REPRESENTING SPATIAL EXPERIENCE AND SOLVING SPATIAL PROBLEMS IN A SIMULATED ROBOT ENVIRONMENT

by

PETER FORBES ROWAT

M.Sc., University of British Columbia, 1972

Techincal Report 79-14

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming
to the required standard.

....

....

....

THE UNIVERSITY OF BRITISH COLUMBIA

October, 1979

Abstract

This thesis is concerned with spatial aspects of perception and action in a simple robot. To this end, the problem of designing a robot-controller for a robot in a simulated robot-environment system is considered. The environment is a two-dimensional tabletop with movable polygonal shapes on it. The robot has an eye which 'sees' an area of the tabletop centred on itself, with a resolution which decreases from the centre to the periphery. Algorithms are presented for simulating the motion and collision of two dimensional shapes in this environment. These algorithms use representations of shape both as a sequence of boundary points and as a region in a digital image. A method is outlined for constructing and updating the world model of the robot as new visual input is received from the eye. It is proposed that, in the world model, the spatial problems of path-finding and object-moving be based on algorithms that find the skeleton of the shape of empty space and of the shape of the moved object. A new iterative algorithm for finding the skeleton, with the property that the skeleton of a connected shape is connected, is presented. This is applied to path-finding and simple object-moving problems. Finally, directions for future work are outlined.

TABLE OF CONTENTS

LIST OF FIGURES

Oh the mind, mind has mountains; cliffs of fall
Frightful, sheer, no-man-fathomed.

Gerard Manley Hopkins

Acknowledgements

I would like to acknowledge my enormous debt to Richard Rosenberg, who has supervised me, and who has given me advice, encouragement and support beyond all expectations of duty throughout the many years I have spent on the Ph.D program. I wish to thank Alan Mackworth for serving on my thesis committee and for always being ready to listen and to give me advice and encouragement; Harvey Abramson, Ray Reiter, Bob Woodham, and John Yuille for serving on my thesis committee; Gordon McCalla, Bill Havens, Rachel Gelbart, Brian Funt, Mike Kuttner, Roger Browse, Jan Mulder, and Randy Goebel for many helpful discussions; Nona and Gwen for drawing the diagrams; and above all Nona, with Ruby, Lena (and Taku), for standing by me through all these years, supporting me in every imaginable way, and for being their own delightful selves.

CHAPTER I

INTRODUCTION

I.1 Aims and motivation

This thesis is animated by a desire to understand the connection between perception and action. Every day we do such simple things as

- avoiding all obstacles in crossing a cluttered room
- navigating through an unfamiliar house
- making and executing a mental plan to go to the local shop or cross a campus
- moving an awkward piece of furniture around a house.

Likewise our pet dogs and cats are good at navigating through their spatial world.

> For an organism to do such tasks so easily, what computational processes are required?

Here you will find the beginnings of an answer to this question, which may be refined in any one of a dozen directions.

The question as stated is too nebulous to be given a meaningful answer; to delineate it more precisely I opted to proceed as follows:

1. Design and implement a simulated robot world which reflects to a certain extent the spatial aspects of a cluttered room or the floorplan of a house.
2. Specify a class of tasks of a spatial nature which the robot might reasonably be expected to solve in this world.
3. Design computational processes which enable the robot to handle these tasks in a reasonably intelligent manner.

This, in summary, has been my research program.

The simulated robot world is carefully designed to enforce a non-trivial treatment of the interaction between perception and action. The robot's sensory input from distant parts of the environment is either non-existent or very inexact and fuzzy, in accord with real world organisms; yet plans have to be made and actions executed. I am thus squarely confronted, albeit very crudely, with the problem of acting in the face of incomplete and inexact knowledge. In the simulated robot world it is possible for the executed actions to be inexact in a similarly fuzzy manner, just as in the real world. So far, however, I have suppressed this feature, in order to ease the achievement of the overriding concern: the creation of a functioning robot-controller.

This thesis is also animated by the belief that it is of fundamental importance to understand the computational processes involved in spatial problem solving. There are several lines of argument to encourage this belief.

First of all, spatial reasoning must be one of the most fundamental ability we possess, since we inhabit a spatial world and if we couldn't solve spatial problems we would always be bumping into things! We are also superbly good at it. For instance, we control large rectangular shapes on winding roads or in parking lot mazes, and a ball-player controls the velocity and spin of a small round object with fine precision.

Second, spatial reasoning satisfies the criteria proposed by [Marr,1976] to guide the choice of a research problem in AI.

> "If one believes that the aim of information-processing studies is to formulate and understand particular information-processing problems, then it is the structure of those problems that is central, not the mechanisms through which they are implemented. Therefore, the first thing to do is to find problems that we can solve well, find out how to solve them, and examine our performance in the light of that understanding. The most fruitful source of such problems is operations that we perform well, fluently, reliably, (and hence unconsciously), since it is difficult to see how reliability could be achieved if there were no sound underlying method."

Spatial reasoning is a problem that we solve "well, fluently, reliably" and largely unconsciously; therefore it is a worthwhile research objective.

Third, there is an evolutionary argument. The simple crayfish runs mazes; birds don't bump into forest leaves and branches; an orca whale races through a kelp bed without touching a stalk; a mouse will rarely fail to reach its cheese or a dog its bone; the monkey swings from branch to branch. So, as "ontogeny recapitulates phylogeny", one might well expect spatial reasoning to underly our higher mental faculties. As an aside, the minuscule brain of the hummingbird solves a devastating spatial problem: given a meadow with a profusion of flowers, each variety having different nectar-producing properties, the humming bird appears to maximize net energy input while foraging and simultaneously minimizes the time expended [Gass et al.,1976]. A truly amazing piece of computation by a very small brain.

Fourth, there is a developmental argument. Young children solve spatial problems such as the classical monkey and bananas problem before they can talk, and the newborn babe, only a few hours old, will react appropriately to a moving object, flinching if it comes dangerously close, and continuing to follow it with eye-movements if it passes behind a stationary object [Bower,1974].

Fifth, our language is permeated by spatial metaphors. Consider the word "permeate" just used. Does it not evoke a visual image consisting of "spatial metaphors", "permeating", in a very physical sense, "our language"? Does not a visual image accompany every sentence one utters? Even the most abstract type of language uses spatial metaphors. For instance, one "builds" an argument "on" a firm "foundation"; one "arrives at" a conclusion.

Sixth, there are many anecdotes concerning the use of spatial reasoning and visual imagery in making fundamental scientific discoveries. For instance, the paper models of

Pauling for the alpha-helix, and of Watson and Crick for the DNA molecule; Faraday's visualization of magnetic lines of force as narrow tubes curving through space; Kekule's discovery of the structure of the benzene molecule by his visualization of a ring of snake-like, writhing, chains, each seizing its neighbour's tail; and in mathematics, Hadamard has documented many instances where a problem was apparently solved by visual imagery.

These arguments in favour of spatial reasoning inexorably lead one to propose the hypothesis that the mechanisms required for spatial reasoning may well underly other abilities that have developed later in evolution, for instance the use of language.

The overall structure of my thesis may now be summarized. I lay out a research program and describe the progress made on several fronts. The overall implementation goal is to build a functioning robot-controller for the simulated robot. The implemented parts are described in detail. For those parts not yet implemented, their theory and design is sketched in some depth, to the point at which, in some cases, further progress can only be made through an attempt at implementation. After all is said and done, this is the feature that distinguishes Artificial Intelligence, as actually practised, from all other intellectual disciplines: the development of theory through program implementation.

I.2 The action cycle

The information-processing component of any organism that physically interacts with the outside world must consist of three distinct parts: sensory receptors, action effectors, and an intermediary that relates the senses and the actions. My main interest is in a sufficient design for the intermediary, which in this thesis will be referred to as the robot-controller. The intermediary could of course be null but that results in a very uninteresting organism which could not long survive in its world. In my case, the design of the intermediary, the robot-controller, is constrained by the requirement that the organism exhibit reasonably intelligent behaviour. (Intelligent behaviour will be taken as a primitive judgment and analyzed no further.)

The major task of a robot-controller, in order to improve the organism's survival chances, is to build a world model: a model of the outside world. In information-processing terms, a world model is a data base of facts which, together with interpretive procedures, enables the prediction of future sensory input. Equivalently, it is a data structure and procedures for making predictions about the outside world. A good world model makes correct predictions most of the time. The purpose of a world model is to allow the construction of plans and thus to better achieve the organism's goals. Building a world model is an inductive task, using sensory inputs as the primitive items of evidence. Thus the world model of an organism is a function of the design of its receptors, and furthermore can never be assumed to be correct - the true nature of the outside world is forever unknowable. As a consequence, different organisms build very different world models. To an octopus, for instance, whose sense of touch can only signal surface texture and curvature, a small smooth perspex sphere is indistinguishable from a long smooth perspex cylinder having the same curvature [Wells,1978].

The interface between an organism and the outside world is defined by the organism's sensory receptors and action effectors, and is necessarily always sloppy. This may be taken as an intuitively obvious fact, or may be supported by the following information-theoretic arguments. On the sensory side one may argue as follows. First, at any moment in time a finite organism can only receive a finite amount of information, whereas there is certainly an unbounded amount of information which could be detected at any one time and moreover, if one believes either that the features of the outside world are continuous, or that the outside world is infinite, or both, then there is an infinite amount of sense-able information. This is a special case of the more general fact that a small finite organism in a large or unbounded world could neither contain nor receive all the potentially sense-able information available from the outside world. A more practical argument: by the very design of natural sensory receptors, all input is digitized, hence is an approximation to quantities that are generally taken to be continuous.

On the action side, one may argue as follows. The very statement that the result of an action was incorrect, or sloppy, implies the existence of a world model which was used as a standard of comparison for the outcome of the action. There are two points here. First, a world model is never one hundred percent exact, so the predicted outcome may be simply wrong. Second, supposing the world model is exact, the computation of the outcome of an action may require an unbounded amount of time. But in the outside world life goes on and actions must be taken, so the computation must be cut off in a fixed amount of time - and so mistakes are inevitable.

The discussion so far is summarized in figure I.1. Our robot-controller functions, at the top level, by the perpetual repetition of the action cycle, a loop containing three parts: perception, planning, and action. In our robot-controller these three processes are performed in serial order, whereas in most living organisms they are presumably performed in parallel.

While I am discussing organisms in general, let me introduce the following terminology: the total sensory (visual, tactile, olfactory, auditory, ...) input at any instant of time is called a sensory (visual, tactile, olfactory, auditory, ...) impression, following David Hume.

In summary, then, the action cycle occupies a fundamental position in the information-processing of any organism. The

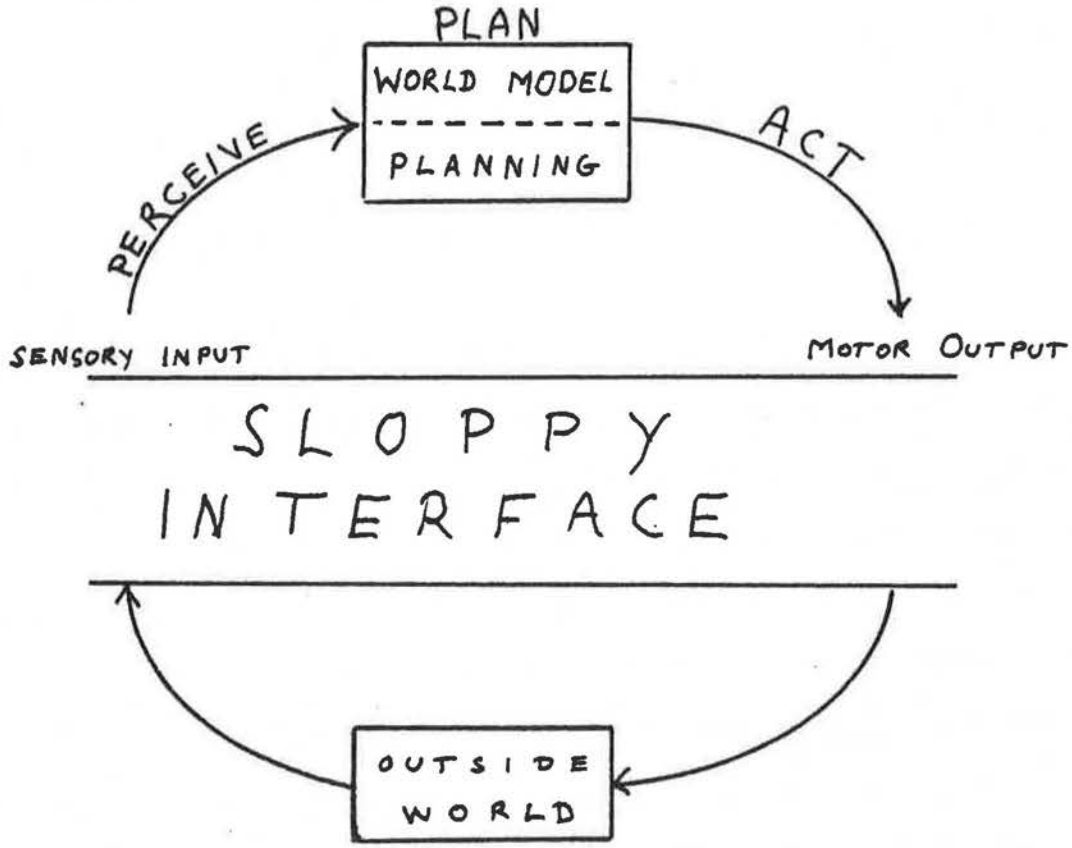


FIGURE I.1. THE ACTION CYCLE.

elucidation of its structure, for the simulated robot, is the main task of my research program. My progress is described in chapters IV and V.

I.3 System overview

The system consists of three main programs: TABLETOP, that simulates the outside world; UTAK, that simulates the robot; and PPA, the robot-controlling program.

TABLETOP simulates a frictionless tabletop with a polygonal restraining boundary. There may be arbitrary polygonal shapes on the tabletop, some fixed and some movable. These shapes constitute the objects of the outside world. The tabletop boundary will be referred to as the verge to avoid confusion later. There are never any holes in a fixed or movable object. On this simulated tabletop the everyday laws of physics hold: that is to say, the shape of an object remains invariant during motion, and if the path of a moving object is obstructed by another object or the verge then the moving object comes to an immediate standstill with a small gap between it and the offending obstruction. The concepts of mass and momentum have not been implemented, though there would be little difficulty in doing so. Consequently there are no "start up" or "slow down" times associated with robot movements, and when a wide moving object collides with an obstacle near one of its lateral extremities no terminal rotation of any kind is simulated: the object simply comes to an immediate halt.

UTAK simulates the robot, Utak[1], who is represented as a dimensionless point and is free to move anywhere there is empty space. Though dimensionless, he cannot slip between two adjacent objects which have point-to-point, point-to-edge, or edge-to-edge contact. He can grasp an adjacent movable object, and can move with and release such a object. An example task environment including Utak is shown in figure I.2.

Utak senses his environment with an eye having a limited field of view and having a variable resolution: fine in the centre (the "fovea") and progressively coarser towards the periphery. The eye may be thought of as a TV camera, suspended at the top of a stalk sticking vertically up from Utak, with the camera pointing directly downwards at the tabletop and its field of view centered on Utak. Thus the eye gets a two-dimensional view of part of the tabletop and an image of Utak always appears at the centre of the field of view.

