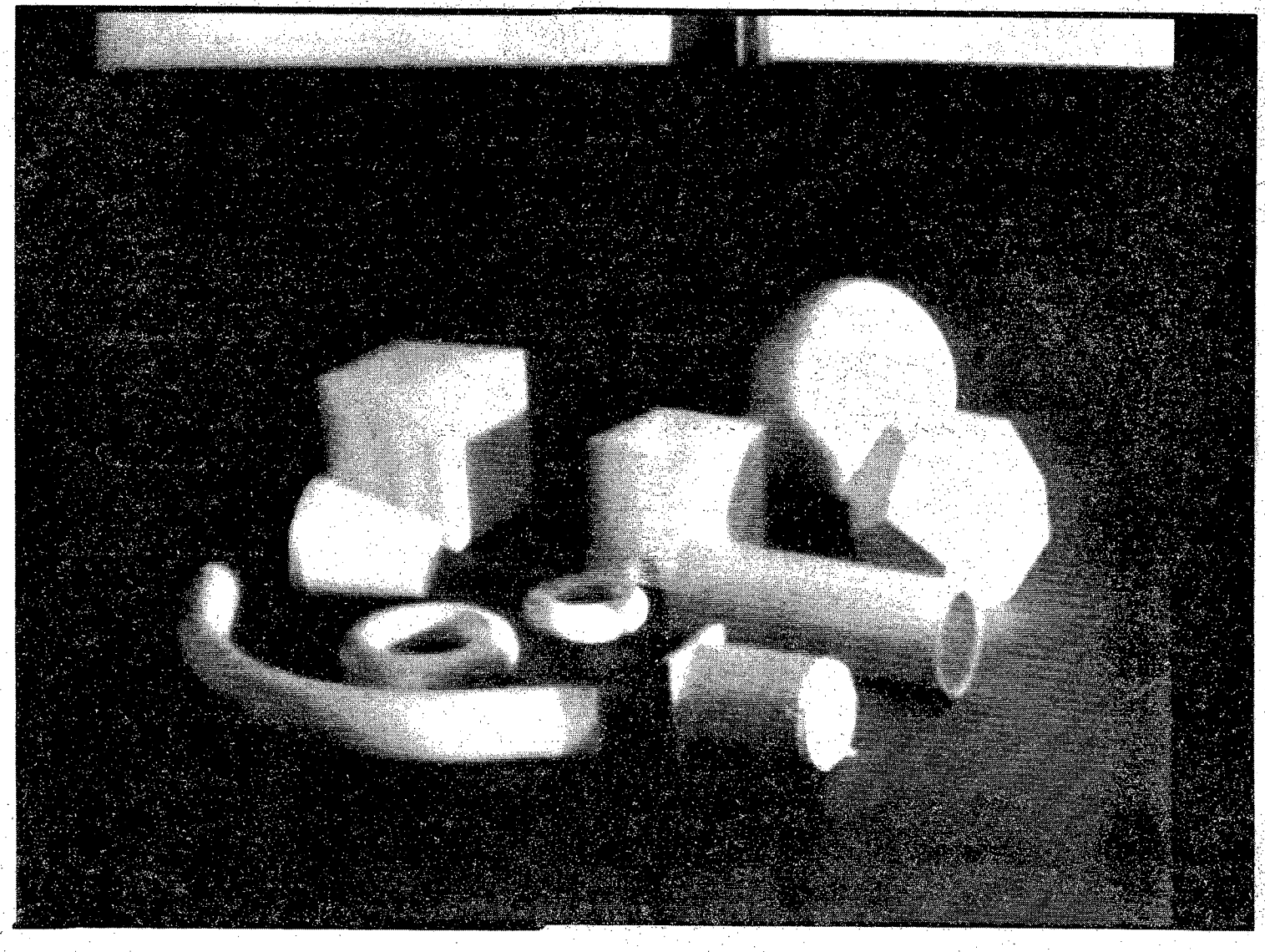


Fig. 2 Reflectance Image of a Scene

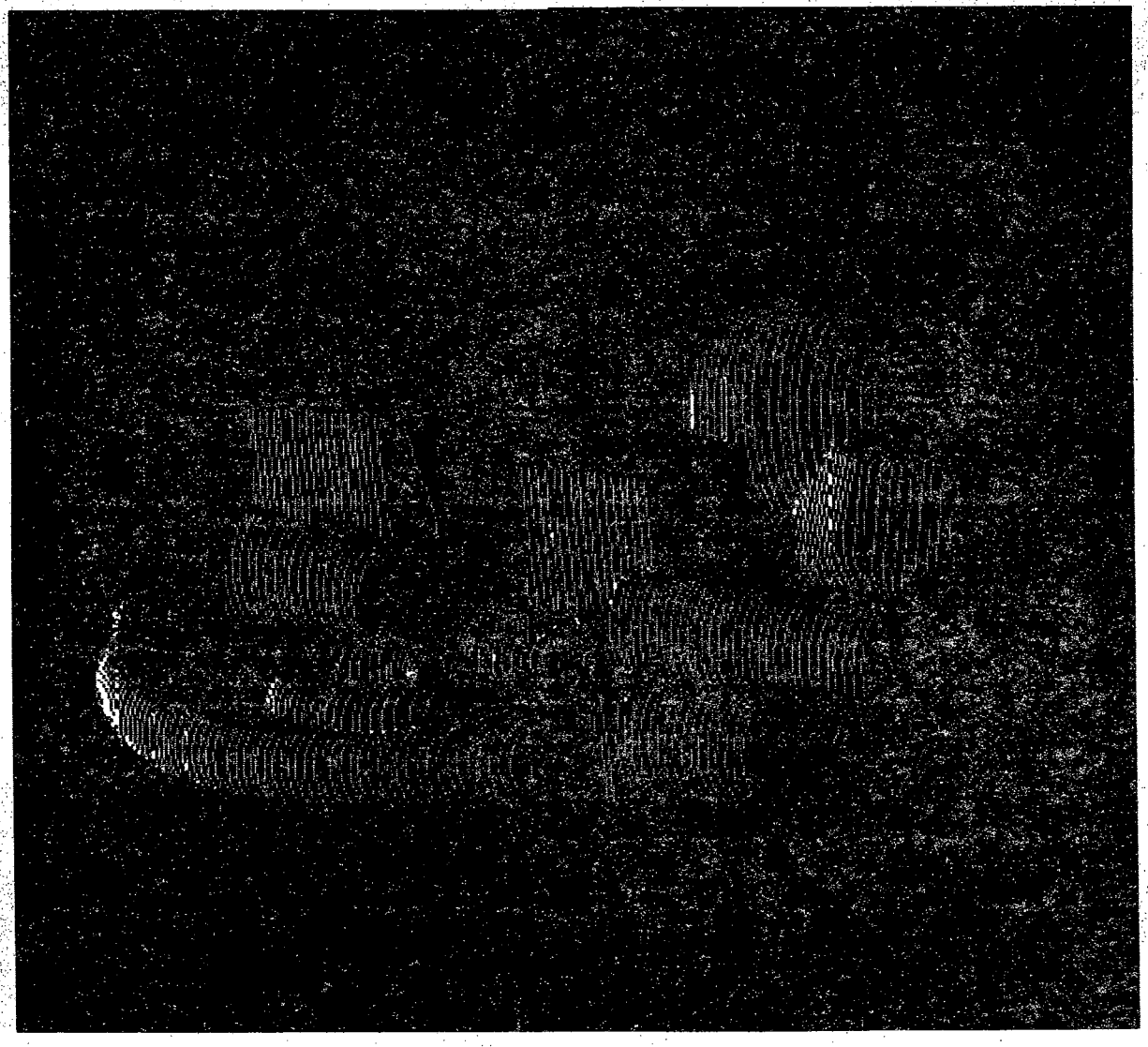


Fig. 3 Split Projected Image of Fig. 2

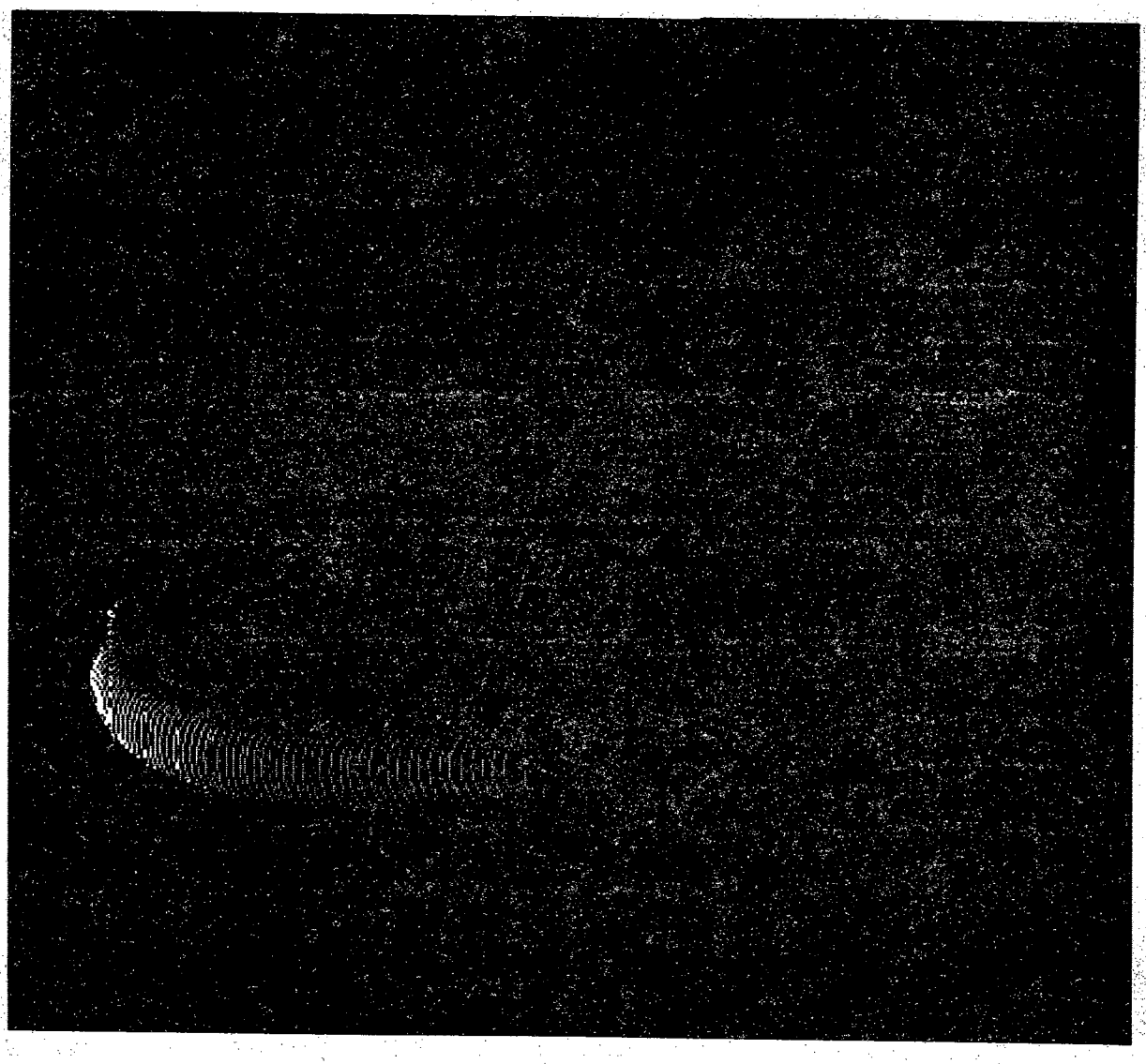


Fig. 4 Segmented C-curved Cylinder

****

PART IV

Management

Chapter 12

An Expert System For New Product Evaluation in Small Companies

Z. Xu and Q. Xue

AN EXPERT SYSTEM FOR NEW PRODUCT EVALUATION IN SMALL COMPANIES

ZHIWEI XU and QING XUE
School of Electrical Engineering
Purdue University

1.INTRODUCTION

The marketing is of vital importance for business companies. This problem is also very complex , even for small companies. Economist have been continuously tried to help the managers with sophisticated mathematical models , but the applications are , in most cases , unsatisfactory . And when economist , despite their painstaking effort , fail to even build a basic model , an expert could control the company's operation successfully and quite at ease . A big difference between the work styles of an economist and a manager can be observed . While the former searches for accuracy with too many unavoidable assumptions and simplifications , a manager masters the whole situation by making decisions with uncertainties form unreliable data through experience-base reasoning or common-sense reasoning. This suggests that marketing may be a fruitful area where a knowledge based expert system can simulate the real situation quite well.

Considering the many unsatisfactory applications of AI to economics, Zadeh [3] commented that the traditional two-valued logic is not suited , and he suggests that fuzzy-logic may supply with a better foundation. After an analysis of a number of

marketing strategies adopted by small business manages , we feel Zadeh's opinion does cope with the real situation . Usually the basic data are fuzzy , the knowledge and experience are fuzzy , the manager's deduction is fuzzy , even the decisions may sometimes be fuzzy . This is exempted by the following deduction which a small company's manager may make when he is considering if a new product should be developed and marketed :

" I know that this product is distinctively new , so customers will surely like it . The cost may be a little high , but affordable for the middle class families. And people are always after new things and they are likely to buy the good they want , even the price is higher than that of an old one . The demand for similar products varies seasonly , so large companies would unlikely be interested in it . Conclusion : the market prediction seems good ." Obviously , it is very difficult , if not impossible , for any economist to formulate the above deduction , let alone to implement it efficiently .

Our purpose in this project is twofold . Firstly , we would like to show that an expert system based on fuzzy logic may be constructed to simulate the real marketing strategies adopted by a small business' manager , thus to help the manager in decision-making . Secondly , we are to locate and classify the problems one would encounter when he is trying to build an expert system for marketing or business management .

We have focused our study on the new product marketability problem for small companies , i.e , deciding if a new product should be developed and put into market . This problem is chosen primarily because of its sufficient significance . As the saying goes ,

"new product is the fresh blood of any enterprise ." In fact , the success and failure of many companies are directly related to the new products they introduced . This problem is also very difficult . To our knowledge , no mathematical methods at all have been successfully applied to it .

We use questionare as the major tool for knowledge acquisition . The problem and relevant knowledge are analysed in chapter 2 . Since both authors are laymen in economics and management , it takes most of the research time to collect enough information and knowledge . Our preliminary finding in knowledge acquisition is that although management procedures of small companies differ econometrically , they are logically common to all companies . That's why we choose fuzzy logic as the foundation of our system .

Chapter 3 formulates the management strategies used by small companies with an abstraction method and a fuzzy logic model . Besides the capability of handling uncertainties , our Kleene-algebra based model has the advantage of minimizing the knowledge base , which seems an open problem in many expert systems .

Chapter 4 describes the architecture of our system . Included are an interpretation subsystem , which interfaces the user with the system ; a static database ; a dynamic database ; and an inference program .

Due to the short research time , our results are quite preliminary . In chapter 5 we discuss our result and suggest some guidelines for further research .

2. THE PROBLEM AND KNOWLEDGE

When a product design is submitted to a company as a candidate of its new products , it must pass the following five phases before serving the customers [11] [17]:

1. Screening . The design is evaluated by a number of experts subject to the resources and goals of the company .

2. Business Analysis . Current and future market situations are analysed and predicated to see if the product is profitability marketable .

3. Product Development

4. Test Marketing

5. Commercialization

Statistics show that about 80%-90% of new product designs failed to pass the five stages , where 70%-80% were rejected at the first two stages . For small companies , the figures are still higher . Because of its limited resources , a small company usually avoids developing a new product unless its profitability is very strong and almost definitely clear . This indicates the importance of Screening and Business Analysis .

To build an expert system to help the manager of a small company in Screening and Business Analysis ,we must first (at least roughly) master the relevant problems and knowledge . We have interviewed with a number of students and professors in economics and management and read lot of textbooks and journal papers . Our approach to knowledge acquisition is chiefly questionare . The following questions are iteratively asked to an expert:

What is the problem?

What is the relevant factors?

What relationship do they have?

What kind of data do they have?

How do you collect data?

How do you solve the problem?

After a careful analysis and systemization of the answers we get , we have the following findings:

The goal: Determine if a new product should be developed .

The factors: There are 18 factors , abbreviated as (1) equipment , (2) patent , (3) time , (4) loaning , (5) salary , (6) distinctiveness , (7) necessity , (8) appreciability , (9) image , (10) preference , (11) income , (12) relationship , (13) early stage , (14) large company , (15) variation , (16) competition-conscious, (17) explicit competitor , (18) potential competitor .

Only some of their meanings are explained below , due to the page limitation .

(2) patent : The money invested on patenting will be large.

(9) image : The company's image has been good in recent years.

(15) competition-conscious : The design of the new product has considered the explicit competitions and has supplied certain features that will enable the new product to compete quite well in the future market , if there is no potential competitor .

The relationship between these factors are quite complicated . A detailed representation is discussed in the following chapter .

Data of these factors are all fuzzy . Even if they are in numerical forms when they are reported , a manager tends to transform them into linguistic variables in his reasoning. The data are collected with different costs . some data are very handy , such as if a company would insist on its economic independence . It is more difficult to get other data . For instance , it will take a lot of time and money to get the information concerning various market situations .

We have encountered many difficulties in sorting out how an expert solves the problem . After a careful study of various methods , we find that there is a common procedure for all companies . Experts use a shame similar to the abstraction concept in the expert system studies . They divide the problem into a number of factors , each factor is then determined in terms of lower- level factors. This procedure keeps going until each factor becomes a primary one which is obtained by information collection . The trace of the procedure does not constitute a tree , though . In our study , the goal of deciding if a new product should be developed is determined by four factors : the cost , the price , the market , and the competitions. Each of these factors is then evaluated in terms of factors of lower levels . For instance , the cost is determined by the equipment , the patent , the developing and manufacturing time , the intency of raising a loan , and the salary .

3. THE MODEL - ABSTRACTION PLUS FUZZY LOGIC

In this chapter , we first review and analyse the decision procedure which appears in the real world and then formulate it with abstraction and fuzzy logic .

In evaluating a new product , the manager of a small company needs to draw his conclusions from a number of factors . These factors are divided into two groups : the controllable and the uncontrollable . Examples of controllable factors include the intency of raising a load , the necessity of increasing salary , and the pricing policy , etc , which the manager has some freedom to specify . Uncontrollable factors include such as the market factors which are determined by the environment and the competition factors . The manager intends to decide not only if a new product should be developed , but also to choose an optimal combination of the controllable factors so that the new product can achieve the greatest profitability .

The manager first collects all the data for each factor which is usually given with upper and lower bounds . If the number of factor is small , he considers every combination of the factors to see the profitability in each case . He then chooses an optimal combination with the largest profitability . Probability methods may be used to determine the risk and uncertainty in each case . This approach ,although very simple and straightforward , is unpractical when a large number of factors need to be considered . The total number of combinations of n factor is $O(2^n)$. Thus when $n > 20$, this brute-force method will fail , even with the help of a computer .

The evaluation of new product usually involves many (tens or hundreds) factors . To cope with the complexity , managers of small companies use a scheme similar to the abstraction method in the expert system study [2] . For example , the profitability of the new product is divided into four factors : the cost , the price , the market , and competition . Each factor is then divided into small factors . For instance , the competition must consider the explicit competitors , the potential competitors , the large companies , and the competition-consciousness of the new product . The variable range of a factor is now determined by its son factors , not necessarily infered from the primary data . If every factor has no more than k son factors , the computational complexity of the method by abstracting is about $O(2^k.n)$, where $k << n$.

Business analysis is usually very costly in both time and money. It may take several months and a lot of money to get such information as the customer's preference , the average income of potential customers , the strategy of a competitor , and so on . Small companies often cannot afford to do information collection . Managers are required to make decision on available data which may be incomplete .

Another feature of the decision procedure is that almost all data are fuzzy ones . Even if some data may be reported to the manager in numerical forms , the manager tends to transform them into fuzzy linguistic variable in his reasoning .

After an analysis of the decision-making strategies appearing in real cases , we propose the following three procedures as a model . Procedure 1 constructs a factor dependence graph . Each node

represents a factor , from which a number of arcs point to its son factors . There is a relation associated with each factor to its son factors , which can be AND , OR , NOT , or a power function . A weight is assigned to each node representing the cost for obtaining the value of the factor . Procedure 1 follows :

Procedure 1. Obtain the factor-dependence graph

Step 1. Determine the goal and the primary factors .

Step 2. The goal is divided into a number of factors ; each factor is then divided into subfactors . This procedure keeps going until all the lowest level factors become primary ones .

Step 3. The manager estimates the upper and lower bounds of each primary factor . This is only a subjective estimation and is easily done .

Step 4. Find the relation of a factor to its son factors by compoting the value for various combination or by other means . This relation is then represented as a fuzzy switching function . The relation of the goal to the primary factors can now be represented as a multi-level acylic graph , where each node represent a factor and its relation to its son factors could be AND , OR , NOT , or some power function .

Step 5. Assign a cost to each primary factor and then obtain the cost of each other factor by summing the cost of all its son factors .

After using procedure 1 , the goal is represented as a fuzzy function of the factors . Four types of operators appear . They are defined as follows .

OR : $x + y = \max(x, y)$

AND : $x \cdot y = \min(x, y)$

NOT : $\bar{x} = 1 - x$

POWER : $x^{k} = x^{k}$

Ignoring the POWER operator for the time being . The other three operators , together with all the factors , form a set of fuzzy switching functions , which is also a Kleene algebra [20]. A nice property of Kleene's algebra is that it can be simplified (or minimized) [21] [22]. We may note that in fact , each node of the factor-dependence graph (except the primary factors) corresponds to a rule in the expert system to be implemented . Thus the Kleene algebra-based model can help reduce the size of the knowledge base . We use procedure 2 to simplify the factor-dependence graph , which can be any minimization procedure in the literature . We use POWER operators only for primary factors , in order to maintain axioms of the Kleene algebra model .

After abstraction and simplification, a factor-dependence graph considering the 18 factors is shown in Fig. 1 for a certain product . This constitutes the static database of the expert system . In oder to simplify the structure of each rule , we have assumed that each factor can have only one of the four relationship (i.e , OR , AND , NOT , POWER) to its son factors .

After the factor -dependence graph is constructed , the system uses Procedure 3 to help the manager in decision-making . This procedure uses a backtracing shame similar to that in MYCIN [4]. Begining from the goal , the procedure tries to find a combination of primary factors which favor the new product with an uncertainty

greater than a predetermined threshold value . Two things are emphasized . Firstly , if the logic value of a factor can be infered , then it should not be asked . Secondly , if we must ask for a number of factors , then only the factor with the smallest cost should be asked for . Procedure 3 is the procedure Value (s) in our system.

4. THE SYSTEM STRUCTURE

Our expert system consists of four components : an interpretation subsystem , a static database , a dynamic database , and an inference procedure .

The interpretation subsystem is the interface between the user and the system. Whenever the value of a primary factor can not be infered , the interpretation subsystem would ask the user for the value of this factor . The question is in the form of multiple choice . For example , when the system is put on , it will ask the user :

"The cost of patent is -------"

a. definitely high

b. extremely high

c. very high

d. quite high

e. a-little high

f. moderate

g. a-little low

h. quite low

i. very low

j. extremely low

k. definitely low

"please choose one letter filling the blank"

When the value of the goal (which is represented as S) is obtained , the interpretation subsystem would print out the result in a sentence . For example , when the cost is very high and the intency of raising a loan is very weak , the small company will not be able to get enough fund for this new product development , thus the system will print out the following answer :

"It is very unlikely that the company should develop this new product ."

The static database stores the knowledge colleted . In our system , it stores the factor-dependence graph of Fig. 1. In other words , it must contains all nodes , one node just like one rule in rule-based system . According to this kind of data structure , frame is used . The whole static database is a frame , one node is a slot , which has three faceties , operator (op) , #sons (nu) , sons (son) . That is ,

(static database (node (operator) (#sons) (sons))) .

The dynamic database maintains all necessary information for Procedure 3 when it is used in searching for a favorite combination of primary factors . dynamic database is a list (dl) , which contains the nodes computed and it value is stored .

Although our system is essential a rule-based system . We use a recurrence procedure (procedure value (s)) for fuzzy reasoning . Begining from the goal (node s) , the procedure determines the value of a node from the values of its son factors with the least cost .

Some example follow

There are three kinds of solution in this system.

(1) value(s) > THVU (0.7) , Solution : The design is approved .

(2) value(s) < THVL (0.3) , Solution : The design is rejected .

(3) THVL < value(s) <THVU , Solution : The design is conditionally approved .

Example 1

Suppose a new design of product does not require patent and increasing salary . The cost of new equipment is extremely low and the intency of loaning is extremely low . Anyway , the cost is low . The independence of the company is extremely strong and the necessity for customers is very high . The expert system gives answer that the design is approved . The dialogue between the expert system and the user will be showed in Appendix .

Example 2 :

For some new design of product , The information about cost is just like above The example , that is , the patent and increasing salary are not required . And the cost of new equipment is extremely low . The intency of loading is extremely low . But the

independence of company is very weak , and the necessity for customers is a little high . According to above situation expert system can not make decision . So it must ask other questions . When it knows the newness is very distinctive and quite appreciable , the answer will be :" The design is approved." In this time , the fuzzy value is 0.7 . The dialogue see Appendix .

Example 3 :

Through interacting with the user (see Appendix) , the expert system know some design of product requires patent, new equipment , and increasing salary . The cost of new equipment is a little high , the intency of loaning and increasing salary is moderate . In general speaking , the cost is not very high . The independence of company is very strong and necessity for customers is very high . Until now no decision can be made , expert system continues to ask questions . More facts will come : The competition-consciousness is a little high . The relationship with big company is a little loose , it is very true that products of big company are in early stage . The competition ability of pertential competitor is moderate, the competition ability of explicit competitor is extremely low . Finally the solution can be given "The design is conditionally approved." (The fuzzy value is 0.4)

6. DISCUSSIONS

In this project , we design an expert system which can help a manager of a small company in evaluating new product designs . The system has the following features :

1. The system provides a fuzzy reasoning environment . A user supplies the system with fuzzy linguistic variables ; the system inferences by evaluating fuzzy switching functions ; and the result is reported in the form of a sentence with fuzzy linguistic variables.

2. The value of a factor will not be asked for unless it can not be inferenced.

3. When the system asks a question , it must be a question which the user can answer with the least amount of cost and which will add information to the system .

After the system is programmed , we have invited several students in the Department of Economics to test it . Their responses suggest that as a preliminary system , it is satisfactory , but quantitative features should be added to it so that the system can give some quantitative results (maybe with uncertainties) from quantitative inputs . For example , they would think the system more attractive if it could give such results as "it is very true that the new product will bring the company $10000 worth of profit ."