The retinal geometry of the eye is shown in figure I.3(a). Each little square constitutes a <u>retinal field</u>, and covers a certain area of the task environment depending upon Utak's

----

[1]Rather than always referring to the "robot" and using the pronoun "it", I will usually refer to "Utak", who may be likened to a semi-intelligent dog, and use the pronoun "him". Of course, no sexual discrimination is intended. Likewise no phylogenetic discrimination is intended either: "Utak" and "him" are simply more pleasant ways to refer to what is merely an abstract device embodied in a computer program used to probe very gingerly into the principles of cognitive science.

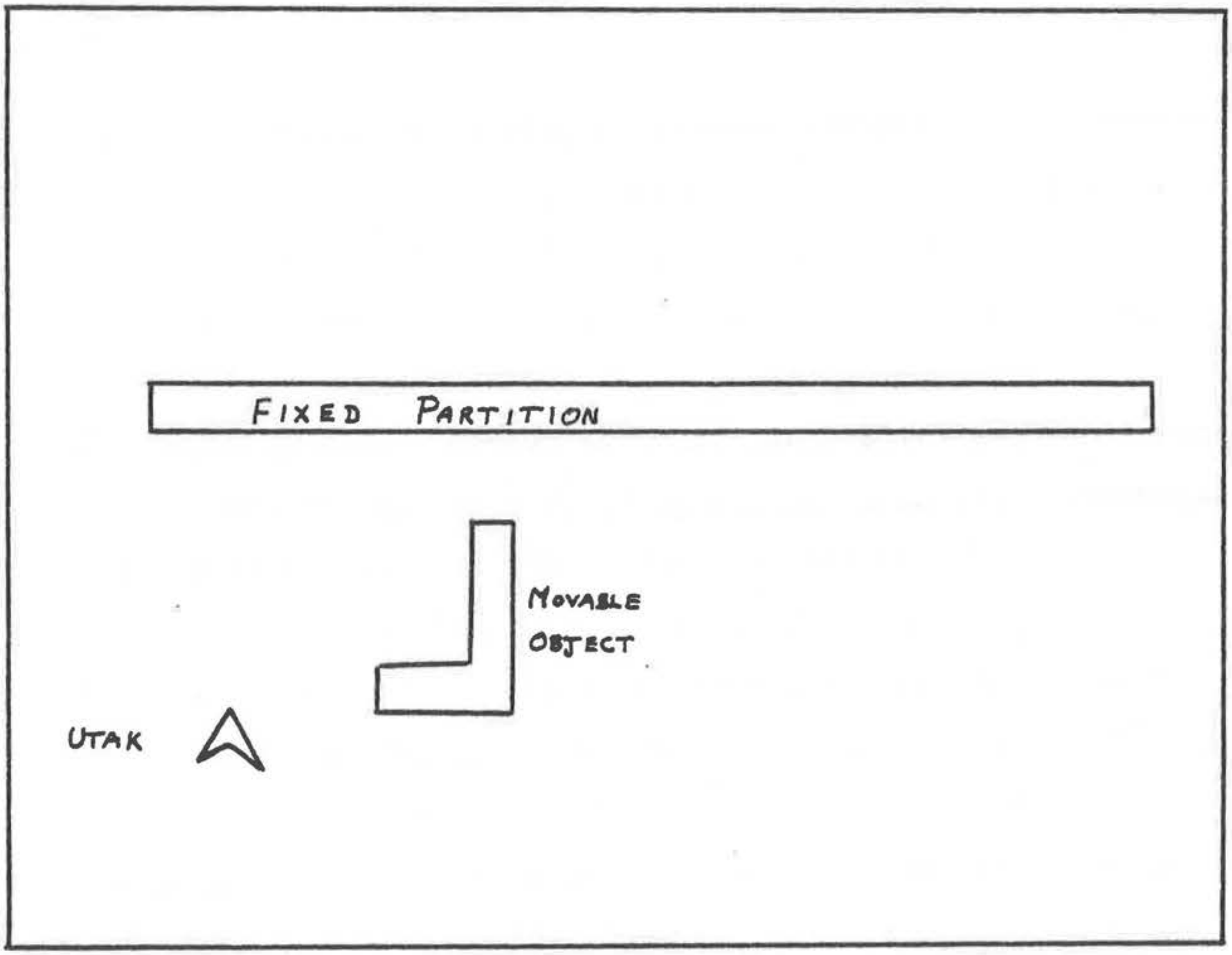


FIGURE I.2. A TASK ENVIRONMENT.

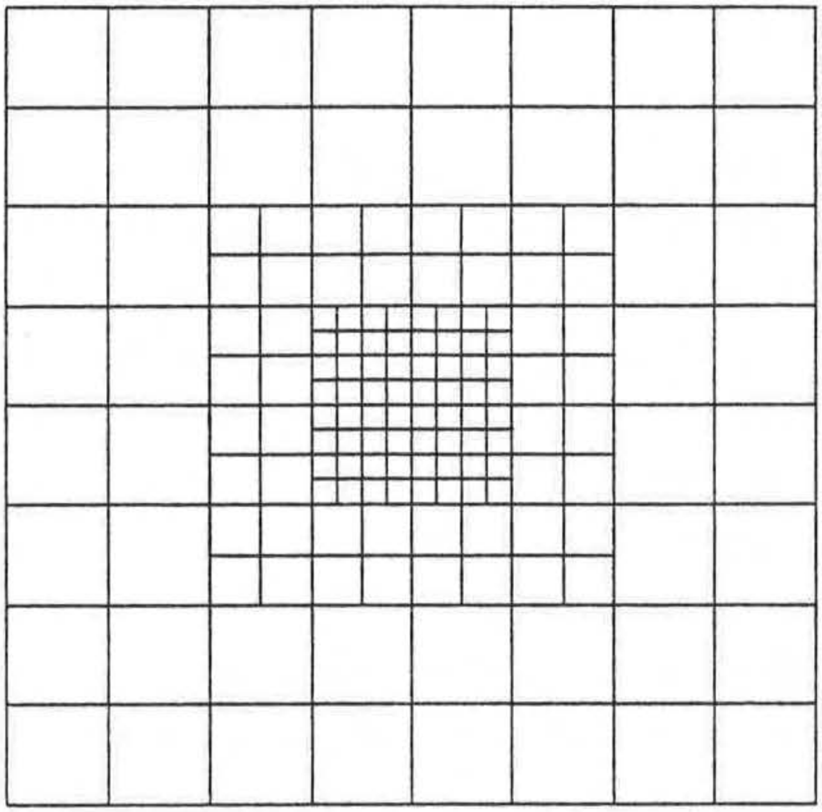


FIGURE I.3 (a) The retinal geometry.

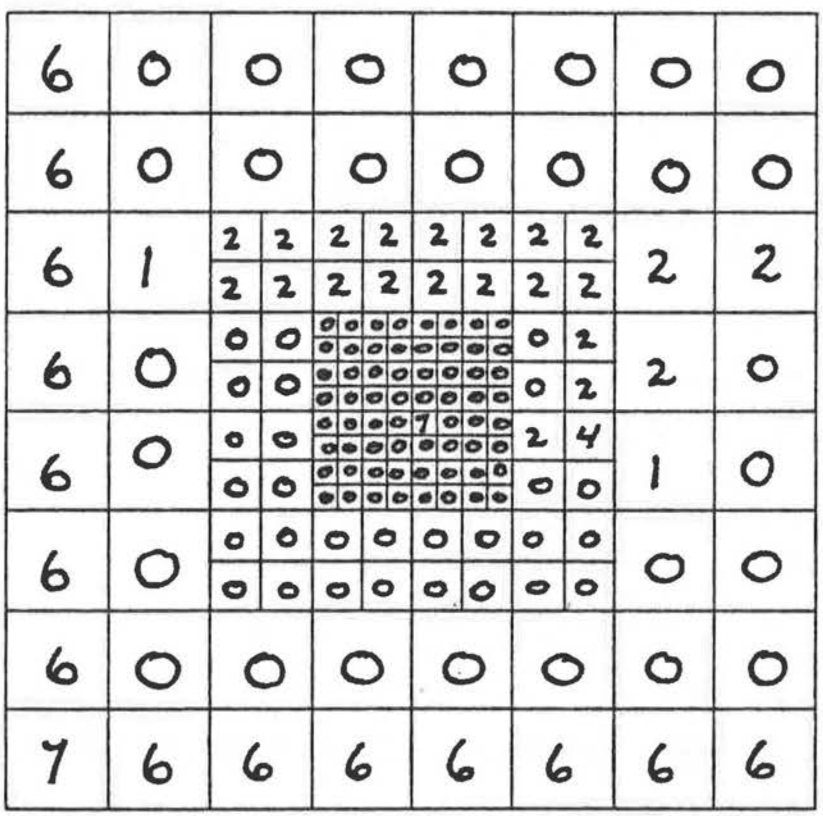


FIGURE I.3 (b) The integers in the squares form the retinal impression corresponding to Utak's position in Figure I.2.

position. Corresponding to each retinal field there is a retinal cell, which registers a graylevel, or integer in the range 0 - 7, that depends on the ratio of object to total area of the task environment covered by the retinal field. A retinal impression is the structured set of graylevels registered by all the retinal cells at one particular instant in time. The retinal impression received by Utak when in the situation of figure I.2 is shown in figure I.3(b). Utak also has eight "tactile" receptors, one in each of the eight basic compass directions, which allow him to sense the colour of an immediately adjacent object. A tactile impression is the structured set of eight colors registered by the tactile receptors at one particular instant of time.

In sum, then, Utak inhabits an outside world which may be likened to a tabletop with confining verges, where he can wander around and move objects, and where his sensory contact with this world consists of a series of retinal and tactile impressions. It is his problem to make sense of all this sensory input (James' "blooming, buzzing confusion") and create a "world model" for planning purposes. That is a major problem for Utak's brain, the robot-controlling program.

The class of tasks given to Utak consists of path finding ("Go to the north-east corner") and object moving tasks ("Push the square movable object into the next room"). The statement of a task may require considerable changes to Utak's world model. Consider, for instance, the object-moving task just mentioned. If Utak has so far seen a square movable object but has only explored what he thinks is one end of a single room, his world model before the task statement will simply consist of a room with one square movable object in it; but after "understanding" the task statement his world model will include an extra room with a doorway which connects it to the room he's currently in at a position consistent with his accumulated sensory experience to date.

PPA, the robot-controller, is divided into three parts: ACCOM, SPLAN, and ACT. PPA is an acronym for perceive, plan, act, the three parts of the action cycle. ACCOM accepts a retinal impression and modifies (accommodates) the current world model in the light of this new evidence; SPLAN is the spatial planner and is responsible for always maintaining a valid plan to achieve the current task by creating a new plan or by updating an old one; while ACT simply computes from the current plan the next action to be executed.

A world model, a task, and a plan are defined at all times in PPA, whatever Utak's actual situation, including the moment before Utak "opens his eye" and receives his first retinal impression. So far, the following defaults have been used. The world model is taken to be a large empty square centred on Utak's initial position. The default task is to explore the assumed world, which means "collect evidence (i.e. sensory input) to confirm the current world model". If the default task results in the specific task of, say, "go to the north-east corner", then the current plan would consist of walk actions to the hypothesized position of the north-east corner. Other possible defaults are "sleep" or "find food".

I have not considered problems of activation or drive. These are clearly important and involve decision-making and rewards. The archetypal problem consists of an organism hunting for food in an environment whose food-supplying characteristics are partly known. The organism is getting hungry (food or fuel running low); what is the best survival strategy for this organism? There is a large scientific literature devoted to such problems in the fields of decision theory and game theory. For me, these problems are secondary to the basic questions of representing the spatial world and solving spatial problems.

The ACCOM program, responsible for understanding incoming sensory impressions, divides into two parts, ACC-INIT and ACC-SUB. ACC-INIT accomodates the initial default world model to the very first retinal impression while ACC-SUB carries out all subsequent accomodations of the world model to incoming sensory impressions.

The spatial planner SPLAN depends on a subsystem called SHAPE to solve path-finding and object-moving problems. SHAPE makes extensive use of the world model. The basic world model is maintained in a format of points and lines specified by means of Cartesian coordinates and will sometimes be referred to as the <u>Cartesian world model</u> or the <u>Cartesian representation</u>. SHAPE functions by projecting and re-projecting all or part of the Cartesian world model onto a digital array (the screen). Path-finding and object-moving problems are solved in simple cases from one projection on the screen; more interesting cases require several projections. A projection of the Cartesian representation onto this digital array will sometimes be referred to as an image representation.

The most important part of SHAPE is the collection of algorithms for solving path-finding and object-moving problems. These are based upon the concept of the skeleton of a two dimensional shape. A more descriptive but more cumbersome term for it is the symmetric axis transform of a shape. To get an idea of what the skeleton of a shape looks like, examine the shape and its skeleton drawn in figure I.4. I found that the skeleton was a useful tool for path-finding problems, and would be a useful heuristic for object-moving problems provided algorithms could be devised to compute it. It turned out that, for reasons given in chapter V, the published skeleton algorithms were unsatisfactory. Luckily, one of these provided a good base from which I was able to derive a satisfactory skeleton algorithm.

There is a very nice mathematical definition of the skeleton of a shape. Consider the set of all circles that lie totally within the shape and partially order them by inclusion; the skeleton is then defined to be the locus of the centre of all maximal circles in this set. In addition, each point of the skeleton has associated with it the radius of the maximal circle

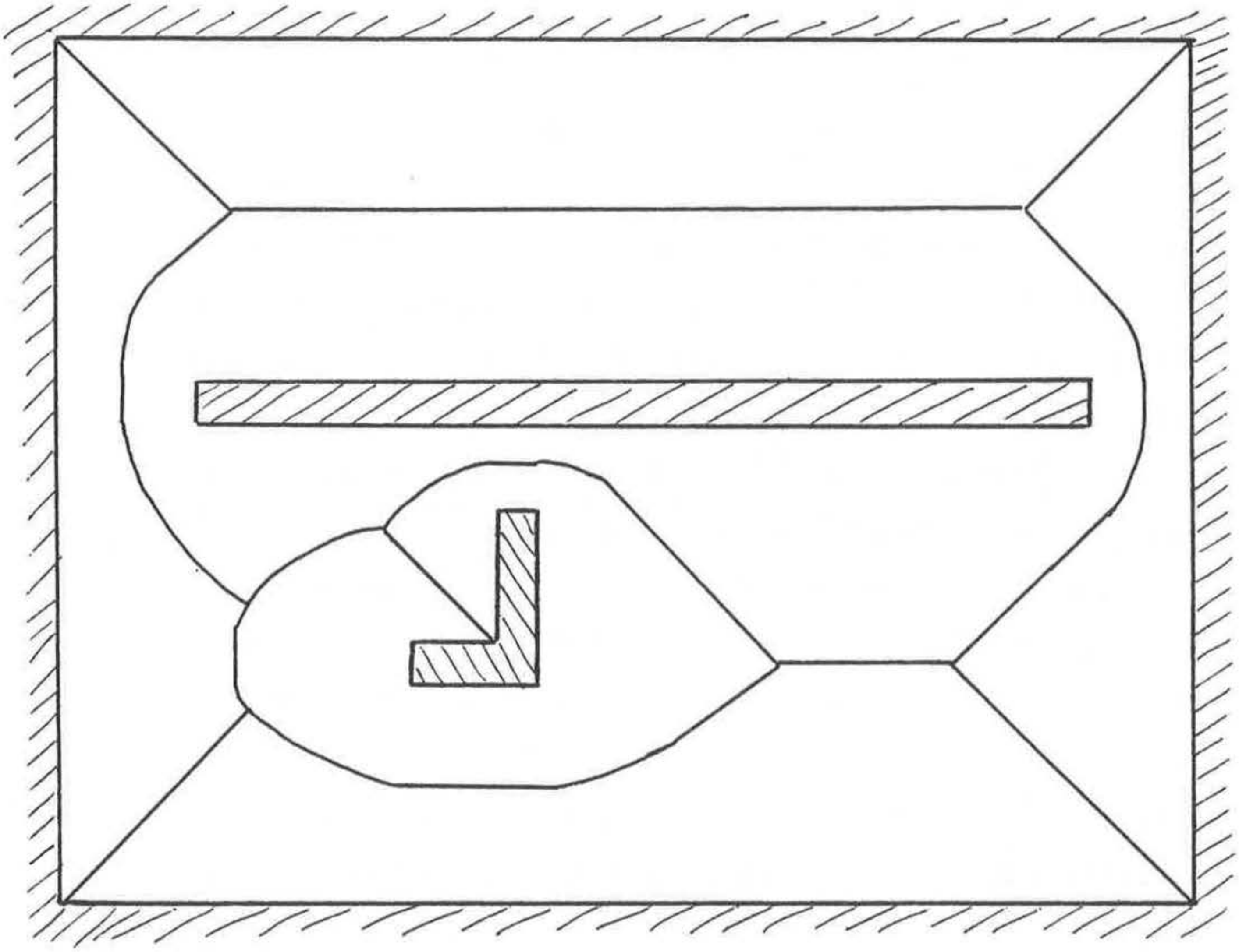


FIGURE I.4. THE SKELETON OF A SHAPE.