We have located some AI problems in the new product evaluation strategy :

1. As the number of factors becomes large , the brute-force method of enumerating all combinations is unpractical . Furthermore , since a result is obtainable only after considering all primary factors , partitioning and early pruning [2]do not help much. Our abstraction procedure (procedure 1) lacks rigorness of mathematics . It's desirable to develop a methodology which is mathematically valid and sound . We have recently noded that the grey system theory proposed in Middle China Institute of

Technology has similar features as the abstraction methodology . We hope such a theory would provide a sound , theoretical model .

2. Since a manager of a small business tends to solve his problem with fuzzy reasoning , the expert system imitating his method must have a fuzzy reasoning subsystem .

3. Because small companies can not afford costly market information collection , questions should not be asked if they can be infered from the available data . If it is necessary to ask question , then ask those which can be answered with the least amount of cost .

Suggestions for Future Research .

In what follows , we intend to suggest some guidelines for future research and to show what our system can do ultimately .

1. Expandability . The economics students we invited to test the system suspected that our system did not contain all the relevant factors . We asked them to name some . It happened that all the factors they suggested are considered in our system . But altogether we only considers 18 factors , which should be expanded and refined . Ultimately ,about 100 factors are unavoidable . We should like to mention , however , that the expandability of our system is quite good . In economics , a factor usually connected directly with only a few (say k) son factors . By using the abstraction methodology , we can establish the factor-dependence graph in $O(2^K.n)$ time , where n is the total number of factors and k is usually less than 10

2. Abstraction. A rigorous mathematical treatment of the abstraction methodology is needed. This is the most important and most difficult problem . As we suggested earlier , the grey system theory may help . Moreover , results in microeconomics or econometrics should be merged into thè system so that not only qualitative but also quantitative data can be processed .

3. Interpretation. The economics students observed that our system is too mastering ,in the sense that the user can only passively answer questions . They suggested that it would be preferable if it would let the user first input some readily available information . This is quite to our surprise , for we think it an advantage . However , their suggestion does indicate that our interpretation subsystem should be cleverer . Some limited capability of natural language understanding should be included .

4. Explanation and Modification . We didn't include an explanation function in our system . For real applications , this function is necessary . A manager would like to know how an approval is derived . And a new-product designer would not be convinced unless he is told the detailed reasons why his design is rejected . An explanation function can easily associated with the system , because of the tracing function of lisp .

A more interesting feature our system can achieve is the function of modification . When the design of a new product is rejected , the system will point out what's wrong with the design and suggest directions for improvement . Unlike medicine applications , where the effect of some therapy can be observed only after a relatively long time , the effects of modification in a new-product design can

be predicated almost instantly by the computer . For example , a design may be rejected because of the following defficiencies : (a). long development time (b) costly new equipment . It may happen that the long development time and the costly new equipment are necessary to develop a product with very high quality so that the product can have a high competition ability in the future market . But the market prediction may be extremely good , so the quality need not be too high . The expert system would consider the cases when the development time is shortened and the cost of equipment is reduced . Then it may suggest that "if the development time can be shortened to 'very short' and the cost of equipment can be reduced to 'a little low ' , then the design will be conditionally approved ."

Reference

[1] Richard O.Duda and Edward H.Shortliffe, Expert Systems Research, SCIENCE,April 1983

[2] F.Hages-Rith,D.Watermon and D.Lenat,eds,Building Expert systems System, ADDISERN-WESLEY,1983

[3] "Coping with the Imprecision of the Real World" by L.A.Zadeh from Communications of the ACM. Volume 27, Number 4, April 1984

[4] J. Cendrowska and M.A.bramer,A rational reconstruction of the MYICN consultation system, Int. J. Man-Machine Studies (1984)20, 229-317

[5] MITSURUISHIZUKA,K.S.FU,JAMES T.P.YAO, Inference procedures under uncertainty for the problem-Redution Method.

[6] Ron Sauers and Rick Waclsh, On the Requirement of Future expert systems

[7] Kleene,S.C.(1952), Introduction to Matamathematics,North-Holland Pub, pp.332-340

[8] L.A.Zadeh, the concept of a linquistic variable and its application to approximate reasoning. American Elsevier Publishing Company, Inc.1975

[9] William H. Brannen, Successful Marketing for Your Small Business,1978 by prentice-hall, Inc.

[10] Roland W. scholz. Decision making under uncertainty, Elsevier science pulishers. B.U.1983

[11] Thomas L. Berg and Abe Shuchman, Product strategy and management, Holt,Rinchart and winston, Inc.1963

[12] Edmin Mansfield, Micro-economics theory and applications. W.W.Norton & Company, Inc.1979

[13] Lee E. Preston , Managing the independent business , Prentic-Hall , Inc.

[14] Robert C. Haring , Marketing of mechanical household refrigerators , Montana state unversity printing department.

[15] Taylor. shaw , Marking - an integrated , anlytical approach

[16] James L. Heskett. Student resource manual to accompany marketing, Macmillian publishing Co. ,Inc. New York.

[17] Stanton , Fundamental of marketing , Mcgraw-Hill Book Company, New York.

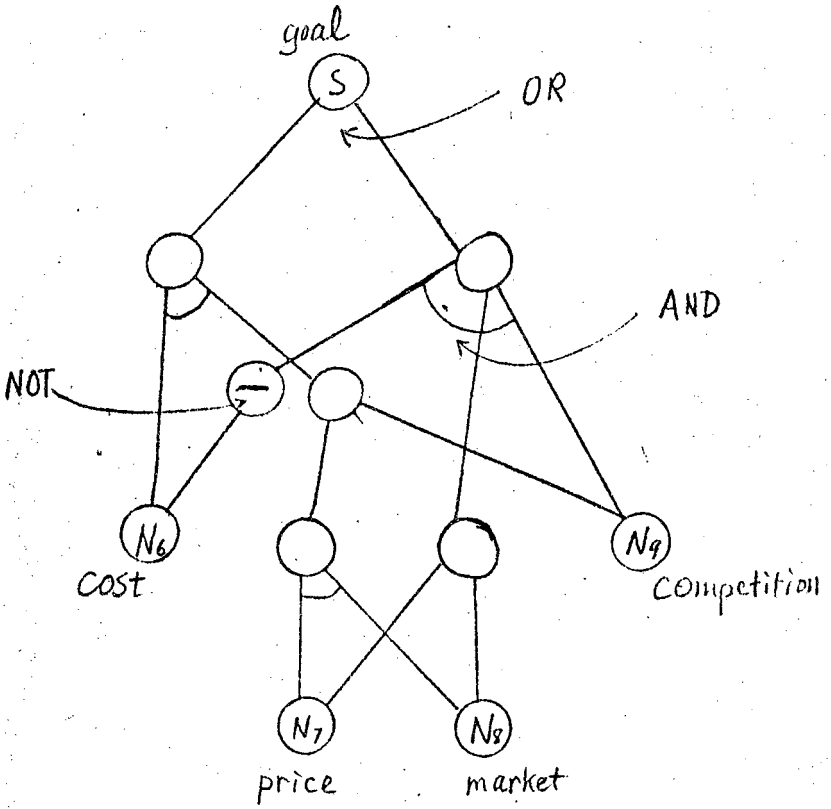
[18] Edwin C. Greif , Basic problems in marketing management, Wadsworth publishing company , Inc. Belmont , California.

[19] Robert M. Fulme and Roger A. Strang, Exploring the new marketing , Macmillan publishing Co. ,Inc. New York.

[20] M.Mukaidono , "A set of independent and complete axioms for a fuzzy algebra ," in Proc. 11th IEEE Intl. Symp. on Multiple-valued logic , May 1981

[21] A. Kandel and J.H.Francioni, "On the properties and application of fuzzy-valued switching functions," IEEE Trans Compute, Vol c-29, 1980

[22] Z. Xu, "Multivalued logic and fuzzy logic -their relationship, minimization, and application to fault diagnosis," IEEE Trans. Comput Vol.c-33, No, pp679-681, July 1984.



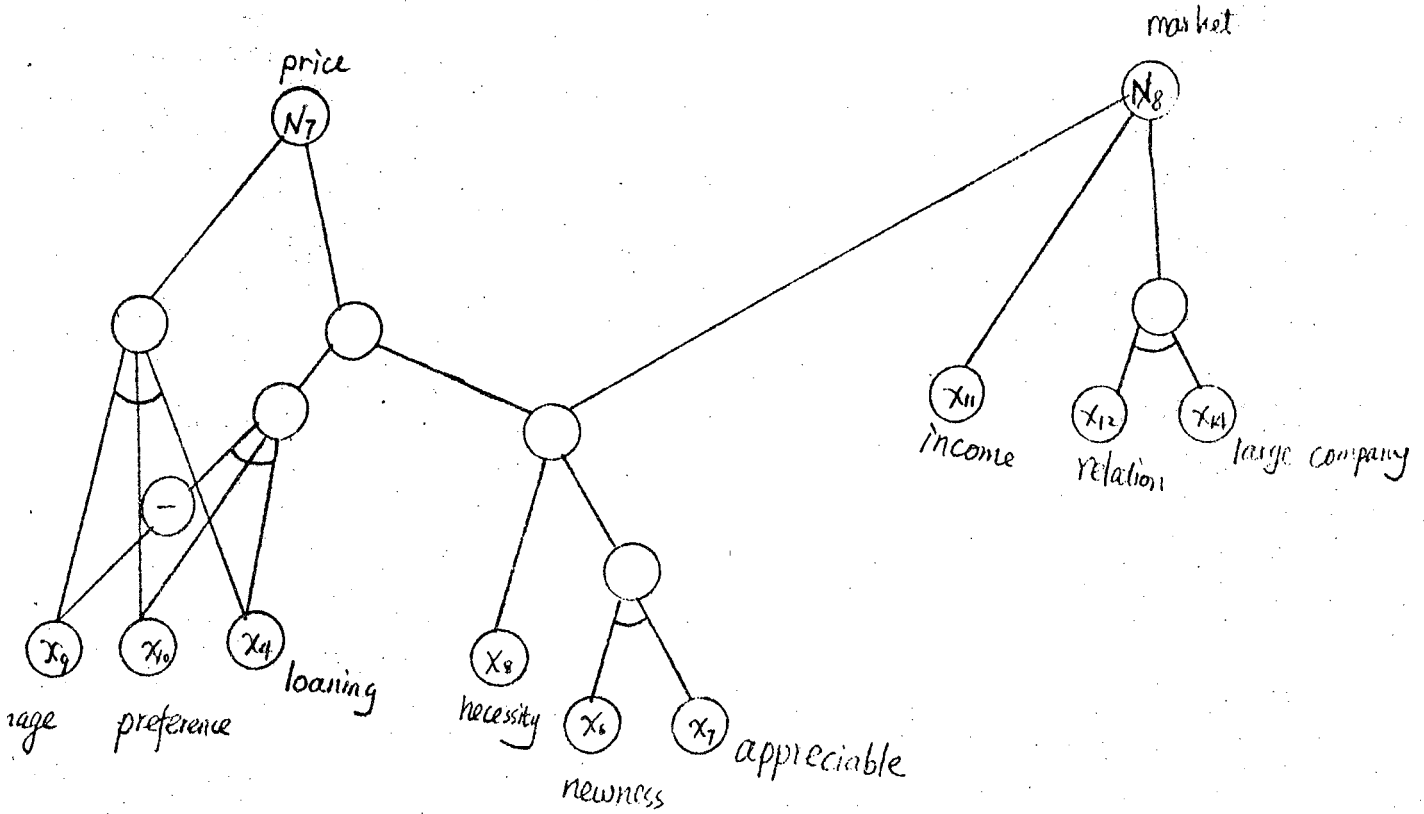
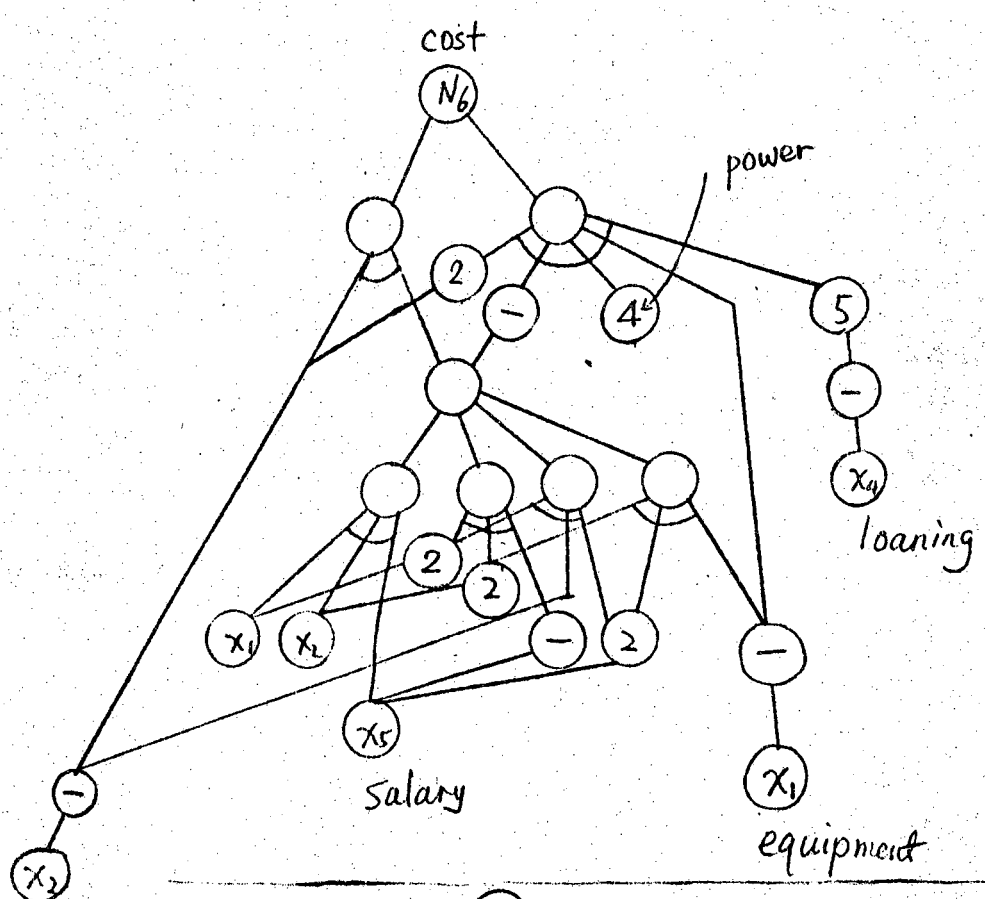


Fig. 1-(a) Factor - Dependence Graph



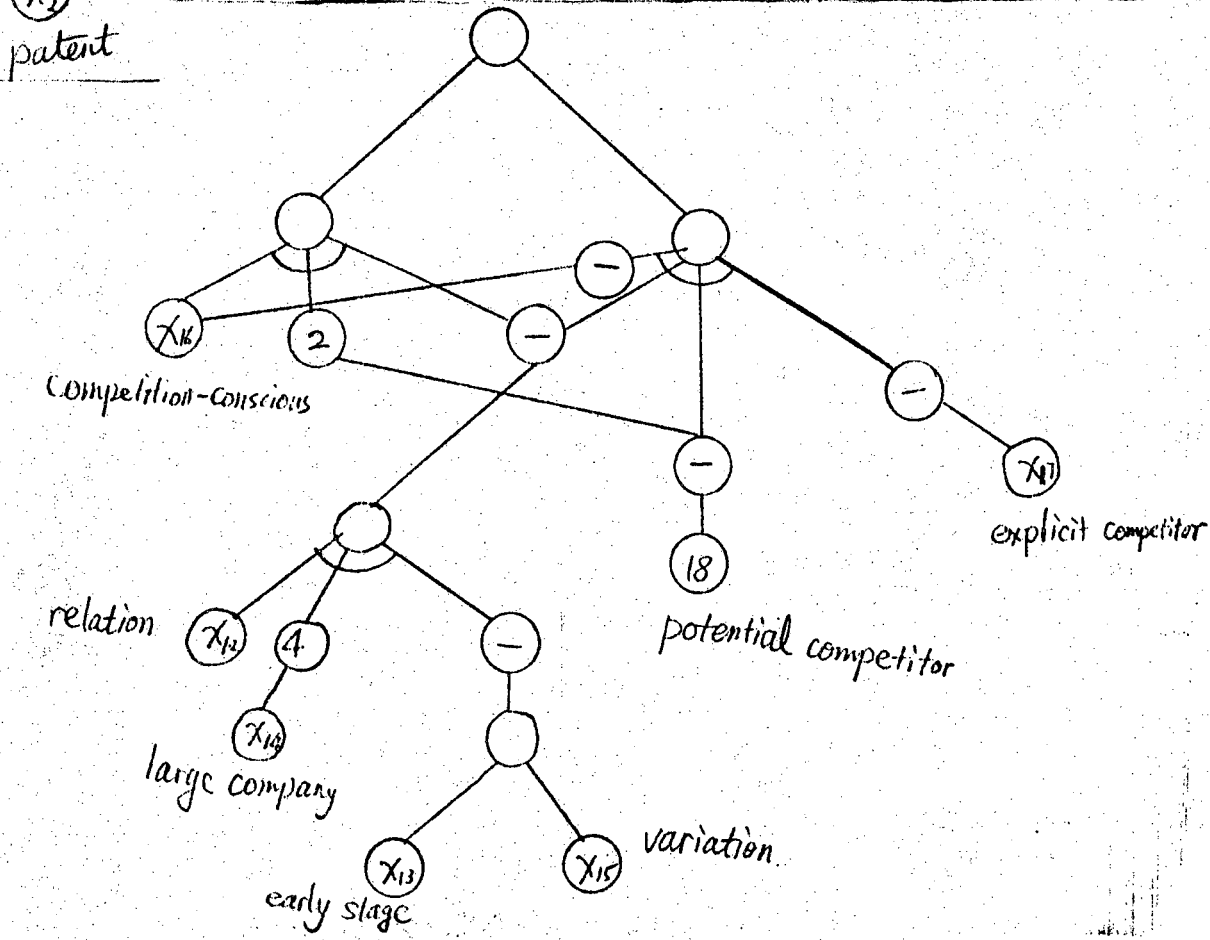


Fig. 1-(b) Factor-Dependence Graph

Appendix
****

Example 1

-> (main 's)

(The cost of patent is -----)
a.definitely high
b.extremely high
c.very high
d.quite high
e.a-little high
f.moderate
g.a-little low
h.quite low
i.very low
j.extremely low
k.definitely low
(please choosing one letter fillins the blank)k

(The cost of new equipment is -----)
a.definitely high
b.extremely high
c.very high
d.quite high
e.a-little high
f.moderate
g.a-little low
h.quite low
i.very low
j.extremely low
k.definitely low
(please choosing one letter fillins the blank)j

(The intency of loanins is -----)
a.definitely strons
b.extremely strons
c.very strons
d.quite strons
e.a-little strons
f.moderate
g.a-little weak

h.quite weak
i.very weak
j.extremely weak
k.definitely weak
(please choosing one letter filling the blank)j

(The necessity of increasing salary is -----)
a.definitely high
b.extremely high
c.very high
d.quite high
e.a-little high
f.moderate
g.a-little low
h.quite low
i.very low
j.extremely low
k.definitely low
(please choosing one letter filling the blank)k

(The company 's preferece of economic independence is -----)
a.definitely weak
b.extremely weak
c.very weak
d.quite weak
e.a-little weak
f.moderate
g.a-little strong
h.quite strong
i.very strong
j.extremely strong
k.definitely strong
(please choosing one letter filling the blank)j

(The product is for customers-----)
a.definitely necessary
b.extremely necessary
c.very necessary
d.quite necessary
e.a-little necessary
f.moderate
g.a-little unnecessary
h.quite unnecessary
i.very unnecessary
j.extremely unnecessary
k.definitely unnecessary
(please choosing one letter filling the blank)c
(The design is approved)

****

Example 2

-> (main 's)

(The cost of patent is -----)
a.definitely high
b.extremely high
c.very high
d.quite high
e.a-little high
f.moderate
g.a-little low
h.quite low
i.very low
j.extremely low
k.definitely low
(please choosing one letter fillins the blank)k

(The cost of new equipment is -----)
a.definitely high
b.extremely high
c.very high
d.quite high
e.a-little high
f.moderate
g.a-little low
h.quite low
i.very low
j.extremely low
k.definitely low
(please choosing one letter fillins the blank)j

(The intency of loanins is -----)
a.definitely strons
b.extremely strons
c.very strons
d.quite strons
e.a-little strons
f.moderate
g.a-little weak
h.quite weak
i.very weak
j.extremely weak
k.definitely weak
(please choosing one letter fillins the blank)j

(The necessity of increasing salary is -----)
a.definitely high
b.extremely high
c.very high
d.quite high
e.a-little high
f.moderate
g.a-little low
h.quite low
i.very low
j.extremely low
k.definitely low
(please choosing one letter fillins the blank)k

(The company 's preferece of economic independence is -----)
a.definitely weak
b.extremely weak
c.very weak
d.quite weak
e.a-little weak
f.moderate
g.a-little strong
h.quite strong
i.very strong
j.extremely strong
k.definitely strong
(please choosing one letter filling the blank)d

(The product is for customers-----)
a.definitely necessary
b.extremely necessary
c.very necessary
d.quite necessary
e.a-little necessary
f.moderate
g.a-little unnecessary
h.quite unnecessary
i.very unnecessary
j.extremely unnecessary
k.definitely unnecessary
(please choosing one letter filling the blank)e

(The newnes of the product is -----)
a.definitely distinctive
b.extremely distinctive
c.very distinctive
d.quite distinctive
e.a-little distinctive
f.moderate
g.a-little non-distinctive
h.quite non-distinctive
i.very non-distinctive
j.extremely non-distinctive
k.definitely non-distinctive
(please choosing one letter filling the blank)c

(The newness of the product is -----)
a.definitely appreciable
b.extremely appreciable
c.very appreciable
d.quite appreciable
e.a-little appreciable
f.moderate
g.a-little non-appreciable
h.quite non-appreciable
i.very non-appreciable
j.extremely non-appreciable
k.definitely non-appreciable
(please choosing one letter filling the blank)d
(The design is approved)

****

Example 3

(The cost of patent is -----)
a.definitely high
b.extremely high
c.very high
d.quite high
e.a-little high
f.moderate
g.a-little low
h.quite low
i.very low
j.extremely low
k.definitely low
(Please choosing one letter fillins the blank)e

(The cost of new equipment is -----)
a.definitely high
b.extremely high
c.very high
d.quite high
e.a-little high
f.moderate
g.a-little low
h.quite low
i.very low
j.extremely low
k.definitely low
(Please choosing one letter fillins the blank)e

(The necessity of increasing salary is -----)
a.definitely high
b.extremely high
c.very high
d.quite high
e.a-little high
f.moderate
g.a-little low
h.quite low
i.very low
j.extremely low
k.definitely low
(Please choosing one letter fillins the blank)f

(The intency of loaning is -----)
a.definitely strong
b.extremely strong
c.very strong
d.quite strong
e.a-little strong
f.moderate
g.a-little weak

h.quite weak
i.very weak
j.extremely weak
k.definitely weak
(please choosing one letter fillins the blank)f

(The company 's preferece of economic independence is -----)
a.definitely weak
b.extremely weak
c.very weak
d.quite weak
e.a-little weak
f.moderate
g.a-little strons
h.quite strons
i.very strons
j.extremely strons
k.definitely strons
(please choosing one letter fillins the blank)i

(The product is for customers-----)
a.definitely necessary
b.extremely necessary
c.very necessary
d.quite necessary
e.a-little necessary
f.moderate
g.a-little unnecessary
h.quite unnecessary
i.very unnecessary
j.extremely unnecessary
k.definitely unnecessary
(please choosing one letter fillins the blank)c

(The desisn of the product is-----)
a.definitely competition-concious
b.extremely competition-concious
c.very competition-concious
d.quite competition-concious
e.a-little competition-concious
f.moderate
g.a-little non-competition-concious
h.quite non-competition-concious
i.very non-competition-concious
j.extremely non-competition-concious
k.definitely non-competition-concious
(please choosing one letter fillins the blank)e

(The correlation is----- between the demand of the product in quest
a.definitely tisht
b.extremely tisht ion and that of a larse company 's product
c.very tisht
d.quite tisht
e.a-little tisht
f.moderate
g.a-little loose.

h.quite loose
i.very loose
j.extremely loose
k.definitely loose
(Please choosing one letter fillins the blank)g

(It is----- that the production of Y is at the early stage)
a.definitely true
b.extremely true
c.very true
d.quite true
e.a-little true
f.moderate
g.a-little false
h.quite false
i.very false
j.extremely false
k.definitely false
(Please choosing one letter fillins the blank)d

(The potential competitors have a -----competibility)
a.definitely large
b.extremely large
c.very large
d.quite large
e.a-little large
f.moderate
g.a-little small
h.quite small
i.very small
j.extremely small
k.definitely small
(Please choosing one letter fillins the blank)f

(The explicit competitors have a -----competibility)
a.definitely large
b.extremely large
c.very large
d.quite large
e.a-little large
f.moderate
g.a-little small
h.quite small
i.very small
j.extremely small
k.definitely small
(Please choosing one letter fillins the blank)j
(The design is conditionally approved)

Chapter 13

Expert System for Inventory Models

K. Y. Tam and H. R. Rao

EXPERT SYSTEM FOR INVENTORY MODELS

K. Y. TAM
H. R. RAO

INTRODUCTION

Maintenance and control of inventory is one of the more important functions in a business environment today. Among others, increasing costs, supply/production uncertainities and flutuating demand dictate that invento is a must. The question of how much inventory to hold is an old one, that has interested both academicians and practitioners alike. As much, several inventory control models have been developed over the years to aid a decisi maker in the choice of an inventory policy relevant to his envisions. The numbers of such models are legion and each of these is valid only under certain assumptions or conditions.

Therefore, a computerised D.S.S. would go a long way in aiding a decision maker in the choice of an appropriate model. This is one such effo In this study, we use a production system called OPS5 to construct a set of expert rules, that eventually dictate the choice of a model. The mini D.S.S represented herein, is limited to about 8 deterministic inventory control models. In describing a model, the system relies on the decision maker to input information pertaining to his envisions, and provides him with formul for computing various parameter of interest (such as, optimum inventory ord quantity, optimum total cost,etc.), and thereby describe an inventory polic suitable to his envisions.

OPS5

The inventory decision support system is implemented in OPS5, which is a class of programming languages known as production systems. It is used primarily in the area of Artificial Intelligence,Expert Systems and Cognitive Psychology. Details of OPS5 will not be explored here. For detail information, readers should refer to the OPS5 Manual. In this section, we will generally discuss the architecture of OPS5, features of each structural component and the control mechanism of the entire system.