with centre at that point; this is known as the quench function. However, the skeleton algorithm that I use does not compute exactly the skeleton as I have just defined it. This is because the circles used in the mathematical definition are drawn using the familiar Euclidean metric for the distance between two points in a plane, whereas my algorithm uses a quasi-Euclidean metric to approximate the Euclidean distance. Using this quasi-Euclidean metric, the skeleton can be computed in a very "local" manner. Consequently I shall, when necessary, distinguish between a Euclidean skeleton, as defined above, and the digital skeleton, as computed by my algorithm. The difference between the two can be observed by overlaying the Euclidean skeleton of I.4 on the digital skeleton of figure I.5.

The skeleton of a shape is the key idea behind the spatial planner. It has a very intuitive appeal and can be used to solve more than just path-finding and object-moving problems. For instance, it can also be used to solve the following problem, otherwise known as the findspace problem [Sussman,1973]. Given a two dimensional shape that you want to put down on a cluttered surface, where do you place it? The theory and algorithms concerned with the skeleton of a shape are worked out in chapter V.

To complete this overview of PPA, the third and final component is ACT, the executive program that computes the next action for Utak from a completed plan. This is not entirely trivial because the size of the next action has to be a function

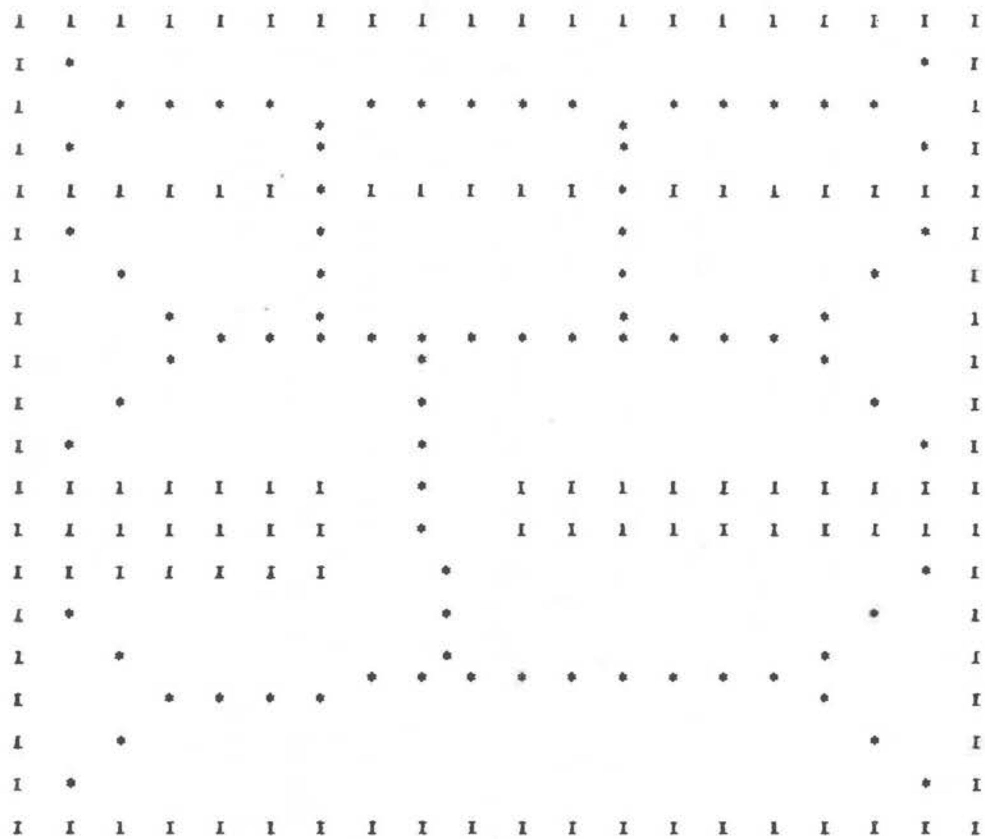


FIGURE I.5. A DIGITAL SKELETON.

of the confidence Utak has in the details of the world model in the vicinity of his current position and of the accuracy with which he can execute an action. The speed at which one runs through a room cluttered with furniture depends both on how well one can register the positions of the items and on how well one can control one's movements.

To summarize this section: I have outlined the major components of PPA, the robot-controller, and I have presented an important underlying concept, the skeleton of a two dimensional shape. The accompanying figure I.6, which illustrates this stage of PPA's design, may be regarded as a first order elaboration of the action cycle of figure I.1.

I.4 System status

All parts of the system have been designed, to varying degrees of detail, and some parts have been implemented. The TABLETOP and UTAK simulation programs, which execute actions and produce tactile and retinal impressions, have been implemented and are described in chapter III. The current status of the other parts of the system, namely ACCOM, SPLAN, and ACT, is described in full in chapter IV. Of the two subparts of ACCOM, ACC-SUB has been designed while ACC-INIT has been designed and implemented. A full implementation of the spatial planner SPLAN has not been attempted, but the overall design of SPLAN and its major subpart SHAPE is complete. SHAPE's most fundamental

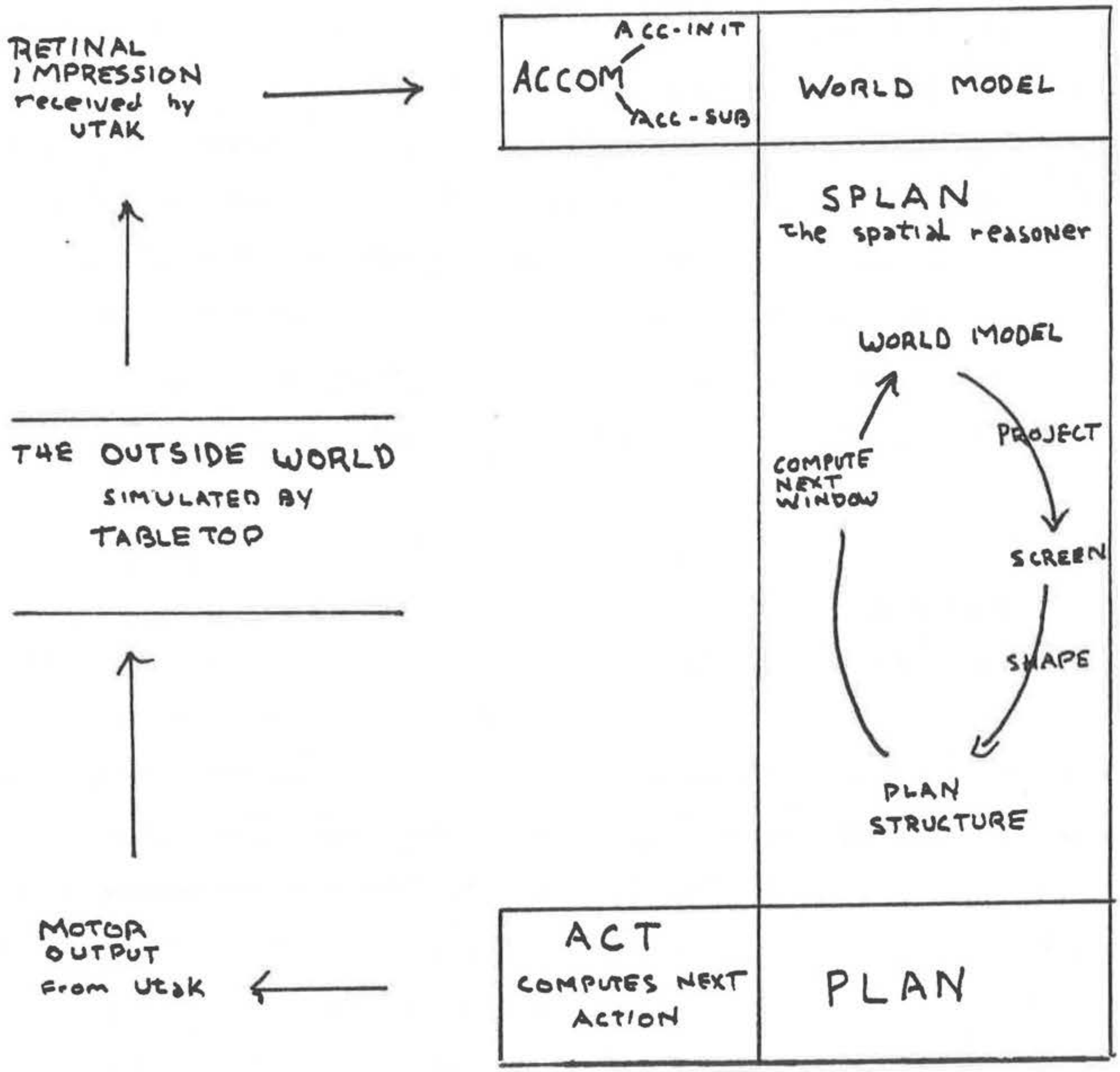

RETINAL IMPRESSION received by UTAK
ACCOM
ACC-INIT
ACC-SUB
WORLD MODEL
SPLAN
the spatial reasoner
WORLD MODEL
PROJECT
SCREEN
SHAPE
PLAN STRUCTURE
COMPUTE NEXT WINDOW
THE OUTSIDE WORLD
SIMULATED BY
TABLETOP
MOTOR OUTPUT from Utak
ACT
COMPUTES NEXT ACTION
PLAN

FIGURE I. 6

operation, the skeleton-finding computation, is complete and implemented; that is the main algorithmic contribution of this thesis.

Taking the engineering view towards AI, I make two contributions in this thesis. One is an efficient system for simulating the motion of rigid two dimensional shapes, the other is the design and partial implementation of a spatial planner that finds paths and produces plans for moving two dimensional shapes around on a flat surface.

I.5 Reader's guide

Chapter II, consisting of two parts, is concerned with background issues. The first part concentrates on giving an overall view of the whole AI enterprise while the second part reviews numerous pieces of work from AI and its sister disciplines that are closely related to my own. Chapter III describes the design considerations and algorithmic details of the simulated robot world, while chapter IV covers the whole robot-controller design. I include in chapter IV a number of task scenarios that my robot-controller, when fully implemented, is designed to be able to execute. Chapter V describes the theory and algorithms for computing the skeleton of a two dimensional shape, and its usefulness for pathfinding and object moving problems. The concluding chapter VI recapitulates the foregoing, discusses my contribution to the AI enterprise, and describes future directions of research.

The appendices include a user's manual for the TABLETOP simulation, a combinatorial lemma required in chapter V, and some proofs concerning the simulation system of Funt [1976] required in section II.4.7.

If you want to see a new iterative algorithm for computing the skeleton read section V.2.4. For a new application of the skeleton, to pathfinding, read section V.3. If you want to see algorithms for simulating the TABLETOP world read section III.1. The overall design of a robot-controller is outlined in chapter IV. Finally, if you want a general review of Artificial Intelligence, read the first three sections of chapter II. The last section of chapter II contains a literature review.

CHAPTER II

BACKGROUND ISSUES

In this chapter my purpose is to briefly sketch the nature of Artificial Intelligence and then to review related work in Artificial Intelligence and other fields.

II.1 Artificial Intelligence[1] is a science with goals and paradigms

Artificial Intelligence is the computer age expression of man's eternal urge to understand his mind and consciousness. Less poetically, it is a scientific discipline whose goals are to make computers more useful and to discover and understand in computational terms the theories and principles that underly intelligence, irrespective of whether the intelligence is displayed by man, animal, or computer. The reader will note a schizophrenic tendency here: on the one hand it is an engineering discipline - [Michie,1978] defines it as "the

----

[1]The name is unfortunate, for it is not the current end-point in a progression that goes animal intelligence, human intelligence, artificial intelligence, ...!*! ... as many a layman seems to think on first hearing the name. Neither is it concerned with computer based support systems for men in space, as one scholar seemed to think. (How about computer cognology?) Also, since one cannot denote a practitioner by the usual scheme of appending "-ist" to the subject's name, one is forced to use cumbersome terms such as "researcher in Artificial Intelligence".

pursuit of engineering goals through machine processing of complex information in terms of human concepts" - while on the other hand it is an intellectual discipline, concerned with understanding intelligence in computational terms, and so perhaps is more akin to philosophy and psychology than other areas of study.

II.1.1 The paradigms of Artificial Intelligence

If Artificial Intelligence is a science then what are its paradigms [Kuhn,1962]? As [Masterman,1970] pointed out, Kuhn used the word "paradigm" in many different senses, so we shall introduce each sense as necessary.

In a social sense, Artificial Intelligence has been around as a clearly defined group of communicating workers since a 10 person summer school was held on the subject in 1955. So Artificial Intelligence is a young science and perhaps still a bit self-conscious as a result.

As for the existence of a generally accepted view of the subject (metaphysical paradigms or "Weltanschauung"; these are what get overthrown in scientific revolutions) initially there was hardly one at all except for the basic belief that an understanding of intelligence would be achieved through computational studies. This has now been stated as the Physical Symbol System Hypothesis by [Newell and Simon,1976]: "a physical symbol system has the necessary and sufficient means for general intelligent action". Very soon the central importance of search was generally accepted, and this too has been enshrined by Newell and Simon as the Heuristic Search Hypothesis: "The solutions to problems are represented as symbol structures. A physical symbol system exercises its intelligence in problem-solving by search - that is, by generating and progressively modifying symbol structures until it produces a solution structure." Now the generally accepted core topics of Artificial Intelligence can be summarized as [Nilsson,1974]: representation of knowledge, search, common-sense reasoning and deduction, and computer languages and systems appropriate for investigating the first three topics.