System Architecture

Bascially, OPS5 consists of three components,namely the productic memory,the working memory and the control system. In Artificial Intelli terminology,the working memory can be viewed as the knowledge base. The production memory together with the control system form the inference component of the whole system.

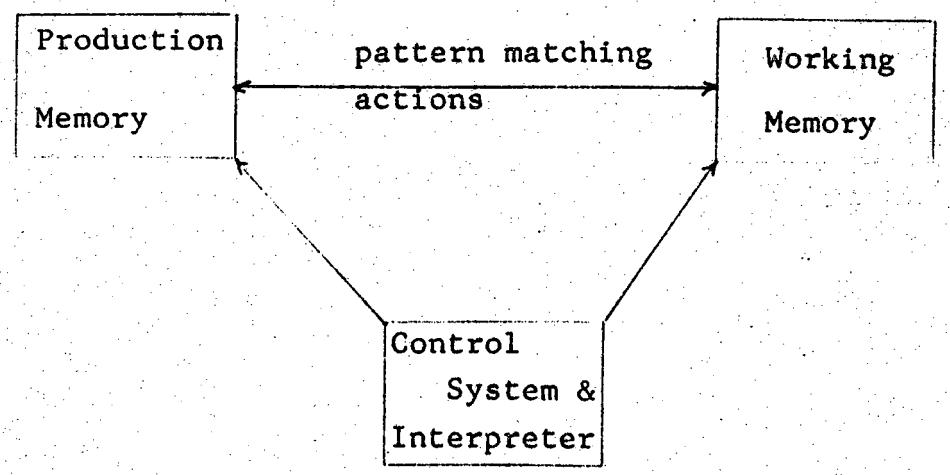
Production System

A production system is a program composed entirely of conditional statements called productions. And these productions are stored in a separate memory called production memory. A typical production is simila to a IF - THEN statement of conventional programming languages : a production that contains n conditions C1 through Cn and m actions A1 through Am means when working memory is such that C1 through Cn are true simultaneously then actions A1 through Am should be executed. The condit part of a production is called the LHS (Left Hand Side) and the action part is called the RHS (Right Hand Side). The LHS of a production consis of one or more pattern; ie one or more expressions that describe working memory elements. The pattern is considered satisfied in the sense that i it matches at least one element in the working memory. If all the patter in a production's LHS are satisfied then LHS is satisfied. The following is an example of a production in OPS5 format and its English explanation

```
( p find-color-block                 :  if there is a goal
    ( goal                           :  which is active
        ^status active               :  to find
        ^type find                   :  a block
        ^object block                :  of a certain color
        ^color <z>  )                :  And there is a block
    ( block                          :  of that color
        ^color <z>                   :
        ^name  block )               :
-->                                  :  Then
    ( make result                    : make an element
        ^pointer  block    )         :  to point the block
    ( modify 1                       : And change the goal
        ^status satisfied ))         :  making it satisfied
                                     :
```

OPS5 Production English Explanation

Example of a typical OPS5 production



Architecture of OPS5 System

Working Memory

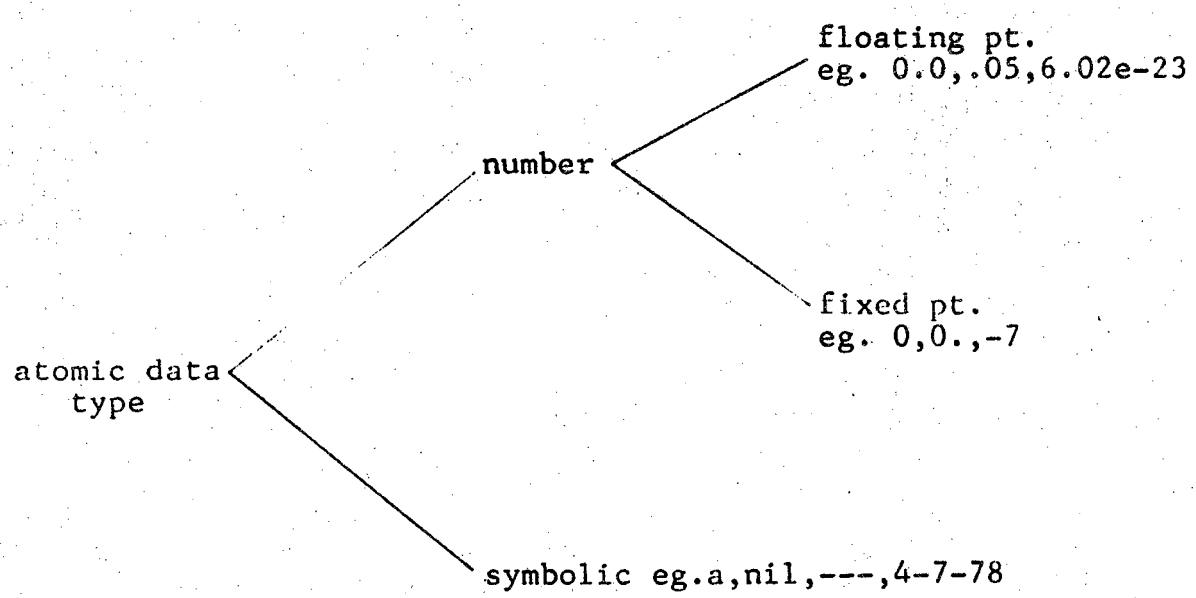
In OPS5, working memory is a set of ordered pairs (time tag, working memory element). A working memory element is a structural (usually a vector or record) of scalar values. The time tag is a unique numerical identifier that is supplied by the interpreter.OPS5, like most programming languages, provides better scalar (sometimes called atomic) data types and structural data types. The elements in working memory may not be scalars. Moreover, the number of elements in working memory varies dynamically at run time. Every element in working memory has an associated integer called the element's time tag. This integer indicates when the element was created or last modified. No two element have the same time tag. Time tags are used in conflict resolution, and they are designate elements by many of the facilities that communicate with the user. Patterns are abstract representations of working memory elements. Oneway a pattern can be an abstraction of a working memory elements is to contain fewer attributes and values than the element. Such a pattern will match any working memory element that contains the information in the pattern.

Control System

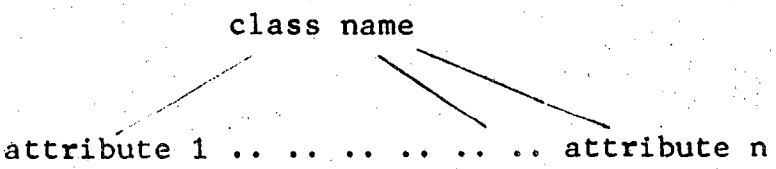
The control system executes a production system by performing a sequence of operation called the Recognize Act Cycle.

1. [Conflict Resolution] Select one production with a satisfied LHS. If no production have satisfied LHS return control to the user.
2. [Act] Perform the actions specified in the RHS of the selected production.
3. [Match] Evaluate the LHS of the productions to determine which are satisfied given the current contents of working memory.
4. If a halt was performed, return control to the user; otherwise go to step 1.

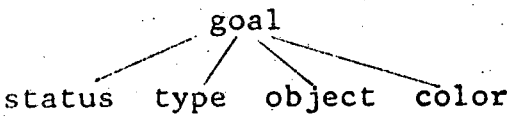
Working Memory Data Types



Scalar Data Type

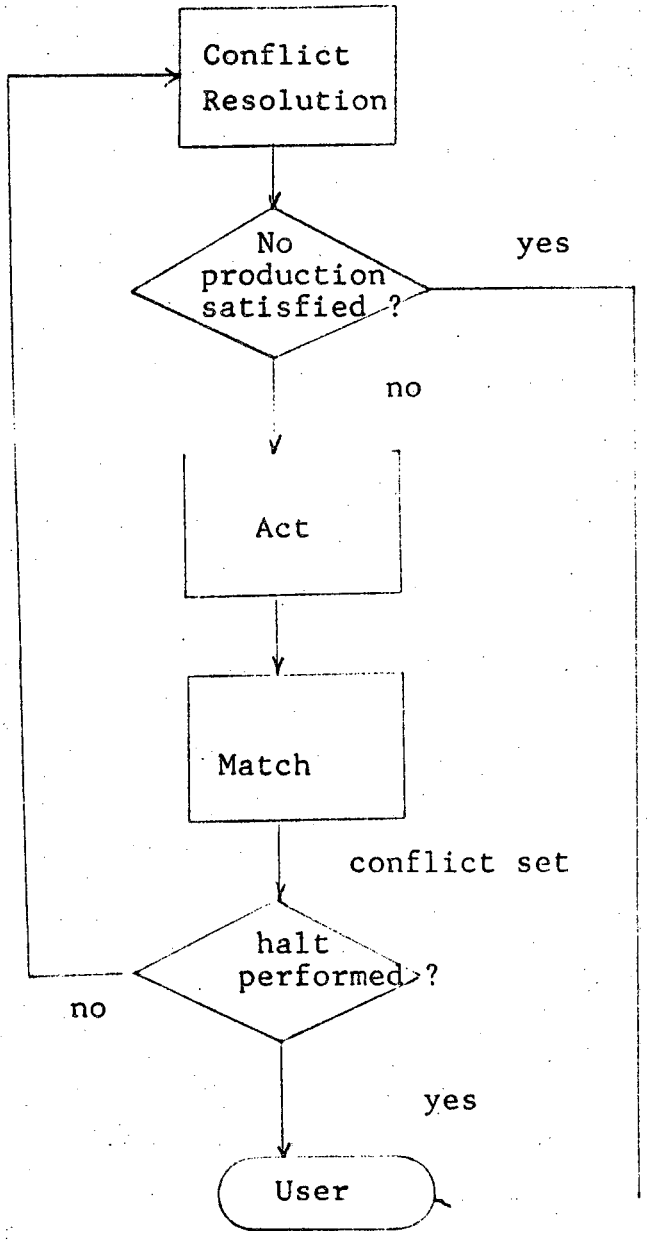


eg.



An instantiation of the above example will be :
(goal ^status active ^type find ^object block ^color red)

Standard Structured Data Type



Mechanism of the Recognize Act Cycle

The Recognize Act Cycle of OPS5 consists of three main processe i) Conflict Resolution ii) Act iii) Match. The output of the Match pro- and the input to Conflict Resolution, is a set of productions called t conflict set. The objects in the conflict set are called instantiation An instantiation is an ordered pair of a production name and a list of working memory elements satisfying the production's LHS. During confli resolution, the interpreter exames the conflict set to find an instant which dominates all the others under the ordering rules listed below. dominant instantiation will be executed in the act phase of the cycle. set of ordering rules for instantiation is called a conflict resolutic strategy. OPS5 provides two strategies LEX and MEA. Although these str are rather complex to describe, what they achieve is simple:

i) Both strategies prevent instantiations from executing more than once.
ii) They make production systems attend to the most recent data in wor memory.
iii) They give preference to production with more specific LHS.

These things are important because they make it easy to add production to an existing set and have the new productions fire at the right time and because they make it easy to simulate common constructs such as lo and subroutine calls. For detail explanation of LEX and MEA strategies readers can refer to the next two pages.

LEX STRATEGY

The LEX conflict resolution strategy contains four rules which are applied in order to find the instantiation that dominates under them.

1. Discard from the conflict set the instantiations that have already fired.If there are no instantiations that have not fired,conflict resolution fails and no instantiations is selected.

2. Order the instantiations on the basis of the recency of the working memory elements,using the following algorithm to compare pairs of instantiations: First compare the most recent elements from the two instantiations.If one element is more recent than the other,the instantaition containing that element dominates.If two elements are equally recent,compare the second most recent elements from the instantiations.Continue in thismanner either until one element of one instantiation is found to be more recent than the corresponding element in the other instantiation,or until no elements remain for one instantiation.If one instantiation is exhausted before the other the instantiation not exhausted dominates;if the two instantiations are exhausted at the same time,neither dominates.

3. If no one instantiation dominates all the other under the previous rule,compare the dominant instantiations on the basis of the specific of the LHSs of the productions.Count the number of tests(for constant and variables) that have to be made in finding an instantiation for the LHS.The LHSs that require more tests dominate.

4. If no single instantiation dominates after the previous rule,make an arbitrary selection of the dominant instantiation.

MEA STRATEGY

The MEA strategy differs from LEX in that another rule has been added after the first.The rule that was second had to be modified slightly to accomodate the new rule.The new rules for MEA are :

1. Discard from the conflict set the instantiations that have alrea fired.If there are no instantiations that have not fired,conflic resolution fails and no instantiation is selected.

2. Compare the recencies of the working memory elements matching th first condition elements of the instantiations.The instantiation using the most recent working memory elements dominates.

3. Order the instantiations on the basis of the recencies of the remaining working memory elements using the following algorithm to compare pairs of instantiations: First compare the most recent elements from the two instantiations.If one element is more rece than the other,the instantiation containing that element dominat If the two elements are equally recent,compare thesecond most re elements from the instantiations.Continue in this manner either u one element of one instantiation is found to be more recent than corresponding element in the othe instantiation,or until no elem remain for one instantiation.If one instantiation is exhausted b the other,the instantiation not exhausted dominates;if the two instantiations are exhausted at the same time,neitherdominates.

4. If no one instantiation dominates all the other under the previou rule,compare the dominant instantiations on the basis of the spec icity of the LHSs of the productions.Count the number of tests(f constants and variables) that have to be made in finding an inst iation for the LHS.The LHSs that require more tests dominate.