Going down one level, what are some of the instrumental paradigms, or generally accepted tools of Artificial Intelligence? The computer and programming languages are, of course, the _sine qua non_ of Artificial Intelligence. There are however some more specific, widely used, tools in the AI'ers toolkit. One such is the production system style of program design. A production system consists of a collection of situation-action rules plus a scheme for choosing which rule to apply next. Any production system can be viewed as a generalization of a behaviouristic stimulus-response system. Another is the semantic net approach; here a system is designed as a data structure of labelled nodes and connecting arcs together with arc-traversal algorithms. This approach is, historically, directly derived from associationist psychology.

The last tool we will mention is the most well established of them all, with a long intellectual history behind it, and the object of some controversy: first order predicate calculus, which is taken directly from traditional mathematical logic.

What are some of the defining problems of Artificial Intelligence? As examples, anyone who uses a computer in attempting to: understand natural language, play games such as chess, control a robot, understand a TV image of a real world scene, or understand speech, is considered to be working in Artificial Intelligence.

What are some of the current hot problems in Artificial Intelligence? Here is a list, culled from [McCarthy,1977] and [Simon,1977].

- the problem of cooperating with others, or overcoming their opposition - this is a task which even the youngest infant handles very well [Donaldson,1978].
- the acquisition of knowledge [Winston,1970], [Winston,1978].
- reasoning about concurrent events and actions.
- expressing knowledge about space, and the locations, shapes and layouts of objects in space.
- the relation between a scene and its two dimensional image - this is the vision problem, currently being attacked by many workers, for instance, [Barrow and Tenenbaum,1978].
- reasoning with concepts of cause and can.
- finally, the problem of representation - what knowledge enables a system to create a representation and operators for

a new and unfamiliar problem? This representation knowledge is to be distinguished from knowledge of how to solve a problem.

In summary, I have outlined this scientific field known as "Artificial Intelligence" by describing it from several Kuhnian viewpoints.

II.1.2 AI has potentially rich relationships with many other fields

There are many relations between Artificial Intelligence and other fields of study. For example, one might expect that work on getting a machine to understand language would form a subfield of linguistics proper, whereas in fact this field of research has a somewhat contentious relationship with traditional linguistics. This relationship is the topic of an enduring, acrimonious debate - an example of competing paradigms, a la Kuhn, due to differing research programs.

As another example, psychology and Artificial Intelligence have enjoyed a somewhat lopsided relationship. The more vocal advocates of Artificial Intelligence believe that Artificial Intelligence research is of monumental importance for psychology [Minsky and Papert,1972], whereas most of the psychological community has simply ignored Artificial Intelligence. The reasons for this schism seem to stem largely from differences in methodology [Miller,1978] - another example of competing research paradigms. In order to keep his science well founded on facts, and hence scientifically respectable, the psycholcgist is concerned to produce falsifiable thecries, that is, theories with demonstrably true empirical consequences. On the other hand, the Artificial Intelligence researcher is concerned to produce falsifiable computational theories, mechanisms and computations with demonstrably true empirical consequences. The difference is that the latter is not so much concerned with the empirical facts of human mental abilities, which by and large are taken as intuitively obvious, but with the empirical facts of computation. For instance any proposed mechanism which encounters a combinatorial explosion or otherwise runs intolerably slowly on any actual or potential computer is unacceptable. Thus an algorithm that is exponentially slow in the domain of interest is unlikely to be acceptable in the long run. To conclude: Artificial Intelligence is not so artificial after all - it is grounded on the empirical, natural, facts of computation.

One might expect there to be some contact between animal behaviour studies and Artificial Intelligence, for at least two reasons. First the behaviour of animals is simpler and therefore the construction of computational processes sufficient to duplicate it should be easier. (When I say that the behaviour of B duplicates the behaviour of A, I mean that there is no significant observable difference between the behaviour of B and of A, as in the Turing test or in Bridgeman's operationalism.) Second, the evolution of human behaviour (i.e.

intelligence) can be traced through many grades of animal behaviour [Jerison,1973]. Consequently one might expect that subsystems that had already been shown to duplicate aspects of animal behaviour occurring earlier in the evolutionary record could be used as building blocks in the construction of systems that duplicate aspects of human behaviour. In fact, apart from a very few studies reviewed later, there has been essentially no contact between the two fields. In passing, let me describe a traditional problem for problem-solving systems that does come from animal behaviour studies - the monkey and bananas problem. A hungry monkey is in a room with a bunch of bananas hanging from the ceiling and a box in the corner; how does the monkey get his food? This problem was solved by Kohler's chimpanzees [Kohler,1925]; typically the chimpanzee piled up three or four boxes in a marginally stable pile to get the bananas.

Artificial Intelligence and neurophysiology have also, perhaps surprisingly, almost no relationship at present. To Artificial Intelligence researchers, neurons are simply another way of implementing algorithms. Some very early work on neural nets by [Kleene,1956] and [Moore,1956], which was based on the now outdated McCulloch-Pitts model of the neuron, branched off from Artificial Intelligence and developed into automata theory. More recent work by [Marr,1976] on vision is clearly related to the known facts about the visual cortex, while some aspects of the "Intrinsic Images" of [Barrow and Tenenbaum,1978] are very similar to the structure of the columns in the visual cortex.

The neural net idea has been developed further by a physiologist [Brindley,1969] and by mathematical biologists, e.g. [Ermentrout and Cowan,1978]. Since human neurophysiology has also evolved, the comments about evolution of behaviour in the preceeding paragraph can be taken over almost word for word, with neurophysiology substituted for behaviour throughout.

On a deeper level one might expect a rich two-way relationship between Artificial Intelligence and neurophysiology or its very close cousin, neurobiology. Take the case of vision. If certain computations are found to be sufficient for vision, then the question facing neurophysiology is "Where and how are these computations being carried out in the CNS?" Conversely, if the human visual system is found to compute certain functions, the question facing Artificial Intelligence is "Why is the CNS computing these functions?" As another example, consider the phenomenon of habituation. Habituation is a gradual decrease in the amplitude or probability of a response to repeated presentation of a particular stimulus. Habituation is ubiquitous in nature and is the simplest type of learning. It has features in common with other kinds of learning and is sometimes a component of more complex learning. Consequently, an understanding of the mechanisms of habituation could be used to build mechanisms for other types of learning. Its biological mechanisms are being slowly teased out by neurobiologists [Kandel,1978]. On the one hand, it is easy from the Artificial Intelligence viewpoint to propose many methods of implementing habituation; the obverse of this is that in Artificial Intelligence one should prefer a learning mechanism which, at some level of description, is consonant with the known facts of habituation.

There is a strong relation between Artificial Intelligence and philosophy. This is not surprising in view of the fact that Artificial Intelligence has to consider some of the major traditional problems of philosphy. In designing almost any Artificial Intelligence system commitments have to be made about the nature of knowledge, how knowledge is obtained, how it is represented and used, and the relation between knowledge and action. The connection between philosophy and Artificial Intelligence is considered by [Sloman,1978], [Dennett,1978a], [Burks,1978], and others. [McCarthy et al.,1978] have described a formalism for expressing "knowing that" and used it to solve two riddles involving knowledge about knowledge. McCarthy, in some recent papers [McCarthy,1977a,b,c], has made inroads on philosophy by approaching many traditional philosophical problems from the Artificial Intelligence viewpoint. In summary, it seems that whereas the influence of philosophy is slight, the Artificial Intelligence viewpoint promises to have an enormous influence on philosophy.

I have sketched the actual or potential interactions between Artificial Intelligence and linguistics, psychology, animal behaviour, the neurosciences, and philosophy. Thus does Artificial Intelligence tread its own well-defined path through the maze of modern science, with the potential for enriching and being enriched by many other fields of the scientific endeavour.

II.1.3 Understanding the world is a prerequisite to doing mathematics

An early dream of Artificial Intelligence researchers was to prove significant mathematical theorems. There was, it seemed, a perfect tool just waiting to be used: predicate calculus, a formal system which can express all of mathematics, and in which proofs proceed by the mechanical application of deduction rules. Put the formal system in a computer, and let it run! The ensuing combinatorial explosion was uncontrollable, and it is now clear that any direct use of a formal system in the traditional manner of mathematical logic is doomed to failure. Moral - a new tool is useless until one has learnt how to use it.

In retrospect, one can say that it was quite unreasonable to expect such a scheme to succeed. Consider the words of Hilbert, who was personally responsible for several formalizations of mathematics [Hilbert,1927]:

> No more than any other science can mathematics be founded by logic alone; rather, as a condition for the use of logical inferences and the performance of logical operations, something must already be given to us in our faculty of representation, certain extralogical concrete objects that are intuitively present as immediate experience prior to all thought. If logical inference is to be reliable, it must be possible to survey these objects completely in all their parts, and the fact that they differ from one another, and that they follow each other,

> or are concatenated, is immediately given intuitively, together with the objects, as something that can neither be reduced to anything else nor requires reduction.

Clearly Hilbert had no illusions about the use of a formal system for discovering mathematical theorems. My own intuition is that the objects of one's thought - noeses - which are being "surveyed" when proving a mathematical theorem, are essentially the same as the noeses involved in manipulating objects in the external world, or in making mundane plans for action in the world. This is said by [Kleene,1952, p.51, using a quote from Heyting]:

> "There remains for mathematics no other source than an intuition, which places its concepts and inferences before our eyes as immediately clear. This intuition is nothing other than the faculty of considering separately particular concepts and inferences which occur regularly in ordinary thinking."

Consequently my guess is that no really significant achievements in mechanical theorem proving are likely to occur until we know how to get machines to handle the real world.

It is well known that there are problems with using a formal system to model mathematical truth. One possible approach might be to use two formal systems, each of which can refer to and hence approximate the other. In one direction this leads to a reconsideration of Minsky's "models of models" problem [Minsky,1969,p.426], in another direction to practical proposals for representation theory. [McCarthy,1977a,p.5] suggests how an approximating formal system might be constructed.

II.1.4 A theory of intelligence will be primarily concerned with representations of the world

As is already clear, any theory of intelligence will above all be concerned with representations. At a very gross level, [McCarthy and Hayes, 1969] classify representations of the world by their adequacy. A representation is called *metaphysically adequate* if the world could have that form without contradicting the facts of the aspect of reality that interest us. For instance, a quantum theorist could, in principle, represent it by a giant quantum mechanical wave equation. But such a representation cannot even express a practical fact such as "this book is red". A representation is *epistemologically adequate* if it can express all the practical facts about the world. First order logic - a formal system - is a candidate epistemologically adequate representation. It can express propositional knowledge but fails on some other kinds of knowledge, such as notions of cause and ability. A representation is *heuristically adequate* if it represents the practical facts and can be used to compute answers to problems. Only representations of the world that are potentially heuristically adequate are of direct interest to Artificial Intelligence, and henceforth I consider only these.

The amount of search involved in solving a problem depends critically on the representation of that problem. For example

[Amarel,1968] considered the Missionaries and Cannibals problem [M&C] and worked through several different representations. The most powerful representation could solve a considerably more general problem than the original problem, and the solution to the simplest M&C problem dropped out of it with almost no search at all. The question arises, how could a system be programmed to find a new representation for a problem, in which the solution will be found with only a little search? This is what happens when one "sees" the solution to a problem. It could be said that this is merely moving the focus of the search from finding a solution of the problem to finding a good representation of the problem. However, the advantage of a good representation is that it may be applicable to many other problems. Moral - develop as many different viewpoints as possible. Humans have a remarkably good representation for the three-dimensional world surrounding us: the result of an eons-long evolutionary search.

Any competent problem solver will have access to several representations for a problem. [Minsky,1975] cites the example of an auto-mechanic repairing a car, who uses electrical, mechanical, and visual representations to solve a problem. We are also endowed, through evolution, with several representations of the outside world, witness the different parts of the cortex devoted to visual, auditory, and tactile representations of the world. There is also psychological evidence for these multiple representations. For instance, [Posner,1972] presents evidence based on reaction time experiments for the existence of distinct representations corresponding to different modalities. [Bower,1974] suggests that an infant is born with an wholistic, multimodal object concept which in the course of development differentiates into many distinct representations. For a problem solving system, the question is, what interactions should occur between different representations? This is largely an unexplored question.

Minsky suggested that an intelligent system be organized as a collection of interacting schemata[2]. A schema consists of a bundle of "slots", one for each member of a collection of closely related features. In the absence of evidence to the contrary, the slots of a schema assume default values. A schema is also likened to a mini-theory for a small part of the world. If one wishes to handle schema theory in first order logic, it might be worth pointing out that the relation between a schema and its default values is similar to the relation between a formal system and its standard model in logic, just as the integers are the standard model for any formalization of arithmetic.

In view of the requirement of multiple representations for

----

[2]Minsky used 'frame', but I prefer to follow [Simon,1977], who pointed out that 'schema' is a more appropriate term, for two substantial reasons. First, the term has been widely used in the AI and psychological literature in the sense with which it was introduced by Bartlett in 1932; second, 'frame' already has a well-defined technical meaning in the AI literature.

a problem solver, Minsky's schema theory should perhaps be augmented by allowing every schema to have many different representations. This might be done as follows. Each small aspect of the world may have distinct verbal, visual, auditory, tactile or olfactory schema. There are associations between verbal schema, between visual schema, etc.; in addition the verbal schema for one small aspect of the world may evoke its visual schema, which may evoke its auditory schema, etc.

To summarize this section: any functioning robot-controller must use a heuristically adequate representation of the world, that is, a world model which reduces to a minimum the search time required to produce a plan or to solve other frequently encountered problems.

II.1.5 A theory of intelligence will describe intelligent systems at many different levels

A closely related issue concerns how to describe a complex information processing system. Marr and Poggio[1976] argue that the information processing of a system such as the central nervous system needs to be understood at four nearly independent levels of description:

(1) that at which the nature of a computation is expressed;

(2) that at which the algorithms that implement a computation are characterized;

(3) that at which an algorithm is committed to particular

mechanisms; and

(4) that at which the mechanisms are realized in hardware.

In general, the nature of a computation is determined by the problem to be solved, the mechanisms that are used depend upon the available hardware, and the particular algorithms chosen depend upon both the nature of the computation and on the available mechanisms.

For example, consider the Fourier transform. The theory of the Fourier transform is well understood, and is expressed independently of the particular way in which it is computed. One level down, there are several algorithms for implementing it. For instance, the Fast Fourier Transform, which is a serial algorithm based upon the mechanisms of the digital computer, and the algorithms of holography, which are parallel algorithms based on the mechanisms of laser optics, can both be used to implement the Fourier transform. The meta point to be made about describing a complex system, is that while the gory details of algorithms and mechanisms are of great importance, the essential thing is to understand the nature of the computation enforced by the problem that is being solved.

\+ \+ \+

To summarize this section, I have:

- briefly outlined Artificial Intelligence by examining it from several Kuhnian viewpoints;

- argued that it is necessary to know how to understand the world before one can hope to know how to do more sophisticated tasks such as mathematics;
- pointed out that any theory of intelligence must be primarily concerned with representations of the world, and secondarily will describe any intelligent system in many different ways.

II.2 Simulating a robot is a promising approach to Artificial Intelligence

A lot of work in Artificial Intelligence is devoted to problems which people find intellectually challenging, that is, problems which require extensive use of one's conscious reasoning abilities. Thus almost by definition they are problems that people are not in general good at. But there are many problems that people solve easily and unconsciously every day - and therefore solve well - and it is the principles lying behind the solution of these "easy" problems which are of most fundamental interest to any budding theory of intelligence. So one has the somewhat paradoxical conclusion that only those problems that are intellectually uninteresting to one's conscious awareness are of fundamental interest to Artificial Intelligence. But therein lies the greatest hope for optimism, for then it is much easier to be objective about the subject matter and not be led astray by intuitive ideas about the functioning of one's own consciousness, a devilishly fallible source of guidance. After all, the greatest scientific progress has occurred in the "hard" sciences, where it is very easy to be objective about the subject matter. Vision and speech-understanding are examples of "uninteresting" problems which are now major subfields of Artificial Intelligence.