5. If no single instantiation dominates after the previous rule,make arbitrary selection of the dominant instantiation.

System Organization

In the previous section, we have discussed the general structure of OPS5 and its mechanism; and in this section we will see how the inventory model expert system is implemented in OPS5.Knowledge about models is explicitly stated in the working memory in the form of structured data type elements.And the knowledge of how to use these models and their assumptions are stored in the form of productions. In the inventory model expert system ,a typical production would be something like :

```
( P p2
    ( Goal ^type  yes  ^status active )
-->
    ( write (crlf) is your problem single item or multi item ? )
    ( write (crlf) type in SI or MI )
    ( make  item ^type (accept)) )
```

Translated into plain English, this production states : " if there is an working memory element called goal whose attribute "type" has a value "yes" and whose attribute "state" has a value "active", then prompt the user for information regarding whether his problem is a single item or a multi item inventory problem and create a new working memory element called "item" with attribute "type" which has a value of "SI" or "MI" (depending on what the user has input).

Several such productions are present. Each of which may augment the knowledge base in some form. A complete listing of the production fo the expert system is in the appendix. As productions operate on existing working memory elements, there obviously has to be some working memory element(s) present when the program starts executing. We make use of the "make" action described earlier to create the first working memory eleme "goal" with two attributes "type" and "status". Initially these may have values like "no" or "satisfied". One of the productions could act on thi element and change its attribute values to "yes" and "active" respective

Using this as a base, our earlier production p2 could execute its RHS which results in the creation of the element "item" in working memory Finally, the presence of certain working memory elements with particu values for the associated attributes, will prompt the choice of a par inventory model.

The system prints out the formulas the decision maker needs to in order to compute relevant parameters like optimum order quantity, optimum cycle length, optimum total cost etc. The system under consid supports about 7 different deterministic inventory models, beginning the classical EOQ models. These are listed below :

1) The fundamental EOQ model.
2) The problem of EOQ with uniform demand and production runs unequal length.
3) The problem of EOQ with finite replenishment rate.
4) The problem of EOQ with finite replenishment rate and short
5) The problem of EOQ with shortage.
6) The problem of EOQ with non-zero lead time.
7) The problem of EOQ with shortage and non-zero lead time.

Details of each model are shown in the next few pages.

INVENTORY CONTROL CONDITIONS

1. Number of items
   a) single item
   b) multi items

2. Number of stages
   a) single stage
   b) multi stages

3. Time Horizon
   a) infinite
   b) finite

4. Lead Time
   a) zero
   b) non zero

5. replenishment
   a) instantaneous
   b) gradual

6. Safety stock
   a) permitted
   b) not permitted

7. Shortages
   a) permitted
   b) not permitted

8. Order quantity
   a) constant
   b) variable

9. Production quantity
   a) constant
   b) variable

10. Inventory cycle length
    a) constant
    b) variable

11. Holding cost
    a) constant
    b) variable

12. Set up cost
    a) constant
    b) variable

13. Ordering cost
    a) constant
    b) variable

14. Demand
    a) deterministic
    b) stochastic

15. Production cost
    a) constant
    b) variable

16. Purchase cost
    a) constant
    b) variable

17. Reorder level
    a) zero
    b) non zero

GLOSSARY OF TERMS

D = units demanded per time period
C_o = ordering cost per order placed (set up cost)
C_c = carrying cost per unit time period
Q = quantity ordered per order
C_s = shortage cost per unit per time period
t_c = inventory cycle (time between orders per production run)
t_L = lead time
Q^* = optimal order (production) quantity (units)
N^* = optimal no. of orders per time period
C^* = optimal total cost
R^* = optimal reorder level
B = no. of units backordered per inventory cycle
S =maximum inventory level
p_1 = production rate
p_2 = depletion rate
$f_1 \ldots f_i$ = limitation factors in multiitem model

AVAILABLE MODELS FOR INVENTORY SYSTEMS

Model 1

Conditions satisfied : 1a,2a,3a,4a,5a,6b,7b,8(or 9)a,10a,11a
12(or 13)a,14a,15(or 16)a,17a

Formula :

$$Q^* = \sqrt{[(2DC_o)/C_c]}$$

$$N^* = D/Q^*$$

$$t_c^* = 1/N^*$$

$$C^* = \left[C_o D/Q^* + C_c \frac{Q^*}{2}\right]$$

Model 2

Conditions satisfied : 1a,2a,3a,4b,5a,6b,7b,8(or 9)a,10a,11a,
12(or 13)a,14a,15(or 16)a,17b

Formula :

$$Q^* = \sqrt{[(2DC_o)/C_c]}$$

$$N^* = D/Q^*$$

$$t_c^* = 1/N^*$$

$$C^* = \left[C_o D/Q^* + C_c \frac{Q^*}{2}\right]$$

$$R^* = t_L D - [t_L/t_c]Q^*$$

Model 3

Conditions satisfied : 1a,2a,3a,4a,5a,6b,7b,8(or 9)a,10b,11a,
12(or 13)a,14a,15(or 16)a,17a

Formula :

$$Q^* = \sqrt{[2DC_o/C_c]}$$

$$N^* = D/Q^*$$

$$C^* = \left(C_o \frac{D}{Q^*} + C_c \frac{Q^*}{2}\right)$$

Model 4

Conditions satisfied : 1a,2a,3a,4a,5b,6b,7b,8(or 9)a,10a,11a, 12(or 13)a,14a,15a,17a

Formula : $Q^* = \sqrt{\left\{\frac{(2C_0 r_2)}{C_c[1 - r_2/r_1]}\right\}}$ $\quad t_c^* = \sqrt{\frac{2C_0}{C_c D[1 - r_2/r_1]}}$

$N^* = D/Q^* = r_2/Q^*$

$C^* = \sqrt{\{2C_0 C_c D[1 - r_2/r_1]\}}$

Model 5

Conditions satisfied : 1a,2a,3a,4a,5a,6b,7a,8(or 9)a,10a,11a, 12a,14a,15(or 16)a,17b

Formula : $Q^* = \sqrt{\frac{2C_0 D}{C_c}}\sqrt{\frac{C_0 + C_s}{C_s}}$ $\quad N^* = \sqrt{DC_c/2C_0}\sqrt{\frac{C_s}{C_c + C_s}}$

$S^* = \sqrt{\frac{2C_0 D}{C_c}}\sqrt{\frac{C_s}{C_c + C_s}}$ $\quad t_c^* = \sqrt{2C_0/DC_c}\sqrt{\frac{C_c + C_s}{C_s}}$

$B^* = Q^* - S^*$ $\quad C^* = \sqrt{2C_0 C_c D}\sqrt{\frac{C_s}{C_c + C_s}}$

Model 6

Conditions satisfied : 1a,2a,3a,4b,5b,6b,7b,8(or 9)a,10a,11a, 12(or 13)a,14a,15(or 16)a,17b

Formula : $Q^* = \sqrt{\frac{2C_0 D}{C_c}}\sqrt{\frac{C_0 + C_s}{C_s}}$ $\quad N^* = \sqrt{DC_c/2C_0}\sqrt{\frac{C_s}{C_c + C_s}}$

$S^* = \sqrt{\frac{2C_0 D}{C_c}}\sqrt{\frac{C_s}{C_c + C_s}}$ $\quad t_c^* = \sqrt{2C_0/DC_c}\sqrt{\frac{C_c + C_s}{C_s}}$

$B^* = Q^* - S^*$ $\quad C^* = \sqrt{2C_0 C_c D}\sqrt{\frac{C_s}{C_c + C_s}}$

$R^* = t_L D - [t_L/t_c]Q^* - (Q^* - S^*)$

Model 7

Conditions satisfied : 1a,2a,3a,4a,5b,6b,7a,8(or 9)a,10a,11a,
12(or13)a,14a,15(or 16)a,17b

Formula :

$$Q^* = \sqrt{\frac{2C_o(C_c+C_s)}{C_cC_s}}\sqrt{\frac{r_1 r_2}{r_1 - r_2}}$$

$$B^* = \sqrt{\frac{2C_oC_c}{(C_o+C_s)C_s}}\sqrt{\frac{r_2(r_1-r_2)}{r_1}}$$

$$S^* = \frac{(r_1-r_2)}{r_1}Q^* - B^*$$

$$t_c^* = \frac{Q^*}{r_2}$$

$$C^* = \sqrt{\frac{2C_cC_oC_s}{C_c+C_s}}\sqrt{\frac{r_1(r_1-r_2)}{r_1}}$$

Evaluation

Production sytem languages (P.S.) have two properties which make them particularly well suited for domains like decision support system or management science domains.

1) P.S. make it easy to implement recognition driven systems.
2) They make it easy to implement systems incrementally. It is often difficult to specific in advance of system design what particular pieces of information will be relevant at any point in the process. Since information has no structuring principle,it is conceptually just a big set of information and it is most naturally stored in a single global memory.

Because a production system language provides sucha memory it is unnec to determine a priori what particular features of situation need to be attended to each rule can watch for a set of features it recognizes. production system language also makes it easy to incrementally build u a system's expertise. For example in a business expert system, the fin subsystem can be added after the production subsystem has been built a checked for diagnostis.

OPS5 has characteristic that make it a particularily appropr vehicle for implementing production systems.

1) OPS5 is an efficient implementation of a recognize act architecture uses techniques that make cycle time essentially independent of the of both production memory and working memory.
2) It provides a special case of conflict resolution strategy which ma it easy to deal with knowledge of exceptional situation.
3) Another strength of OPS5 is language generality which makes it easy tailor the design of an expert system to fit the characteristic of domain. This generality also facilitates diverse data representatic and control structure within a single program and permits different organizing principles in different sectors of a program. Thus OPS5 easily handle multiple goals.
4) In a general purpose production system language,pattern matching is crucial. It affects all aspects of the design, from control constru to data representations. OPS5 permits compiler patterns that are pr by a very efficient pattern matcher

4) In a general purpose production system language, pattern matching is crucial. It affects all aspects of the design from control constru to data representations. OPS5 permits complex patterns that are proce by a very efficient pattern matcher.

However, unlike EMYCIN, KAS and EXPERT,the OPS5 model does not have sophisticated facilities for expanding the database or explaining its reasoning process. It does not provide an interrupt facility to handle new data on arrival. A major weakness of OPS5 is its lack of a sophistic programming environment. Although the language does have some debugging aids, the overall programming environment lacks the power of a typical LISP system. Another drawback is that OPS5 looks only remotely like its English equivalent. Hence it may need a human translator.

Conclusion

The expert system for inventory models works very well for the problem domain discussed herein. While the domain here has been restri to a handful of deterministic inventory models, one could expand this range to include multistage problem, multiitem problems and stochastic inventory problem as well. While the existing system relies on the rec act cycle which in essence involves pattern recognition, attempts coul be made at making the users contribution less direct. For example inst of asking the user to state explicitly whether demand is stochastic or deterministic, the system could establish this fact for him based on historical data. In this case, the system would use other management science models such as regression analysis or time series analysis to arrive at a conclusion.

Further Research

Provision will have to be made for the system to interface wit an existing relational database and mathematical subroutine libraries. As a first step, we have interfaced the rule based expert system writt in OPS5 with a simple regression routine written in LISP.

Eg. A production rule of the following form is written in OPS5.

```
( p r1
      ( demand ^type linear ^var product1 )
-->
      ( call regress (make DQTY)))
```

if demand is linearly related to product1
then call the routine "regress" and make the working memory eleme called DQTY.

"regress" is a routine written in LISP which must be declared to the interpreter before it is used in the RHS. The declaration is u follows :

```
(external regress)
```

"regress" uses data inputted in the LISP environment, and calculates the unknown parameters. To make "DQTY", we first need to literalize it using