Since the whole human mind is such an impressive and unfathomable a phenomenon, most work in Artificial Intelligence has been devoted to some small aspect of it. But it is quite likely that there are basic principles of intelligence involved in how the various aspects, e.g. perception, memory, planning, action, or speech, are woven together. Further, these might reasonably be expected to appear in simpler organisms in simpler form. Thus the complete simulation of some very simple organism would be a worthwhile study in Artificial Intelligence, for instance, a simulation of a starfish, or crayfish, or turtle. However there is a problem here, in that even though a great deal is known about many aspects of many different organisms, no-one has put together all the information about one single organism. It should be added that there are simple creatures, most notably <u>Aplysia</u>, or sea hare, which have been extensively investigated by neurobiologists [Kandel,1976] and would therefore be good candidates for the first serious simulation of a complete living creature. The other suggestion is to invent some simpler world and organism and work out all the details of the organism-controlling program. Toda[1962] carried this out from the psychologist's point of view, and more recently Dennett[1978] suggested this in the context of a philosophical critique of Artificial Intelligence. This is the path I have followed, with emphasis not so much on problems of control but rather on problems of spatial representation.

This approach involves methodological problems. One has to make many arbitrary decisions in constructing both the simulation and the organism-controlling program. In the simulation, the world and the robot should on the one hand satisfy some criteria of "naturalness" or "animal-like-ness", and on the other hand be computationally feasible to simulate and cheap enough that extensive experimentation can be done to observe the performance of the bug using various organism-controlling programs. The design of the organism-controlling program should on the one hand reflect ideas derived from observation of actual creatures, from the analysis provided by psychology, as far as it goes, from studies in animal behaviour, and from intuition based on one's own introspection, while on the other hand it cannot avoid being constrained by the material being used to construct it, namely the architecture of the machine in which the program runs, and by the concern that it, too, should be computationally feasible to run. In fact the most one can do is to settle upon some arbitrary design for the robot world based on some unstated criteria of "naturalness" and proceed with the design of the controlling program.

And when that is designed and built, how is it to be judged? Again, given an arbitrarily designed simulated robot world this is, strictly, an impossible task since there is nothing to compare it with. One has to rely on intuitive notions of "naturalness", "interest", or "elegance". This position would be improved if two or more different designs for the organism-controlling program were built, for then at least an inter-design comparison could be made. Or even better if the simulated robot world could be seriously compared with an actual organism in its natural environment, as proposed above, for then the performance of the controller could be compared with the behaviour of a real organism.

And finally, even if some interesting principles are uncovered in the course of building a whole series of controlling programs for various simulated robot worlds, there is always the possibility that a qualitative discontinuity principle is at work which would say roughly that the simulated robot worlds used are so much simpler than the whole human/environment system that no interesting principle true at the level of complexity of the simulated robot worlds is going to carry over to the highly complex human/environment system.

In conclusion, the road to knowledge via robot simulation studies is strewn with methodological potholes; but the route is obvious and promises to lead to new vistas!

II.3 The current AI tradition for the design of planning and problem solving systems is not easily adaptable to my purpose

My research is, in part, a reaction against the usual AI approach to the design of planning and problem-solving systems. This approach is so widespread that it may justifiably be called a tradition. Extending the terminology of [Sloman,1971], we call it the *Fregean tradition*. The series of programs LT [Newell, Shaw, & Simon,1957] - GPS [Newell & Simon,1963] - BLOCKS [Winograd,1972] - STRIPS [Fikes, Hart, & Nilsson,1972] - HACKER [Sussman,1975] - NOAH [Sacerdoti,1977] - EL [Stallman & Sussman,1977] - DESI [McDermott,1978] TMS [Doyle,1978] epitomizes this tradition. One's everyday behavior is intimately related to one's ongoing perception and actions, yet these systems say nothing about perception and only very little about action, i.e. executing a plan. The purpose of this section is to amplify and justify this complaint, suggest a remedy, and show why, temporarily, I shun this tradition.

II.3.1 An exegesis of some AI planning and problem-solving systems

LT, the logic theorist, was given the task of proving the first 52 theorems in the *Principia Mathematica* of Whitehead & Russell. All these are theorems in the sentential calculus. To generate subproblems it used substitutions, detachment, and forward and backward chaining, and to reduce the search space size it used matching and similarity tests. It proved 38 of the 52 theorems and failed on 14. Most of these failures were due to time and space limitations.

LT is, historically, very important in AI. The techniques introduced in LT are widely used in AI and are incorporated into most modern AI programming languages. Its importance lies, however, not so much in the techniques invented by Newell, Shaw, and Simon, - these are undeniably important technical contributions - but rather in the type of problem attempted, and in the type of solution which was found acceptable. In Kuhnian terms, Newell, Shaw, and Simon established a paradigm which was followed by mainstream AI for the greater part of a decade, and which still casts a significant shadow in the current AI scene.

The following year [McCarthy,1958] published a proposal for an Advice Taker program. This program was to be able to reason verbally and be able to accept advice. To illustrate its functioning he considered the everyday type problem of constructing a plan to get from the desk in one's home to the airport. The basic idea was that for a program to be capable of learning something it must first be capable of being told it. That may seem like a respectable basis, but there are reasons to believe it is not quite the right way to develop an intelligent system. One often becomes very competent at some skill without being able to express it in words or being able to accept verbal advice about it. Verbal expression of a skill comes after, not before, the acquisition of competence at the skill. To give a very personal example, I have two daughters aged 6 and 8 who can now ski proficiently - yet they have been told, verbally, nothing about technique; their only instruction has consisted of having their hands held for several hours on beginners' slopes.

The Advice Taker proposal influenced several later programs. The program of [Black,1964], the program QA3 of [Green,1969], the MICROPLANNER language of [Sussman, Winograd, & Charniak,1971], and STRIPS all leaned heavily on the Advice Taker. Indeed the functioning of MICROPLANNER closely follows the outline on pp.406-409 of [McCarthy,1958].

GPS, another landmark in Artificial Intelligence [Newell and Simon,1963], is a program whose design goal was to simulate human thought. It handled a variety of intellectual tasks, such as the missionaries and cannibals task, some integration problems, proving theorems in the first-order predicate calculus, and the monkey and bananas problem. GPS deals with a task environment consisting of <u>objects</u> which can be transformed by various <u>operators</u>; it detects <u>differences</u> between objects; and it organizes the information about the task environment into <u>goals</u>. Each goal is a collection of information that defines what constitutes goal attainment, makes available the various kinds of information relevant to attaining the goal, and relates the information to other goals. There are three types of goals:

Transform object A into object B,

Reduce difference between object A and object B,

Apply operator Q to object A.

Basically, GPS achieved a goal by using a means-ends heuristic to recursively set up subgoals whose attainment would lead to the attainment of the initial goal.

Meanwhile there was another line of work emanating from studies in mathematical logic. The contributions from [Gilmore,1960], [Prawitz,1960], [Davis,1963], and others, all aimed to mechanize mathematics. This effort was consolidated by the *resolution principle* of [Robinson,1965].

At its simplest, in propositional logic, the resolution principle says that from the two clauses AvB and ¬AvC one may deduce BvC. BvC is called the *resolvent* of AvB and ¬AvC. The full resolution principle may be succintly described in terms of this example as follows. Generalize it by allowing extra disjunction and predicate symbols and lift it to the first order predicate calculus, so that free variables may appear as arguments of the symbols A, B, C,.... Introduce a matching algorithm - known as *unification* - to compute substitutions for variables such that the arguments of A in the first clause become identical with the arguments of A in the second clause, if this is possible at all. The resulting rule of deduction is the resolution principle, and can be proved to be complete for the first order predicate calculus - that is, any provable well formed formula (*wff*) can be deduced by sufficiently many applications of the resolution principle.

The resolution principle thus reduces mechanical theorem-proving to one rule of inference which subsumes, by the mathematically elegant unification algorithm, the substitutions, matching, and similarity tests of LT. However, there remains considerable choice in deciding what pair of clauses and what pair of predicate symbols (technically, literals) to resolve together. The question of search strategy was studied by [Kowalski,1969]. He derived search strategies for resolution that generalized the A* algorithm of [Hart, Nilsson, & Raphael,1968], and stated conditions under which these strategies were admissible and optimal. However, his strategies are independent of the semantics of the clauses and literals under consideration, and consequently the search space is not significantly reduced despite the mathematical elegance of the strategies. Hopes of being able to control the size of the search space rose when the programming languages MICROPLANNER, CONNIVER, and QA4 appeared. In these, it is possible to recommend that, in trying to prove a wff of a certain type, other facts of certain restricted type should be tried first. However this facility does not, in general, sufficiently reduce the search. [Reiter,1972] proposed the use of models in theorem prover to help control the search, basing his approach on the geometry theorem proving machine of [Gelernter,1959]. His proposal was to present to the theorem prover a model of the axiomatic system involved. In addition he proposed a set of procedures for extracting information about the model when required by the theorem prover, and a flexible, general, interface between such a semantic subsystem and the purely syntactic logical system. So far, this has not led to any startling breakthrough. There still seems to be no generally accepted way of using semantics to control the size of the search space in a theorem prover. In summary, the resolution principle may be somewhat negatively characterized as elegantly exposing the combinatorial explosion which seems to be inherent in any straight-forward attempt to do mechanical mathematics based on Fregean formalisms.

The _frame problem_ arises whenever a theorem prover is used to reason about actions. This was first done by [Green,1969] in the QA3 program, a resolution theorem prover. Suppose you have a system of axioms that describes a situation in the world - a world model - and perhaps have deduced some facts about this situation. For example, if A, B, and C are blocks, two axioms might be (ON A B) and (ON B C), and an obvious deduction is (ABOVE A C). If the effect of an action is modelled by changing the system of axioms then after the action one cannot be sure, formally, which of the previous deductions are still true and which are now false. One seems to need axioms saying that certain facts remain unchanged when the effects of an action are modelled. This is exactly what Green did. Every predicate had an extra argument position for a state-variable - a _situational fluent_ in the terminology of [McCarthy & Hayes,1969] - which assumed a new value whenever an action was executed in the world model. An action was modelled by at least two axioms. One axiom described the direct effects of an action and the others said, loosely, that those axioms describing attributes of objects of the world model that are not directly affected by the action, are still true after the action. However, between the multiplicity of axioms describing the world model, the axioms describing both the effects and non-effects of each action, and the inherent inefficiencies of a resolution theorem prover, QA3 was only able to handle the simplest of problems. In any but the most trivial tasks, it would be quickly overcome by the combinatorial explosion.

Let us call MICROPLANNER, CONNIVER, and QA4 the procedural languages. They represent an advance over QA3 as follows. They do not use state variables. In CONNIVER and QA4 each collection of axioms that describes a particular situation in the world is maintained as a context, and whenever the effects of an action are modelled, a new context is sprouted from the old. So far, the only devices used for modelling the effects of an action - in other words, for sprouting a new context - have been the addition and deletion of facts (axioms) from a context, even though more powerful methods are available in the procedural languages.

STRIPS [Fikes and Nilsson,1971] can be described as the successful marriage of GPS and the theorem proving approach. The problem space consists of an initial world model, a set of operators which affect the world model, and a goal statement. STRIPS attempts to find a sequence of operators which will transform the initial world model into a model in which the goal statement is true. A world model is represented as a set of wffs in the first-order predicate calculus. In the robot problems to which STRIPS was initially applied, an operator corresponds to an action routine whose execution causes changes in the surrounding real world. An operator consists of a precondition wff, which must be satisfied in a world model for the operator to be applicable, and a function which describes how the world model is to be changed when the operator is applied. This function is specified by two lists, the _add_ list and the _delete_ list. The effect of applying an applicable operator to a given world model is to delete from the model all those clauses specified by the delete list and to add all those clauses specified by the add list.

STRIPS begins by applying a resolution theorem prover to attempt to prove that the goal wff G0 follows from the initial world model M0. If the proof succeeds then G0 is trivially true in the inital world model. Otherwise the uncompleted proof is taken to be the "difference" between M0 and G0. Next, operators that might be relevant to "reducing" this difference are sought. These are the operators whose effects on world models would allow the proof to continue. The precondition wffs of the relevant operators are then taken to be new subgoals, and STRIPS is applied recursively to these. A search strategy is used to control the order in which relevant operators are applied. STRIPS terminates when a sequence of operators has been found which transforms M0 into a world model in which G0 is true.

STRIPS was later extended by storing generalized plans in a triangle table format [Fikes,Hart,Nilsson,1972]. This was used in two ways: to allow similar problems to be solved without re-planning, and to assist in monitoring the progress of the robot in the course of executing the plan.

An interesting question arises concerning the abilities of STRIPS. There are problems STRIPS can solve and there are very similar problems STRIPS cannot solve; at the same time STRIPS has a perspicuous structure. This naturally suggests the question: is there an interesting and useful way to characterize those problems - world model plus goal wff - actually solvable by STRIPS?

HACKER [Sussman,1975] is also concerned with producing plans but works by a process of debugging almost right plans, or skill acquisition. HACKER is endowed with several databases of assertions. One contains all the BLOCKS world knowledge required in the course of solving a BLOCKS type problem, while others contain information about programming techniques, types of bugs, types of patches for bugs, and techniques for summarizing bugs. HACKER starts with a dumb initial trial procedure for the task. The trial procedure executes, and if it fails a "process model" of the state of the computation at the point of failure is constructed. HACKER then examines this to discover why the procedure failed. That is to say, HACKER attempts to classify the bug into one of several known types. If the attempt is successful then HACKER's built-in knowledge about bug-types is used to propose a modification to the trial procedure. The process of "trial and patch bug" is then repeated, iteratively, until a satisfactory procedure is obtained. If an attempt to classify a bug fails, then HACKER basically resigns. Otherwise, HACKER ends with a fully debugged procedure that can successfully solve any of a certain general class of blocks world tasks. Loosely speaking, HACKER compiles a procedure from a database of all the necessary facts and advice.

The important contribution of HACKER was not its planning ability - it wasn't good - nor even its learning ability - which was of a distinctly new type - but the technical idea of retaining the reasons why a certain action was performed or why a new piece of code was added to a procedure. This is the idea underlying the dependency-directed back-tracking of the system EL of [Stallman & Sussman,1977], and was developed further in the TMS system of [Doyle,1978].

Neither STRIPS nor HACKER could obtain the optimal solution to the following problem in the BLOCKS world. There is a tabletop and three blocks A, B, C. A and B rest on the table and C lies on A. The goal is to build a tower A on B on C, with C on the table. These systems fail because the goal is stated as (AND (ON A B) (ON B C)), and both STRIPS and HACKER proceed to attack each subgoal independently. Achieving either of the ON subgoals interferes with achieving the other. If you first put A on B, you can't put B on C; if you first put B on C (which is on A), you can't put A on B. This is an example of interacting subgoals.

[Sacerdoti,1975], [Tate,1975], and [Warren,1975] all wrote systems to handle such problems; I will briefly sketch NOAH, Sacerdoti's system. NOAH builds a network of goals and subgoals, represented as a procedural network. The subgoals required to achieve a goal are stored in a partial order; a temporal order is imposed only when necessary to resolve a conflict between brother subgoals. NOAH constructs a plan to achieve a goal in a layered fashion by expanding one subgoal at a time, keeping a careful watch for possible interactions, until primitive actions are reached. In this way a fully detailed plan is constructed before execution begins; errors are handled by re-planning to achieve the failed subgoal and patching the new plan into the original procedural net. To summarize: NOAH is a very elegant system which represents a current peak in the technology of planning systems for a BLOCKS type world.

II.3.2 Criticisms of the Fregean tradition in planning and problem-solving

The AI tradition, based upon Fregean formalisms, can be criticised on two levels: one is purely technical, the other is philosophical. On the technical level there are at least three criticisms. First, there is the difficulty encountered in reasoning about actions. This is the frame problem, described previously. Second, there is the difficulty encountered in handling a continual incoming stream of possibly contradictory facts, an ability required of any organism that receives sensory input from a changing outside world. This might well be termed the accommodation problem: how to accommodate a database of axioms to an incoming stream of evidence about the perpetually changing outside world. Third, there is no known semantics for a changing database of axioms - Tarski-Kripke semantics only apply to static axiom systems. In addition, if one is interested in reasoning about the natural numbers -- which is presumably the case if one is trying to automate mathematics -- it would be well to recall the well-known fact that no Fregean formal system can fully capture the concept of the natural numbers. Lastly, there are many difficulties encountered in trying to reason about causes, abilities, and knowledge about knowledge in a Fregean formalism.

On a more philosophical level, the act of writing down a Fregean formula implies an attempt to capture a timeless, actionless aspect of the world; yet in AI one is above all concerned with action and change. It's as though the "dimension" of a Fregean formula is of the wrong type for the problem being tackled - just as in physics, dimension theory demands that the dimension type of a formula match the dimension type of the phenomenon described by the formula.

At this point I must call a halt. A continuation of this line of argument leads to deeper waters[3] than I care to enter at this point, and, to do it justice, would take far more space and time than can be afforded in this thesis.

In developing a robot-controller one is, primarily, concerned with reasoning about actions and with continually accommodating the world model to the sensory input stream of evidence; secondarily one desires computationally efficient, or at least tractable, algorithms for carrying out these processes.

Throughout the exegesis I pointed out that, in effect, the combinatorial explosion has not been brought under control. For some special proof procedures, [Cook & Reckow,1974] and [Tseitin,1968] have given this a more precise statement. Without introducing any special terminology, their result - theorem 10 in [Cook & Reckow,1974] - can be re-stated as follows:

> For infinitely many n, there exists a theorem with n clauses for which the number of steps in its shortest proof is at least exponential in (log(n) squared).

Thus one may conclude that the evidence, so far, from studies of complexity suggests that the computational requirements of

----

[3] Because it leads to the conclusion that the metaphysics of Platonism, as found in the philosophical tradition which starts with Plato and continues with Descartes, Kant, Frege, Russell, and modern analytic philosophers, is suspect. A new metaphysics can be based on the notion of "process" as in Whitehead's Process and reality. This is part of another great tradition, largely ignored by modern philosophers, which can be traced from Aristotle through medieval philosophers to Bergson, Whitehead, Husserl, and others.

resolution theorem-proving are of an intractable nature.

There is, however, an important open problem here. As already mentioned, [Kowalski,1969] derived heuristic search algorithms for theorem-proving that were generalizations of A*. But [Martelli,1977] analyzed the worst case behaviour of A*, found it was 2**n, and replaced A* by a new algorithm B whose worst case behaviour was n**2, a significant improvement. The obvious open question is: can Martelli's analysis and improvement of A* be carried over to Kowalski's search algorithms?

In conclusion, I hope that the knowledgeable reader has some notion of why I feel that the Fregean tradition in AI planning and problem-solving systems is, perhaps, on the wrong tracks, and consequently can understand why, in my research, I have chosen to take another approach.

II.4 A survey of closely related topics

The purpose of this section is to provide a fairly comprehensive survey of closely related work, and a brief description of two related topics, namely imagery and behavioural theories.

I start with the literature on analyses and simulations of organisms. There are many such studies, all more or less independent, and each with its own particular orientation. I have tried to classify them according to their emphasis, but no mutually exclusive classification seems possible. The headings I have chosen are:

- functioning robot simulations;
- analyses of simple organisms, without simulation;
- studies based on animal behaviour;
- applications of decision theory;
- cognitive maps.

The inclusion of cognitive maps here may seem a little out of place, but a moment's consideration, of the fact that all such studies are concerned with how an animal or man finds its way around its environment, shows that it is quite appropriate.

I then proceed to the literature on spatial representation and reasoning. This falls easily into two classifications:

- spatial planners conceived as potential tools for architects and others;
- systems for simulating the motion of rigid bodies.

There is, in addition, one published system for path-finding [Thomson,1977], which I do not include here since it is more appropriately covered in my section V.1 on path-finding. Similarly I do not review the literature on the skeleton here since that is done in section V.1.

II.4.1 Previous robot simulations

[Nilsson & Raphael,1967] simulated a robot and its environment in order to study the key problems in designing and controlling a robot. Their later design of Shakey, the SRI robot, was based on this preliminary exploration. Their simulated robot resides on an arbitrarily large checkerboard containing both movable and nonmovable objects. The robot can move forward, turn right or left, and sense when it "bumps into" an object. It stores information about the location and properties of objects in its environment and uses its sensory inputs to establish, correct, or update this information. The robot can make specified changes to its environment by pushing the appropriate movable objects.

The design of the simulated system contained several important basic features that any real robot in a much richer environment would need. These include the robot's model of its environment, a problem-specification language for communicating with the robot, a heuristic problem-solving program, and a robot executive program for overall control. The tasks consisted of "goto" and "pushto" problems. Plans to solve these tasks were constructed by using Moore's maze-solving algorithm on the array of locations. The robot could sense the contents of the square immediately in front of it, and use this to correct the world model.

Of the published studies that I know of, theirs is the closest to mine in terms of overall aims and design. However, my simulated world, Utak's sensory equipment, and Utak's robot-controller are all more sophisticated than the corresponding parts of their system.

Becker and Merriam ([Becker,1972], [Becker & Merriam,1973], [Merriam,1975]) simulated a robot cart in a two dimensional world which used a sophisticated eye with a fovea to pick up information about its surroundings. Initially a city street environment was used but subsequently a "Martian" landscape was used. This eye could either gather coarse information from a large area or could "zoom" down and obtain detailed information from a small area, and could change its focal point. Thus the eye could be used for two conflicting tasks: keeping a lookout for new objects, and focussing down on one object to get more detailed information. This conflict was resolved by the eye-controlling program which took into account such factors as drive, salience of an object, progress, effort..., and which produced a natural-looking scan path when looking at a street scene. The design of the eye-controlling program was unfortunately not specified. The eye could also track a fixed object when the robot moved. The later simulation of the environment took into acount the finite size of the robot chassis and simulated the visual occlusion of, for instance, Martian hills by Martian mountains. A long term memory was used which stored no spatial information.

Theirs is a more sophisticated simulation of the world and a different eye, but no design of the robot executive, or report of the simulated robot executing a goto or pushto task, appears to have been published.

II.4.2 Three analyses of simple organisms

Each of these analyses approach the behaviour of an organism from a distinct point of view. Simon's paper is a game theoretic analysis of the survival of an organism in an environment in which he derives one equation relating organism to environment; Toda's paper is in the same vein but uses decision theory; while Becker is concerned with the structure of a representation for external events and how this structure should develop over time.

[Simon,1956] considered a simplified organism with circular vision, with a single need - food - and only three kinds of activity: resting, exploration, and obtaining food. It has to survive on a plane with isolated point sources of food. He derived an equation showing how the chances of survival of the organism depended on four parameters, two describing the environment and two describing the organism, assuming the organism behaved in the obvious "rational" way. Thus he found that an organism in its natural environment requires only very simple perceptual and choice mechanisms to satisfy its several needs and to assure a high probability of its survival over extended periods of time. He also showed how multiple goals could be satisfied with a very simple choice mechanism. This analysis was achieved without the use of utility functions as in decision theory. Simon's analysis cast serious doubts upon the usefulness of then current economic and statistical theories of rational behaviour as bases for explaining the characteristics of human and other organismic rationality. (And from this dissatisfaction sprang forth LT & GPS?)

As a device to unify the various ways in which psychology views man (perception, learning, motivation, emotion, ...), [Toda,1962] studied the design of a solitary robot on a distant planet. The robot's job is to collect uranium randomly distributed on the surface, and the robot obtains energy from eating a certain fungus that grows at random locations on the surface. The bodily design, perceiving program, and choice program were all considered. The choice program has to choose what direction to travel in at each moment. Extending Simon's approach, a decision-theoretic analysis based on maximizing the amount of uranium collected is given and a choice strategy specified. The effect of obstacles on the choice strategy and how various approximations could reduce the computational effort required are also considered. The robot uses no stored representation of the environment.

[Becker,1973] analyzed a simple robot world in what I call "Baconian" terms. The robot observes events as they happen and then tries to induce, in true Baconian style, representations to predict such events in the future. (My system may be said to function in "Popperian" style.) He proposed a representation and a system of processes by which the robot could store and manipulate the experience it gained through interacting with its environment. The world consists of a smooth shelf on which coloured blocks may be placed and manipulated, a simple movable square eye with 9 square retinal fields, and a hand that appears in the eye as a 1X1 red square. The world obeys the laws of physics. A history is kept of motor commands and of query commands with their sensory answers. From this historical record the robot tries to induce a semantic-net-like representation, which it uses to predict the outcome of future actions.

Becker's approach is based on one simple idea: that if B followed A in the past, an organism should remember that fact, so that the organism can expect B to follow A in the future. Becker's approach is very interesting not because it succeeds, but because it clearly illustrates the difficulties associated with a Baconian approach. These appear right at the start of his analysis. First, given the continual stream of kernels (motor commands and sensory input), there is the problem of deciding which kernels are significant. If this decision is attempted at too low a level of representation, as, I claim, Becker does, it ends up being based on quite arbitrary criteria. (Just as a hypothetical Baconian scientist could make a million observations in a situation, but since that is not feasible, must decide somehow which ones are of interest.) Second, supposing a significant kernel has been chosen, there is the problem of deciding how many nearby kernels may have a causal relation to the chosen one and should therefore be stored as part of this 'event'. What if two causally related kernels are separated by large periods of time, as might occur in object occlusion problems? Becker has no satisfactory solution to this problem, which might be termed a 'windowing' problem. Third, several numerical scales are introduced and manipulated on an ad hoc basis to provide measures of criticality, confidence, cost etc., which are used to enable rules (derived from events) to be generalized, or differentiated into distinct subrules. These apparently arbitrary numerical scales are a very unsatisfactory feature. In sum, a very interesting proposal, but mainly for its faults and not for its successes.

II.4.3 Simulations based on animal behaviour

[Ludlow,1976] describes a model animal which was designed to simulate aphid behaviour. This model is only concerned with alternations between several different types of behaviour, e.g. walking, feeding, probing, flying, wingspreading. The model is based on the concepts of centres, drives, and reciprocal inhibition between centres. For each activity there is a separate centre. The centres inhibit each other. When a particular centre is active the inhibition from it is sufficiently strong to suppress an equally stimulated rival; but the centre fatigues. In such a system only one centre is active at a time (although it is possible for several centres to be active concurrently, such a configuration is unstable). The system exhibits hysteresis: once an activity is started it will persist for a period even when the drive level necessary to elicit the activity has been reduced by the performance. This would seem to be a necessary feature of any organism which can execute many different behaviours, to prevent thrashing. This approach might usefully be incorporated in an AI system controlling several different behaviours.

[Friedman,1967] analyzes and extends the Lorenz - Tinbergen theory of instinctive animal behaviour by adding "Selection of Releaser Mechanisms" to the executive control hierarchy. The computer simulation of a small animal (ADROIT) that moves in a plane with a small number of circular obstacles was programmed, with a control program designed along the lines of the afore-mentioned theory. ADROIT avoids obstacles when en route to a goal by reading the angles and ranges to the edges of cylinders. The structure of the "Behavioural Unit" to carry out a "go to" command was exhibited. No representation of the world was involved.

[Arbib & Lieblich,1977] are concerned to bridge the gap between human memory studies and the psychological literature on animal learning and conditioning. The major reason for the huge research effort on animal behaviour has been the Thorndike - Pavlov - Bitterman theory that the underlying processes of learning are the same in all animals, including man [Bitterman,1975]; consequently this is an important direction of research. They propose a theory of how an organism couples its memory structure to its specific action routines so that it may operate in its spatial environment in an intelligent manner.

They adopt a world model in the form of a graph with nodes containing drive-related information and edges containing sensorimotor features. The theory specifies the general drive dynamics, the way in which the world model is updated, and the way in which the rat decides where to move next in the world. Their theory explains some experimental results that relate rat learning and spatial behaviour.

II.4.4 Robot simulations based on decision theory

[Jacobs & Kiefer,1973] consider the decision-making component of a robot that operates in a poorly known environment, where each action may have many possible outcomes. An approach based on maximizing the expected utility resulting from each decision is developed. The decision to execute a particular action is viewed as a move in a game against the environment; the outcome of an action is the environment's move in the game. The estimated utility of a decision is evaluated by backing up from the terminal stages of a plan, using the fact that the utility assigned to a set of uncertain outcomes is the expected value of their utilities. The decision that maximizes the expected utility is chosen. This approach is used to control a simulated insect-like robot which seeks food, collects material for a nest, and may be stung by an enemy. The robot's task is to build a nest. The task is not explicitly represented to the robot but is specified through the utility functions for eating, adding material to the nest, finding material, and being stung. Likewise, eating is not represented as a goal except through its utility function and in fact with the utility used eating will never occur if the time since the previous meal ever exceeds a certain bound - so the poor robot will starve. However, the (negative) goal of starvation is not represented either. No stored representation of the environment is used.

[Coles et al.,1975] and [Feldman & Sproull,1977] apply decision theory to symbolic problem solving. Their respective examples are essentially equivalent and can be stated as a modified version of the monkey and bananas problem. In this version several boxes are available to be pushed under the bananas but not all are suitable, and the monkey is provided with a device for sensing "suitability" from a distance. Unfortunately the device is not reliable and may give false positive and false negative answers. All the actions of the monkey - walking, pushing, climbing, sensing suitability - have energy costs. The techniques of decision theory are used to find the best solution strategy, using a utility function defined in terms of energy cost. The utility function is used to reveal tradeoffs among various strategies for achieving various goals, taking into account such factors as reliability, the complexity of steps in the strategy, and the value of the goal. It is also used to formulate solutions to the problems of how to acquire a world model, how much planning effort is worthwhile, and whether verification tests should be performed. Feldman & Sproull discuss many other possible applications of decision theory in robot problem solvers.

Feldman and Sproull's paper supports their claim that "a combination of decision-theoretic and symbolic artificial intelligence paradigms offers advantages not available to either individually". However, although I can't yet pinpoint it exactly, I confess to a queasy feeling when applying probability theory to symbolic reasoning. The basic definition of the theory is the probability of an event, defined as "the limiting value of the relative frequency of occurrence of the event in a long sequence of observations of randomly selected situations in which the event may occur" [Parzen,1960]. Philosophically this is very unsatisfactory. Bayes' theorem, an important rule for computing conditional probabilities, is even more unsatisfactory. The task of clearly delineating these difficulties and proposing a new definition of probability is beyond the scope of this thesis. All that can be said is that there are many inklings around, and in chapter IV I will give some indication of the direction required.