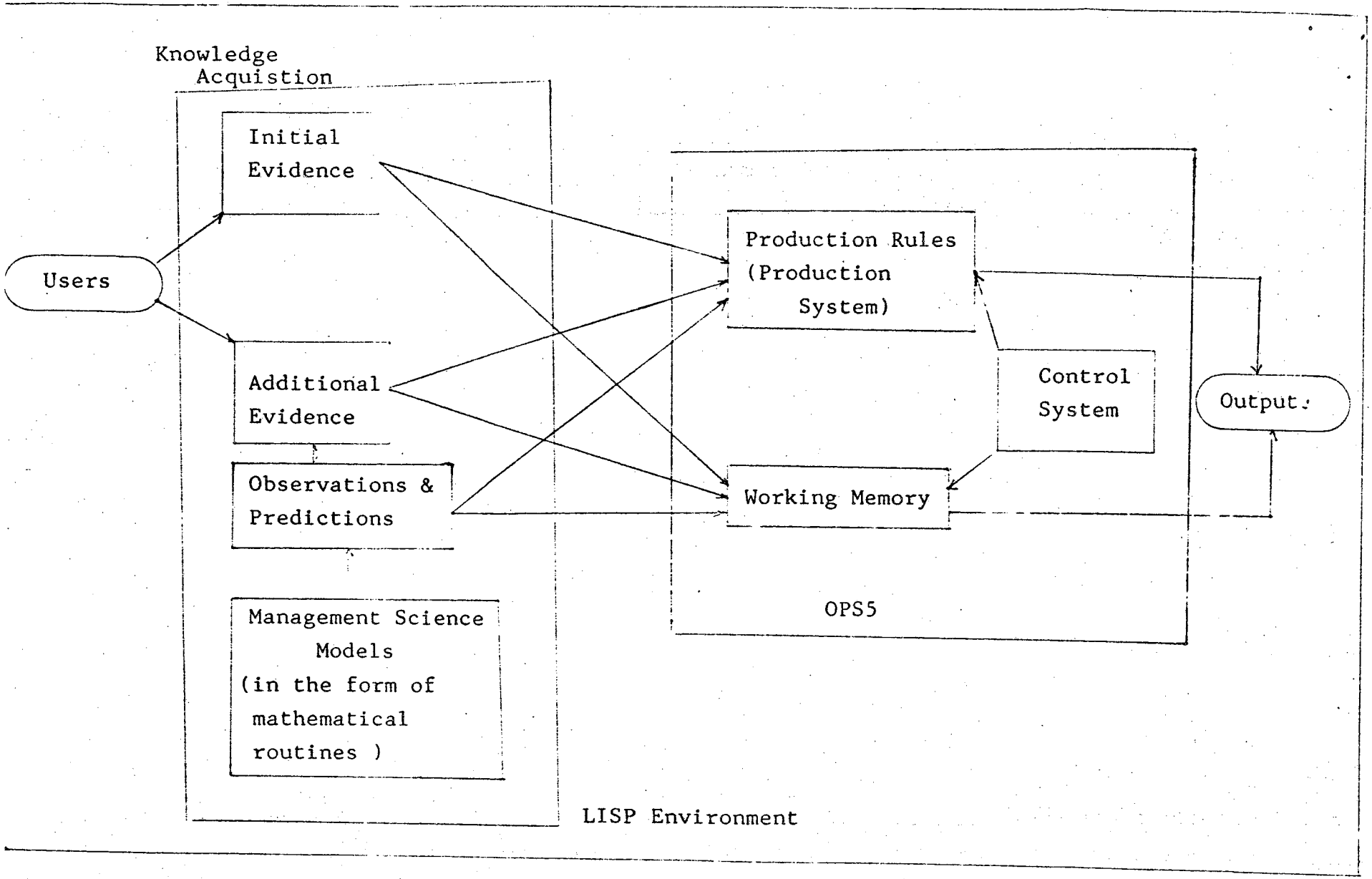
```
( literalize DQTY
        alpha
        beta
        DmQty )
```

where alpha and beta are the coefficients of the simple regression model are written into the working memory using ($assert),($tab) and ($value): functions provided by OPS5 to communicate with a LISP routine.

The above is an example of the number of issues that need to be implemented and pursued for future research.

References

1. The Handbook of Artificial Intelligence-Vol 1,2,2 by Cohen Paul and Feigenbaum Edward. Heuristic press Ca.
2. Foundation of Decision Support Systems by Bonczek, Holsapple and Whiston A.B. Academic Press.
3. Production/operation Management by Schmennes W. Roger. Science Research Associates.
4. OPS5 User's Manual by Forgy Charles.CMU-CS-81-135.
5. Principle of Artificial Intelligence by Nilsson N.J. Tioga Publishing Company.
6 LISPcraft by Wilensky Robert. W.W. Norton & Company.
7. A practical guide to designing Expert Systems by Weiss S.M. Rulikowski C.A. Aowan & Allenheld.
8 Operations Research Techniques by Moskowitz H & Wright G.P. Prentice Hall.



Extension of the present Inventory Models
Expert System

Appendix 1

Listing of the inventory models expert system productions.

```
;in this project models are assumed single stage, infinite time horizon
;safety stock is not permitted, production quantity is constant, holding cost
;set up cost is constant, demand is deterministic


;terminology
;d: units demanded per time period
;co: odering cost per order placed
;cc: carrying cost per time period
;q: quantity ordered per order
;cs: shortage cost per unit
;tc: inventory cycle
;tl: lead time
;qo: optimal order quantity
;no: optimal number of orders
;oc: optimal total cost
;r: optimal reorder level
;b: # of units backordered
;s: maximum inventory level
;p1: production rate
;p2: depletion rate

;number of items
;   a: single item
;   b: multi  item
;lead time
;   a: zero
;   b: non-zero
;replenishment
;   a: instantaneous
;   b: gradual
;shortages
;   a: permitted
;   b: not permitted
;inventiry cycle length
;   a: constant
;   b: variable
;reorder level
;   a: zero
;   b: non zero



(literalize  model
   shortage
   reorderlevel
   inventory
   leadtime
   replenishment    )

(literalize  goal
   status   )

(literalize  want
   type     )

(literalize  object
    name
   category7
   category17
```

```
   category10
   category4
   category5    )

(p r1
     (goal   ^status start)
     (model  ^shortage  nil
        ^reorderlevel nil
        ^inventory nil
        ^leadtime  nil
        ^replenishment  nil    )
-->
     (make goal   ^status active)
     (write  (crlf)hello! this is expert system
        (crlf)i need info about category shortage  )
     (modify 2  ^shortage(accept))                  )

(p r2
     (goal    ^status   active)
    (object  ^name two  ^category7    b     )
    (model   ^shortage    b
         ^reorderlevel   nil
         ^inventory   nil
         ^leadtime    nil
         ^replenishment    nil  )
-->
     (write  (crlf)you can use model1 2 3 4
        (crlf)i need info about category reorder level  )
     (modify 3     ^reorderlevel(accept))               )

(p r3
     (goal       ^status   active)
     (object  ^name three     ^category17   a)
     (model      ^shortage    b
          ^reorderlevel   a
          ^inventory   nil
          ^leadtime    nil
          ^replenishment    nil    )
-->
     (write(crlf)you can use model 1 3 4
      (crlf)i need info about category inventory cycle   )
     (modify 3      ^inventory (accept))             )

(p r4
     (goal     ^status   active)
     (object ^name five   ^category10   a     )
     (model    ^shortage    b
         ^reorderlevel   a
         ^inventory   a
         ^leadtime    nil
         ^replenishment    nil    )
-->
     (write(crlf)you can use model 1 4
      (crlf)i need info about category lead time)
(modify 3         ^leadtime (accept))      )

(p r5
     (goal     ^status     active)
     (object ^name seven   ^category4      a     )
     (model    ^shortage      b
```

```
          ^reorderlevel     a
          ^inventory     a
          ^leadtime      a
          ^replenishment       nil    )
-->
     (write(crlf)you can still use model1 4
           (crlf)i need info about category replenishment )
     (modify  3     ^replenishment  (accept))          )

(p r6
     (goal     ^status    active)
     (object   ^name ten  ^category5     b      )
     (model    ^shortage     b
         ^reorderlevel    a
         ^inventory    a
         ^leadtime     a
         ^replenishment     b      )
-->
     (make goal     ^status    active    )
     (write(crlf)your final model is 4
           (crlf)"
           qo=[[2*co*p2/cc[1-p2/p1]]]**.5
           n=d/q
           oc=[[2*co*cc*d[1-p2/p1]]]**.5
           tc=[[2*co/[cc*d[1-p2/p1]]]]**.5"
           (crlf)do you want other case?  )
     (make     want    ^type(accept))        )

(p r7
     (goal      ^status     active)
     (want      ^type       no     )
-->
     (write(crlf)thank you  goodbye)
     (halt))

(p r8
     (goal      ^status    active)
     (want      ^type      yes    )
-->
     (make    goal    ^status   start)
     (make    want    ^type     nil  )
     (make    model   ^shortage    nil
               ^reorderlevel  nil
               ^inventory  nil
               ^leadtime   nil
               ^replenishment    nil     ))

(p r9
     (goal      ^status    active     )
     (object   ^name nine    ^category5     a          )
     (model     ^shortage     b
          ^reorderlevel    a
          ^inventory    a
          ^leadtime     a
          ^replenishment     a          )
-->
     (write(crlf)your final model is 1
           (crlf)"
           qo=[[2*d*co]/cc]**.5
           no=d/qo
```

```
          tc=1/no
          oc=[co*d/qo+cc*qo/2]"
          (crlf)do you want other case?  )
     (make     want    ^type (accept))    )

(p r10
     (goal     ^status    active  )
     (object  ^name eight   ^category4     b       )
     (model    ^shortage     b
          ^reorderlevel    a
          ^inventory    a
          ^leadtime     b
          ^replenishment     nil   )
-->
     (write(crlf)in system no such model
          (crlf)do you want other model?)
     (make   want     ^type   (accept))  )

(p r11
     (goal     ^status   active)
     (object ^name six    ^category10   b)
     (model   ^shortage    b
         ^reorderlevel   a
         ^inventory   b
         ^leadtime    nil
         ^replenishment    nil    )
-->
     (write(crlf)to use model3  i need more info about category lead time
     (modify 3    ^leadtime (accept))        )

(p r12
     (goal  ^status    active)
     (object ^name seven  ^category4   a     )
     (model   ^shortage   b
         ^reorderlevel  a
         ^inventory  b
         ^leadtime   a
         ^replenishment   nil     )
-->
     (write(crlf)to use model3 i need more info about category replenishr
     (modify 3        ^replenishment   (accept))      )

(p r13
     (goal ^status    active  )
     (object ^name ten   ^category5   b     )
     (model    ^shortage   b
          ^reorderlevel  a
          ^inventory  b
          ^leadtime   a
          ^replenishment   b     )
-->
     (write(crlf)in system no such model
          (crlf)do you want other model? )
     (make   want   ^type(accept))        )

(p r14
     (goal    ^status   active   )
     (object ^name nine  ^category5    a        )
     (model   ^shortage    b
         ^reorderlevel   a
```

```
      ^inventory    b
      ^leadtime     a
      ^replenishment     a          )

  (write(crlf)your final model is 3
        (crlf)"
        qo=[[2*d*co/cc]**.5
        no=d/qo
        oc=[co*d/qo+cc*qo/2]"
        (crlf)do you want other case? )
  (make   want      ^type(accept))      )

r15
  (goal     ^status   active    )
  (object ^name eight    ^category4      b)
  (model    ^shortage     b
      ^reorderlevel    a
      ^inventory    b
      ^leadtime     b
      ^replenishment     nil     )

  (write(crlf)in system no such model
        (crlf)do you want other model?)
  (make     want   ^type (accept))       )

r16
  (goal     ^status   active)
  (object ^name four   ^category17    b      )
  (model    ^shortage     b
      ^reorderlevel    b
      ^inventory    nil
      ^leadtime     nil
      ^replenishment     nil     )

  (write(crlf)to use model2 i need more info about category inventory cy
  (modify 3    ^inventory (accept)))

r17
  (goal       ^status     active)
  (object ^name six     ^category10      b     )
  (model      ^shortage        b
        ^reorderlevel      b
        ^inventory      b
        ^leadtime       nil
        ^replenishment        nil     )

  (write(crlf)in system no such model
        (crlf)do you want other model? )
  (make   want     ^type  (accept))    )

r18
  (goal     ^status   active  )
  (object ^name seven   ^category4     a         )
  (model    ^shortage     b
      ^reorderlevel    b
      ^inventory    a
      ^leadtime     a
      ^replenishment     nil     )

  (write(crlf)in system no such model
```

```
          (crlf)do you want other model?  )
    (make   want    ^type (accept))       )
```