These simulations serve to confirm Simon's conclusion that traditional decision theory is not appropriate to the analysis of behavioural systems.

II.4.5 Cognitive maps

A traditional field of psychology is concerned with cognitive maps ([Trowbridge,1913], [Tolman,1948], [Moore &

Golledge,1976], [Kuipers,1978]). A person's cognitive map is the knowledge a person has about the spatial structure of large-scale space. Thus the topic of cognitive maps is relevant to my work.

The functions of a cognitive map are to assimilate new information about the environment, to represent one's current position, and to answer route-finding and relative-position problems. It is built up from observations made as one travels through the environment. [Kuipers,1978] presents a computational model (the TOUR model) of the cognitive map that uses multiple (5) representations for the cognitive map, and builds up knowledge by observations and by interactions between the separate representations. Whereas TOUR gains new knowledge by discrete observations at a small number of fixed places, Utak gains new knowledge by receiving a new retinal impression at a new position and resolving the differences between the actual impression and the predicted retinal impression by modifying the hypothesized shape of the environment. Utak's skeleton of the environmental empty space is very similar to Kuiper's cognitive map when regarded as a network of routes. Whereas TOUR is only concerned with city-street networks and not at all with shape, PPA explicitly represents the two dimensional shape of the environmental space. In sum, Kuiper's work is somewhat complementary to mine.

II.4.6 Spatial planning systems

[Eastman,1973] reviews current programs and describes a new program, GSP, for solving two dimensional spatial arrangement tasks. Given a space S (e.g. a large rectangular room), several smaller rectangular design units (DUs) (e.g. the parts of a computer), and several S-relations between the DUs (e.g. an edge-adjacency requirement or a sight-line requirement for the operator's desk), the problem is to find an arrangement of the DUs in the space S which satisfies all the S-relations. The overall design of GSP is as a backtracking depth-first search. Various heuristics are described which improve the search, derived from the S-relations. An important part of GSP is the location proposer which, when given an arrangement of some of the DUs in S, proposes locations for a new DU which are consistent with the arrangement already made. Only arrangements in which the sides of the DUs are aligned with the sides of S are considered.

[Pfefferkorn,1975] described another spatial planner, DPS, which relaxed the restriction that all shapes be rectangular by allowing non-convex polygonal shapes, and which allowed a new type of spatial constraint on an arrangement: a path constraint, which says that all the empty space in the arrangement must be connected. DPS uses a representation of space occupancy of an arrangement in which convex polygons are the primitives. Some are marked empty and some are marked occupied. These convex polygons are called space blocks. Each space block is in turn represented as a set of sides, and each side as a set of points. When a new shape is added to an arrangement every space block intersected by a side of the new shape is broken into two separate space blocks and the occupancy marked accordingly. The location proposer essentially proposes all the corners of empty space blocks. As in GSP the constraints are used to guide the search.

Both systems explore a search tree of space layouts where the branching factor at each node is controlled by the location proposer and by other heuristics which decide in what order to try fitting new shapes. The primitive shape concept used is a convex polygon represented as a list of boundary points. Their main fault from my point of view is that these systems are concerned only with object placement, not with path-finding or object moving.

II.4.7 Systems for simulating the motion of rigid objects

[Baker,1973], dissatified with conventional methods for spatial simulation, desired one in which the spatial relationships of points were explicit. To this end he presented the design of an iterative array of logic circuits which could simulate the continuous rigid translation or rotation of arbitrary shapes, and implemented a simulation of this array. The system consists of a rectangular array of logical circuits, each representing a unit square. Each circuit has a local coordinate system to keep track of a single point as it crosses the square. Its path may be a straight line or a circular arc. On reaching the side of a square, control and modified local coordinates of the point are passed to the neighbouring square. An object is represented as a collection of points (where for technical reasons the minimum distance between points must be greater than the square root of 2(root2)), and its motion simulated by following the paths of all the constituent points. The system was not developed to handle collisions.

[Funt,1976] argues that a computer program can derive benefits from the use of analogues in the same way that people do. To this end, he implemented a system WHISPER. The purpose of this system was to solve two dimensional blocks world stability problems by the use of a so-called analogue. I will not comment on his arguments concerning the use of analogues; from my point of view WHISPER was intended to be a performance system for simulating rigid object motion under the influence of gravity.

The input to WHISPER is a two dimensional array of colored squares on which the side view of a configuration of distinctly coloured, arbitrarily shaped, blocks has been drawn. Typically the corresponding real world situation contains many instabilities and under the influence of gravity would immediately collapse in a flurry of block motions and interactions: rotation, collision, sliding, and free fall.

WHISPER simulates this collapse on the input array and produces as output the same array but with the block positions updated to display their predicted final resting places. The simulation makes extensive use of a retina which resembles the human retina in some respects. Under control of the main program which knows about gravity, the block motions and interactions are computed through the use of several operations on the retina, including finding centre of area, finding contacts between blocks, visualization of rotation, and finding symmetry. The retina consists of a circular array of non-overlapping circular retinal fields, or bubbles. The bubble size increases with distance from the retinal centre. Each bubble has an associated processor, so that the whole retina is conceived of as a fixed number of processors operating in parallel and communicating only with their immediate neighbours. Funt's retina is similar in this respect to Baker's iterative array of automata. Only the color of an object becomes known to WHISPER's main program, while an object's shape and other properties reside in the diagram.

WHISPER's movement primitives are simple and after a few simulated motions the depiction of an object on the array disintegrates into a multitude of small isolated pieces. A precise demonstration of this fact appears in appendix A.3. The conclusion is that the simulation of rigid motion provided by WHISPER is not suitable for my purpose.

[Howden,1969] considers the sofa-moving task; that is, produce a plan for moving a two dimensional shape from one place to another when constrained to remain within the walls of a surrounding, and in general non-convex, two dimensional shape. The edges of the walls and of the sofa are represented as lists of points using chain-encoding [Freeman,1974]; consequently it is easy to simulate rigid object motion. It is not, however, so easy to detect the intersection of the sofa and the walls. I am not convinced that the algorithm as described in this paper will work, though it can be extended to do so. Presumably the author used such an extension, since he reported on a running program. In a pre-execution step, the points of the wall are sorted into an array of buckets, which, in the extended algorithm, must be probed twice for every point on the perimeter of the sofa. So the wall array is usually referenced

2 * (length of sofa perimeter)

times for every intersection test performed. A sofa-moving plan is produced as follows. At any (integral) point within the walls there is a small number of possible actions of translation or rotation which may be applied to the sofa; the permissible actions are those for which the intersection test fails. The plan is produced by executing an undirected, looking backwards, heuristic search through the state space entailed by the set of permissible actions at each point. That this scheme performed at all is somewhat surprising - apparently it did, on some poorly specified examples. It would perform particularly badly in the simplest case - a small sofa within a large empty containing space.

II.4.8 Imagery

Mental imagery is relevant because the SHAPE subsystem of PPA can be viewed as a model of mental imagery even though that was not the goal of SHAPE's design. Mental imagery has been discussed in the psychological literature by [Bartlett,1932], [Hebb,1968], [Piaget,1954], [Shepard,1978] and many others. This is how Shepard, in the conclusion of his recent review, presents the current status of mental imagery in psychology:

> I submit that there are both logical and analogical processes of thought, and that processes of the latter type, though often neglected in psychological research, may be comparable in importance to the former. By an analogical or analog process I mean just this: a process in which the internal states have a natural one-to-one correspondence to appropriate intermediate states in the external world. Thus, to imagine an object such as a complex molecule rotated into a different orientation is to perform an analog process in that half way through the process, the internal state corresponds to the external object in an orientation half way between the initial and final orientations. And this correspondence has the very real meaning that, at this half-way point, the person carrying out the process will be especially fast in discriminatively responding to the external presentation of the corresponding external structure in exactly that spatial orientation. The intermediate states of a logical computation do not in general have this property. Thus, a digital computer may calculate the coordinates of a rotated structure by performing a matrix multiplication. But the intermediate states of this row-into-column calculation will at no point correspond to - or place the machine in readiness for - an intermediate orientation of the external object.

To summarize: thanks to the searching reaction time experiments of Shepard and his colleagues, the notion of analogical thought process now has a firm piece of evidence to rest on.

I have already mentioned the apparent importance of mental imagery in scientific and mathematical discovery; in addition one could justifiably interpret Hilbert's "concrete objects" (p. 34) as visual imagery.

There is currently a debate over whether the notion of mental imagery can be used as a scientifically respectable explanatory construct. The main protagonists have been [Pylyshyn,1973,1976] and [Kosslyn and Pomerantz,1977]. This cannot be discussed here. The latest word in this debate, and a review, is provided by [Anderson,1978].

II.4.9 Behavioural theories

In attempting to design a robot controller one is, essentially, developing a behavioural theory. Thus it is worth taking a brief look at work in this area.

[Hebb,1949] developed a cell-assembly theory of behaviour, which has been extended by [Good,1965], [Bindra,1976], and others. It is intended to be a physiological theory of thought. He approaches his theory from two directions: the psychological facts of attention and orientation, and the then current facts of neurology. Hebb describes a cell-assembly as a "tridimensional lattice-like assembly of cells, that I have supposed to be the basis of perceptual integration." Again, he writes, assemblies are "diffuse, anatomically irregular structures that function briefly as closed systems, and do so only by virtue of the time relations in the firing of constituent cells... An individual cell or transmission unit may enter into more than one assembly, at different times... At any one moment, the action of an assembly may be considered to be on an all-or-none basis" [p.196-7].

[Bindra,1976] extends and diversifies the cell-assembly theory and introduces a new concept, the pexgo. A pexgo underlies the "currently excited, distinctive neural organization that underlies the identifying response made in relation to a stimulus entity, as well as the awareness (subjective experience described as percept or image) of that stimulus entity." Though suggestive, the cell-assembly/pexgo theory is at an insufficiently precise stage of development to be of any direct benefit.

The fault of the Hebb-Bindra theory is perhaps this: it tries to explain human thought directly in terms of (the poorly known) neuronal structure, which might be likened to trying to explain a big computer program such as an operating system or Winograd's SHRDLU directly in terms of machine code, by-passing all mention of PLANNER, PROGRAMMER, LISP, stacks, assemblers and all the other wonderful descriptive vocabulary of computer science. In other words, the difference in descriptive level, the gap between neuron and thought, is too great to be bridged by one single reductionist theory. Artificial Intelligence, using the language of computer science, is in an ideal position to build the requisite intermediate theories.

[Miller, Galanter and Pribram,1960] also sketched out some ideas on behaviour; their most specific suggestion was the importance of "TOTE" units (test, operate, test, exit) in executing plans. The TOTE concept is related to the notion of a FAP (fixed action pattern), which is used by ethologists to describe animal behaviour and to trace the evolution of behaviour.

In pondering why computers have had so little success in carrying out human information processing tasks, [Miller,1974] concluded, first, that the reason is because there is no satisfactory theory of cognitive organization, and second, that the best hope for progress is to develop a theory to handle the structure of the physical world. My work is a small step in the direction of Miller's second conclusion.

\+ \+ \+

So what can I conclude from our survey of the literature? I will start with the negative conclusions and proceed in a positive direction. First of all, though many of these studies look superficially similar to our project, few have any positive content from our viewpoint. The lessons to be learnt are mainly "don't"s. Here they are.

- Hebb - Bindra - don't try to do too much with one theory
- Jacobs & Kiefer - don't try to apply decision theory directly to behaviour
- Coles, Feldman & Sproull - decision theory and Artificial Intelligence don't really mesh together
- Ludlow, Friedman - irrelevant because they model behaviour without a world model
- Becker & Merriam - they get bogged down in simulation details; no functional robot-controller designed or implemented.
- "Baconian" Becker - don't use the Baconian approach to representing experience.
- Simon, Toda - interesting high level analyses of rational behaviour, but irrelevant at our level of synthesis.
- Imagery - this is an acceptable notion; any model of it is of interest. My model of it arises as a side effect of a system designed for spatial reasoning.

Of the three studies on the simulation of rigid motion, Baker's is promising but not carried far enough, Funt's simulation is not satisfactory after the first few moves, while Howden's is computationally rather expensive for use as an experimental tool. When it comes to spatial planning, Eastman and Pfefferkorn get bogged down in heuristic search because their underlying representation of space is inadequate, and Howden's approach results in a combinatorial explosion. More positively, Nilsson & Raphael's is interesting, but only as a precursor of my own work. This leaves only Kuipers, who models common-sense knowledge of large-scale space, and Arbib & Lieblich, who model a rat's cognitive map.

Kuipers showed how fragmentary pieces of information about one's spatial environment can be integrated in the course of experience to form a graph-like cognitive map. Arbib & Lieblich used a graph for their world model and showed how it could be modified as a result of innate drives and of external rewards. The lesson to be learnt here is that a world model in the form of a graph is a promising idea. This is not incorporated in the design of PPA but obviously should be taken up as soon as possible.

+ + + + +

I will now summarize this chapter on background issues. I first delineated the nature of this modern science, Artificial Intelligence; then I described the importance of, and interaction between, representation and search in any theory of intelligence; then I presented my own approach to the subject. The next section sketched the traditional Artificial Intelligence approach to planning and problem-solving, and found it to be wanting for my purposes; while in the last section I reviewed the literature on similar projects but found there to be a notable lack of positive content, only cautionary tales. All told, the reader should now have a good feeling for the background to my work; let me now advance to the first embattlement.

CHAPTER III

THE SIMULATED ORGANISM-ENVIRONMENT SYSTEM

This system is the basic experimental tool for my research. It provides sensory input for, and accepts motor output from, a simulated organism that I call Utak. Only the functional input-output characteristics of this system are directly relevant to the rest of my thesis. For this purpose you need only read the rest of this introductory section and section III.2.2, and peruse the examples in sections III.1.3 and III.2.3. The aim of this chapter is to describe the simulated organism-environment system and to describe the tasks that such an organism, if endowed with a competent organism controlling program, might reasonably be expected to solve.

The system is called TABLETOP. It simulates the physical motion on a smooth tabletop of objects which have the form of polygonal planar shapes. The tabletop is bounded by a verge so that an object can never fall off. An object moves only when Utak is both holding this object and executing a pushto or turn command. The physics involved is essentially trivial:

(a) The shape of an object is invariant under translation and rotation.

(b) If a motor command to go a certain distance in a certain direction would result in Utak colliding with an object or the verge, then he halts a short distance before the first intersection of his path with such an obstacle.

(c) Similarly, if Utak is grasping an object and is executing a push command that would result in the object or Utak colliding with some obstacle, then Utak and the object come to an immediate halt a small distance before the point at which the this collision would have occurred.

(d) When Utak is grasping an object he and the object are, temporarily, considered as one new object.

(e) Utak can go between two neighbouring objects only if the width of the gap between them is greater than a certain minimum value.

To put it in a nutshell, TABLETOP simulates the permanence and impermeability of the shape of physical objects.

In building the TABLETOP system I aimed to produce an experimental tool that was inexpensive to use. I was not concerned to find exact solutions to collision problems. Thus, the approximate solutions to collision problems that the TABLETOP system computes are quite sufficient for my purposes. In section III.3.1 I sketch one way that TABLETOP could be extended to compute exact solutions.