555

```
(p r19
     (goal  ^status   active  )
     (object ^name nine   ^category5    a   )
     (model    ^shortage    b
          ^reorderlevel   b
          ^inventory   a
          ^leadtime    b
          ^replenishment    a)
-->
     (write(crlf)your final model is 2
           (crlf)"
           qo=[[2*d*co]/cc]**.5
           no=d/qo
           tc=1/no
           oc=[co*d/qo+cc*qo/2]"
           (crlf)do you want other case? )
     (make    want   ^type (accept))     )

(p r20
     (goal     ^status   active)
     (object ^name ten   ^category5    b     )
     (model    ^shortage    b
          ^reorderlevel   b
          ^inventory   a
          ^leadtime    b
          ^replenishment    b     )
-->
     (write(crlf)in system no such model
          (crlf)do you want other model? )
     (make    want    ^type  (accept))    )

(p r21
     (goal   ^status   active)
     (object ^name one ^category7    a)
     (model  ^shortage    a
        ^reorderlevel   nil
        ^inventory   nil
        ^leadtime    nil
        ^replenishment    nil    )
-->
     (write(crlf)you can use model 5 6 7
          (crlf)i need info about category reorder level)
     (modify 3 ^reorderlevel   (accept))     )

(p r22
     (goal    ^status   active)
     (object ^name three  ^category17   a     )
     (model   ^shortage    a
        ^reorderlevel   a
        ^inventory   nil
        ^leadtime    nil
        ^replenishment    nil    )
-->
     (write(crlf)in system no such model
          (crlf)do you want other model?  )
     (make    want   ^type (accept))       )
```

```
(p r23
     (goal      ^status    active    )
     (object ^name six    ^category10    b          )
     (model     ^shortage     a
          ^reorderlevel    b
          ^inventory    b
          ^leadtime     nil
          ^replenishment     nil      )
-->
     (write(crlf)in system no such model
           (crlf)do you want other model?  )
     (make    want    ^type(accept))          )

(p r24
     (goal    ^status     active)
     (object ^name five ^category10     a      )
     (model  ^shortage       a
        ^reorderlevel     b
        ^inventory     a
        ^leadtime      nil
        ^replenishment      nil     )
-->
     (write(crlf)you can use model 5 6 7
           (crlf)i need info about category lead time )
     (modify 3 ^leadtime (accept)))

(p r25
     (goal       ^status    active  )
     (object ^name eight     ^category4     b         )
     (model      ^shortage     a
           ^reorderlevel    b
           ^inventory    a
           ^leadtime     b
           ^replenishment     b        )
-->
     (write(crlf)in system no such model
           (crlf)do you want other model?  )
     (make      want     ^type  (accept))      )

(p r26
     (goal    ^status    active    )
     (object ^name eight ^category4     b          )
     (object ^name nine ^category5     a         )
     (model  ^shortage     a
        ^reorderlevel    b
        ^inventory    a
        ^leadtime     b
        ^replenishment     a     )
-->
     (write(crlf)your final model is 6
           (crlf)"
           qo=[[2*co*d/cc]**.5]*[[co+cs]/cs]**.5
           no=[[d*cc]/[2*co]]**.5 * [cs/[cc+cs]]**.5
           oc=[2*co*cc*d]**.5 *[cs/[cc+cs]]**.5
           r=t1-[t1/tc]*qo-[qo-s]"
           (crlf)do you want other case? )
     (make  want    ^type(accept))          )

(p r27
     (goal     ^status    active )
```

```
     (object ^name seven  ^category4    a       )
     (model   ^shortage    a
         ^reorderlevel   b
         ^inventory   a
         ^leadtime    a
         ^replenishment    nil )
-->
     (write(crlf)you can use model 5 7
           (crlf)i need info about category replenishment)
     (modify 3 ^replenishment (accept))                )

(p r28
     (goal    ^status   active  )
     (object ^name nine  ^category5    a       )
     (model   ^shortage    a
         ^reorderlevel   b
         ^inventory   a
         ^leadtime    a
         ^replenishment    a        )
-->
     (write(crlf)your final model is 5
           (crlf)"
           qo=[[2*co*d]/cc]**.5 * [[co+cs]/cs]**.5
           no=[[d*cc]/[2*co]]**.5 * [[cs/[cc+cs]]**.5]
           oc=[2*co*cc*d]**.5 * [[cs/[cc+cs]]**.5]"
           (crlf) do you want other case?)
     (make     want   ^type (accept))    )

(p r29
     (goal     ^status    active)
     (object ^name ten   ^category5     b     )
     (model    ^shortage     a
          ^reorderlevel    b
          ^inventory    a
          ^leadtime     a
          ^replenishment     b     )
-->
     (write(crlf)your final model is 7
           (crlf)"
           qo=[[2*co*[cc+cs]]/[cc*cs]]**.5 * [[p1*p2]/[p1-p2]]**.5
           tc=qo/p2
           oc=[[2*cc*co*cs]/[cc+cs]]**.5 * [[p2[p1-p2]/p1]]**.5"
           (crlf)do you want other case? )
     (make   want   ^type (accept))     )

(p rule31
     (goal ^status  active)
     (object ^name five  ^category10 a)
     (model    ^shortage  b
          ^reorderlevel b
          ^inventory a
          ^leadtime  nil
          ^replenishment  nil )
-->
     (write(crlf)to use model2  i need more info about category lead time
     (modify 3    ^leadtime (accept))     )

(p r32
     (goal ^status active)
     (object ^name eight  ^category4 b)
```

```
    (model    ^shortage  b
         ^reorderlevel b
         ^inventory a
         ^leadtime  b
         ^replenishment  nil)
-->
    (write(crlf)to use model2  i need more about category replenishment)
    (modify 3      ^replenishment   (accept))   )

(p r33
    (goal    ^status    active)
    (object  ^name four  ^category17   b)
    (model   ^shortage   a
        ^reorderlevel  b
        ^inventory  nil
        ^leadtime   nil
        ^replenishment   nil   )
-->
    (write(crlf)you can use model5 6 7
          (crlf)i need info about categoty inventory cycle)
    (modify  3   ^inventory (accept))   )

(p r34
    (goal   ^status   active)
    (object   ^name eight  ^category4   b)
    (model    ^shortage   a
         ^reorderlevel  b
         ^inventory  a
         ^leadtime   b
         ^replenishment   nil   )
-->
    (write(crlf)to use model6 i need more info about category replenishm
    (modify 3   ^replenishment (accept)) )
(p r30
    (begin)
-->
    (make  goal   ^status   start)
    (make  model  ^shortage    nil
             ^reorderlevel   nil
             ^inventory   nil
             ^leadtime    nil
             ^replenishment    nil   )
    (make object ^name ten  ^category5    b)
    (make object ^name eight  ^category4    b)
    (make object ^name six  ^category10   b)
    (make object ^name four  ^category17   b)
    (make object ^name two  ^category7   b)
    (make object ^name nine  ^category5   a)
    (make object ^name seven  ^category4    a)
    (make object ^name five  ^category10    a)
    (make object ^name three  ^category17    a)
    (make object ^name one  ^category7    a)
    (make object ^name zero                 ))


    (make begin)
```

Appendix 2

Sample runs of the inventory models expert system.

```
Script started on Tue Dec  4 10:29:39 1984
$ (lisp)
Franz Lisp, Opus 38.69
-> (load 'src.o)
[fasl src.o]
t
-> (load 'ai)
[load ai]
**********************************t
-> (run)

1. r30 1
2. r1 2 3
hello! this is expert system
i need info about category shortage a

3. r21 15 13 17
you can use model 5 6 7
i need info about category reorder level b

4. r33 15 7 19
you can use model5 6 7
i need info about categoty inventory cycle  a

5. r24 15 11 21
you can use model 5 6 7
i need info about category lead time a

6. r27 15 10 23
you can use model 5 7
i need info about category replenishment a

7. r28 15 9 25
your final model is 5

          qo=[[2*co*d]/cc]**.5 * [[co+cs]/cs]**.5
          no=[[d*cc]/[2*co]]**.5 * [[cs/[cc+cs]]**.5]
          oc=[2*co*cc*d]**.5 * [[cs/[cc+cs]]**.5]
do you want other case?yes

8. r8 15 26
9. r1 27 29
hello! this is expert system
i need info about category shortage a

10. r21 30 13 32
you can use model 5 6 7
i need info about category reorder level b

11. r33 30 7 34
you can use model5 6 7
i need info about categoty inventory cycle a

12. r24 30 11 36
you can use model 5 6 7
i need info about category lead time b

13. r34 30 5 38
to use model6 i need more info about category replenishment a
```

```
14. r26 30 5 9 40
your final model is 6

            qo=[[2*co*d/cc]**.5]*[[co+cs]/cs]**.5
            no=[[d*cc]/[2*co]]**.5 * [cs/[cc+cs]]**.5
            oc=[2*co*cc*d]**.5 *[cs/[cc+cs]]**.5
            r=t1-[t1/tc]*qo-[qo-s]
do you want other case?yes

15. r8 30 41
16. r1 42 44
hello! this is expert system
i need info about category shortage a

17. r21 45 13 47
you can use model 5 6 7
i need info about category reorder level b

18. r33 45 7 49
you can use model5 6 7
i need info about categoty inventory cycle a

19. r24 45 11 51
you can use model 5 6 7
i need info about category lead time a

20. r27 45 10 53
you can use model 5 7
i need info about category replenishment b

21. r29 45 4 55
your final model is 7

            qo=[[2*co*[cc+cs]]/[cc*cs]]**.5 * [[p1*p2]/[p1-p2]]**.5
            tc=qo/p2
            oc=[[2*cc*co*cs]/[cc+cs]]**.5 * [[p2[p1-p2]/p1]]**.5
do you want other case?yes

22. r8 45 56
23. r1 57 59
hello! this is expert system
i need info about category shortage b

24. r2 60 8 62
you can use model1 2 3 4
i need info about category reorder level a

25. r3 60 12 64
you can use model 1 3 4
i need info about category inventory cycle b

26. r11 60 6 66
to use model3 i need more info about category lead time a

27. r12 60 10 68
to use model3 i need more info about category replenishment a

28. r14 60 9 70
your final model is 3
```

```
          qo=[[2*d*co/cc]**.5
          no=d/qo
          oc=[co*d/qo+cc*qo/2]
do you want other case?yes

29. r8 60 71
30. r1 72 74
hello! this is expert system
i need info about category shortage b

31. r2 75 8 77
you can use model1 2 3 4
i need info about category reorder level b

32. r16 75 7 79
to use model2 i need more info about category inventory cycle a

33. rule31 75 11 81
to use model2 i need more info about category lead time b

34. r32 75 5 83
to use model2 i need more about category replenishment a

35. r19 75 9 85
your final model is 2

          qo=[[2*d*co]/cc]**.5
          no=d/qo
          tc=1/no
          oc=[co*d/qo+cc*qo/2]
do you want other case?yes

36. r8 75 86
37. r1 87 89
hello! this is expert system
i need info about category shortage a

38. r21 90 13 92
you can use model 5 6 7
i need info about category reorder level a

39. r22 90 12 94
in system no such model
do you want other model?yes

40. r8 90 95
41. r1 96 98
hello! this is expert system
i need info about category shortage b

42. r2 99 8 101
you can use model1 2 3 4
i need info about category reorder level a

43. r3 99 12 103
you can use model 1 3 4
i need info about category inventory cycle a

44. r4 99 11 105
you can use model 1 4
```

i need info about category lead time b

45. r10 99 5 107
in system no such model
do you want other model?yes

46. r8 99 108
47. r1 109 111
hello! this is expert system
i need info about category shortage a

48. r21 112 13 114
you can use model 5 6 7
i need info about category reorder level b

49. r33 112 7 116
you can use model5 6 7
i need info about categoty inventory cycle b

50. r23 112 6 118
in system no such model
do you want other model?yes

51. r8 112 119
52. r1 120 122
hello! this is expert system
i need info about category shortage b

53. r2 123 8 125
you can use model1 2 3 4
i need info about category reorder level a

54. r3 123 12 127
you can use model 1 3 4
i need info about category inventory cycle b

55. r11 123 6 129
to use model3 i need more info about category lead time a

56. r12 123 10 131
to use model3 i need more info about category replenishment a

57. r14 123 9 133
your final model is 3

```
qo=[[2*d*co/cc]**.5
no=d/qo
oc=[co*d/qo+cc*qo/2]
```

do you want other case?yes

58. r8 123 134
59. r1 135 137
hello! this is expert system
i need info about category shortage b

60. r2 138 8 140
you can use model1 2 3 4
i need info about category reorder level a

61. r3 138 12 142

```
you can use model 1 3 4
i need info about category inventory cycle a

62.  r4 138 11 144
you can use model 1 4
i need info about category lead time a

63.  r5 138 10 146
you can still use model1 4
i need info about category replenishment b

64.  r6 138 4 148
your final model is 4

            qo=[[2*co*p2/cc[1-p2/p1]]]**.5
            n=d/q
            oc=[[2*co*cc*d[1-p2/p1]]]**.5
            tc=[[2*co/[cc*d[1-p2/p1]]]]**.5
do you want other case?no

65.  r7 149 150
thank you goodbye
end -- explicit halt
 34 productions (238 // 576 nodes)
 65 firings (150 rhs actions)
 36 mean working memory size (62 maximum)
 60 mean conflict set size (201 maximum)
 87 mean token memory size (172 maximum)
nil
-> (exit)
$
script done on Tue Dec  4 10:39:02 1984
```

Appendix 3

Simple regression model written in LISP language.

```
(defun regress()
   (prog()
      (include test)
      (include data)
      (sum_cols 9 2)
      (mean_cols 9 2)
      (xsquare 9 2)
      (xysquare 9 2)
      (bcoeff)
      (acoeff)))
```

```
(cond[(zerop (*sqrsum* 1)) (return nil)]
     [t(setq b (quotient (*sqrsum* 0) (*sqrsum* 1)))
       (print b)])
   (terpri)))
```

567

```
(defun acoeff()
  (princ "alpha coeff  ")(terpri)
  (prog()
        (setq a (diff (*colmean* 0) (times b (*colmean* 1))))
        (print a)
        (terpri)
        (princ "the simple linear regression equation is")
        (terpri)
        (princ "y = ")
        (print a)
        (princ " + ")
        (print b)
        (princ " *  x + e ")
        ($reset)
        ($tab 2)
        ($value a)
        ($value b)
        ($assert)
        (terpri)))
```

```
(defun fill-image(l)
  (array *image* flonum(length l)(length(car l)))
  (fill-image-aux 0. l))


(defun fill-image-aux(i l)
    (cond [(null l)]
          [t(fill-row i 0. (car l))
            (fill-image-aux (add1 i)(cdr l))]))


(defun fill-row(i j sl)
    (cond[(null sl)]
         [t ( store (*image* i j) (car sl))
            (fill-row i (add1 j) (cdr sl))]))


(array *colsum* flonum 10)
(array *colmean* flonum 10)
(array *sqrsum* flonum 10)
(defun sum_cols(n m)
  (loop for j from 0 to (- m 1)
                do
                (loop for i from 0 to (- n 1) with tsum=0
                                sum(*image* i j) into tsum
                        finally
                                (store (*colsum* j) tsum))))


(defun mean_cols(n m)
  (loop for j from 0 to (- m 1)
                do
                (store(*colmean* j)
                                (quotient(*colsum* j) (float n)))))


(defun xsquare(n m)
  (loop for i from 0 to (- n 1) with sumxsqr=0

    sum(times(diff(*image* i 1) (*colmean* 1))
             (diff(*image* i 1) (*colmean* 1)))

             into sumxsqr
             finally
             (store(*sqrsum* 1) sumxsqr)))


(defun xysquare(n m)
  (loop for i from 0 to (- n 1) with xysumsqr=0
    sum(times(diff(*image* i 1) (*colmean* 1))
             (diff(*image* i 0) (*colmean* 0)))

              into xysumsqr
              finally(store(*sqrsum* 0) xysumsqr)))


(defun bcoeff()
  (princ "beta coeff  ")(terpri)
  (prog()
```