Previous simulations of the physics of planar polygonal shapes (reviewed in II.4.7 above) have either been incomplete, incorrect, or computationally expensive to use, whereas my TABLETOP system is complete, correct to within certain limitations which I specify later, and efficient. By complete I mean that both motion and collisions are handled. TABLETOP is cheap to use, has been used extensively, and has proved to be a viable experimental tool.

The design of TABLETOP is based on the use of two representations for objects, the Cartesian and the digital. The <u>Cartesian representation</u> of an object specifies the shape of the object by a list of points where each point is specified by two positive real numbers. The points are the points of inflection on the boundary of the shape. An <u>edge</u> in the Cartesian representation of an object is a pair of consecutive points in the list. Utak himself has a Cartesian representation, or <u>position</u>, consisting of a single pair of positive reals. In addition, Utak has an absolute orientation. Note that I am here referring to the simulation of Utak, not the robot-controller for Utak.

The TABLE is a two dimensional array where each entry corresponds to a square in a two dimensional grid of squares covering the surface of the simulated tabletop. Each object has an associated <u>colour</u>, one of the letters A,B, ... Z. Two objects may have the same colour, and the verge always has the colour 'B'.

Now imagine the Cartesian representation of an object with colour c superimposed on the TABLE grid. The <u>digital representation</u> of the object is defined to be the set of squares of TABLE that lie within, or are intersected by the edges of, the Cartesian representation of the object. All squares in the digital representation of an object are assigned the object's colour c. The digital representation of an object is also called the projection of the object onto the TABLE. Utak has a digital representation, or projection, consisting of the square of TABLE that contains his position. The colour assigned to this square is the BUGMARK, an asterisk on the CRT display of TABLE. His Cartesian position lies outside the Cartesian representations of all the objects on the tabletop. Normally his projection, also, is outside all the digital representations of all the objects, but it can happen that it lies within the projection of an object that he is currently grasping or has recently letgo.

Utak, all the objects, and the verge are projected onto the TABLE array when TABLETOP is in operation. The TABLE array can be displayed on a screen for a human user to watch. Remember that Utak does not "see" this display; his visual input is described in subsection III.2.2.

This chapter is organized as follows. Section III.1 describes the TABLETOP simulation system. This includes the method used, the problems encountered, and the specification of the requisite algorithms. Section III.2 discusses the design and capabilities of Utak, and includes examples of his sensory-motor experience. In the final section (III.3) an extension and generalizations of TABLETOP are considered. It is shown that an important part of the TABLETOP simulation is easily adapted for parallel computation, and that the TABLETOP method generalizes to three (or more) dimensions. Also, it is shown how to extend TABLETOP to obtain exact answers to collision problems.

III.1 The simulated environment, TABLETOP

This is an independent system that simulates the effect of motor commands issued by Utak. It is instructive for a user to sit down with TABLETOP and attempt a task such as manipulating an 'L' shaped object through a narrow doorway.

The L-shaped object problem is the archetypal task for Utak. Indeed, in Kuhnian terminology, this is the paradigm problem for this approach to understanding spatial intelligence. When, or if, progress in the construction of the organism-controlling program for Utak has advanced to the point where Utak is able to solve this problem autonomously then I believe that non-trivial advances will almost certainly have been made towards understanding the nature of some computations that are of fundamental importance for successful organisms. At that point it will be of great interest to interpret the known facts about biological brains in terms of these computations.

III.1.1 An overview of the simulation method

A *slide* is the simplest action of Utak. This is the movement of Utak along a line segment. It is simulated by sequentially checking each square of the TABLE grid that is intersected by Utak's position as he moves along the line segment. If a non-empty (coloured) square of TABLE is encountered before the end of the line segment, then first the point of intersection with the obstructing square is found, second the halting position of Utak is obtained by backing off slightly from this point. This is done by taking a point a small distance ϵ back along the line segment from the point of intersection. If the Cartesian representations of two neighbouring objects are so close that no empty square of TABLE lies between their digital representations then Utak is unable to slide between the two objects.

When Utak is not grasping an object he can only execute a slide action or a grasp action. Utak can grasp an object if he is not already grasping some object and if his digital representation is adjacent to a square in the object's digital representation. Two squares are adjacent if they are horizontally, vertically, or diagonally adjacent. Thus there are eight TABLE squares adjacent to Utak's digital representation.

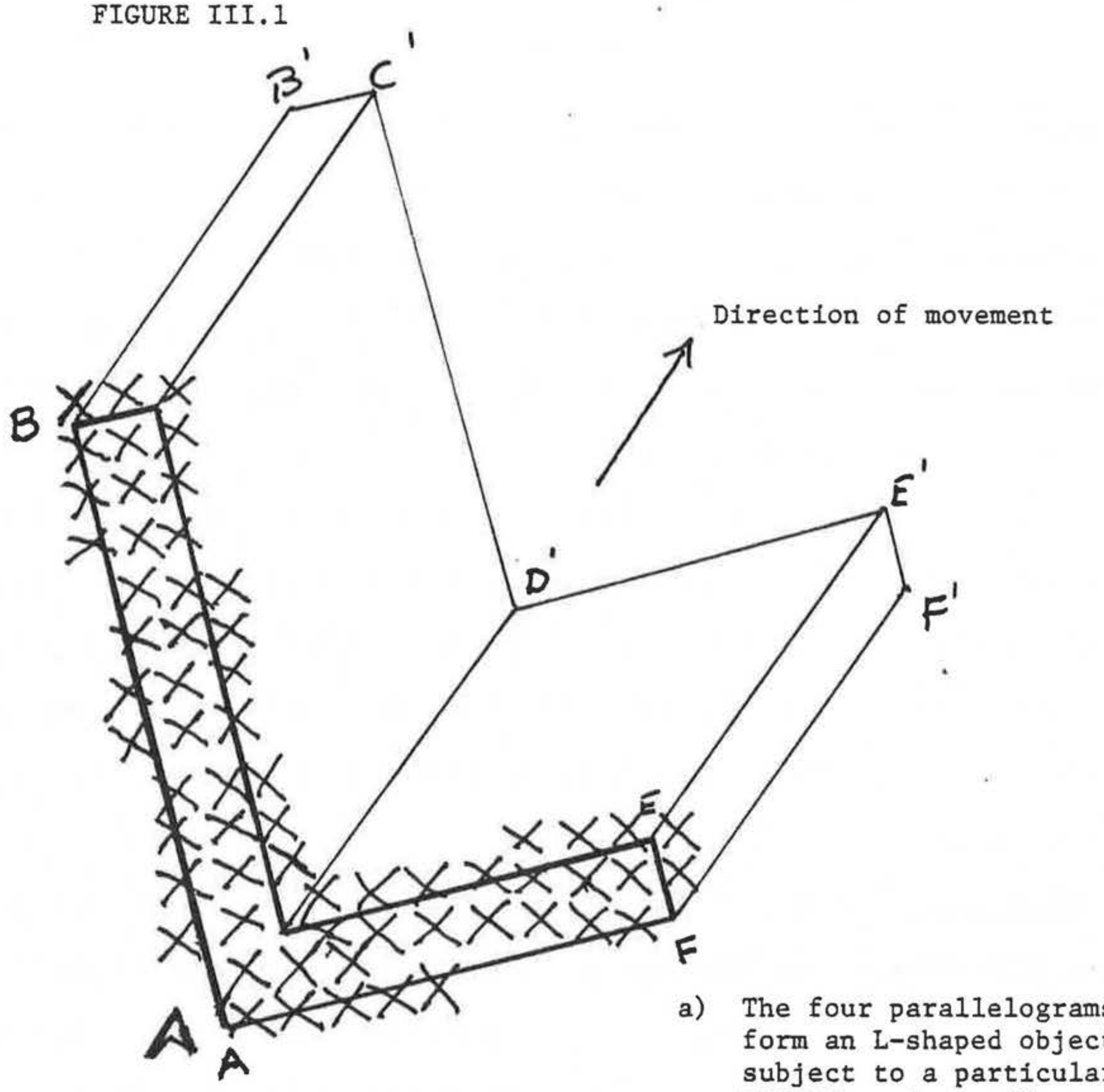
When Utak is grasping an object he can only execute pushto, turn, or letgo actions. In this state the relative position of Utak's Cartesian representation and the object's Cartesian representation remains invariant under translation -- caused by pushto actions -- and rotations -- caused by turn actions. However, whereas Utak's digital representation is always a single square, an object's digital representation may appear to change rather drastically if the size of the TABLE squares is of the same order of magnitude as the size of the object. The simplest example of this effect is given by an object whose Cartesian representation is a square of exactly the same size as the TABLE squares. If this object's Cartesian representation is exactly aligned with a TABLE square then its digital representation is just that TABLE square, but if the object is moved diagonally a small distance then its digital representation becomes a larger square consisting of four TABLE squares.

Suppose that the user of TABLETOP requests a pushto action whose intent is to move the grasped object in a straight line through distance d in direction θ. The angle θ is measured clockwise from some fixed direction. The following method is used to compute the distance d' actually traversed before a collision, if any, occurs. d is the intended distance, d' is the achieved distance. Basically the method is to scan the area of TABLE that would be swept out by the object in the course of the translation. The distance to the nearest obstacle found, if any, determines the achieved distance.

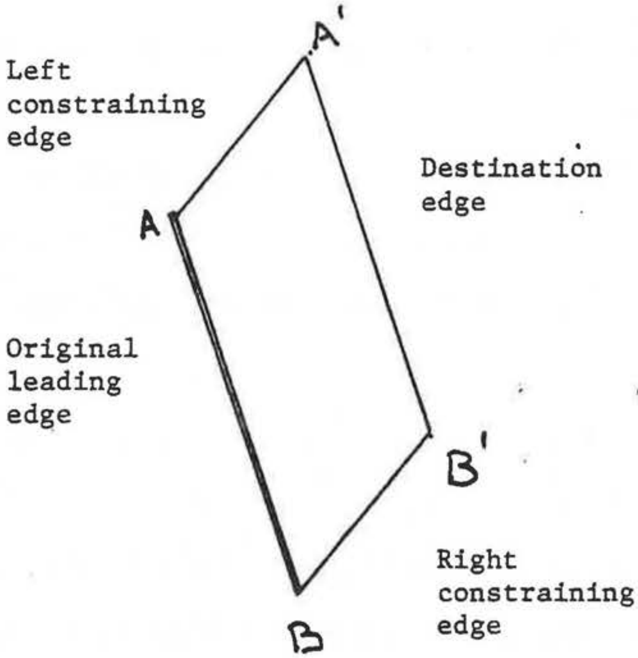
First the current digital representations of Utak and of the object are erased. Then the achieved distance for Utak is computed, just as for the slide action. The achieved distance for the grasped object is computed as follows. First the leading edges relative to the direction θ are determined. These are the edges of the Cartesian representation whose outward normal has a direction in the range $(\theta-90^\circ,\theta+90^\circ)$. As the object is moved, each leading edge sweeps out an area in the shape of a parallelogram. Each such parallelogram is to be imagined as superimposed on the TABLE grid. The parallelograms for an L-shaped object subjected to a particular pushto action are shown in figure III.1. For each leading edge E a scanning process is started that scans those squares of TABLE that lie within or intersect the parallelogram PE generated by E. For any scanned square that is non-empty (coloured), the minimum distance from the edge E to the nearest point of the subpart of the square lying within PE is computed. Then distance DE is defined to be d, if no non-empty squares are found in the scan, or else to be the minimum over the minimum distances for each scanned square. The minimum of the DE's for each leading edge E, less a small quantity ϵ, is returned as the achieved distance for the object. Finally the overall achieved distance d' is the minimum of the achievable distances for the object and for Utak. Then both Utak and the object are re-projected onto the TABLE grid at the computed final position. By construction, these new projections never overlap the projection of an obstacle.

Now suppose that the TABLETOP user requests a turn action, whose intent is to rotate the object grasped by Utak by ϕ radians about Utak's position U. Two methods will be described for computing the angle ϕ' actually rotated before a collision, if any, occurs. ϕ is the <u>intended</u> rotation, ϕ' is the <u>achieved</u> rotation. Both methods scan for obstacles in the area that would be swept out by the object in the course of rotation. The

FIGURE III.1



a) The four parallelograms form an L-shaped object subject to a particular PUSHTO action. Also shown are the Cartesian and digital representations of the L-shape.



b) Features of a parallelogram generated by an edge during a PUSHTO action.

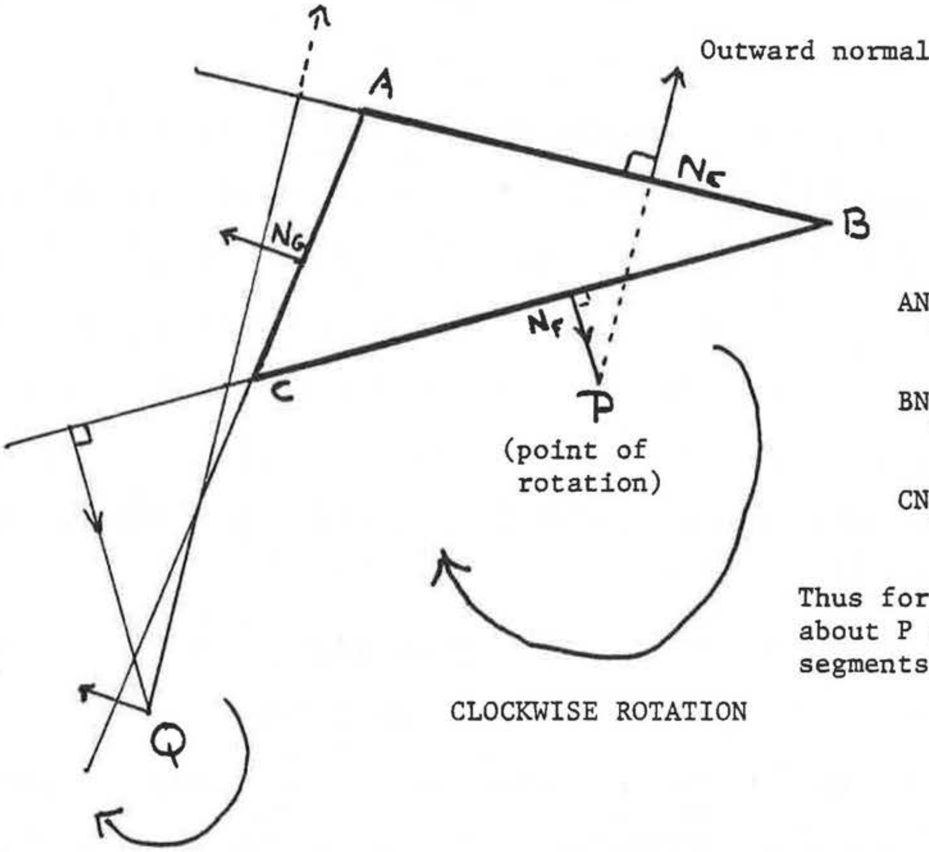
angle to the nearest obstacle, if any, determines the achieved rotation. The first method finds all the obstacles whereas the second may miss a small one. Although the second method is the one currently implemented, the first method, since it parallels the method used for translation and may be of independent interest, is described first.

First the projections of Utak and the grasped object are erased. Since Utak's position does not change in a rotation, his motion does not directly contribute to the collision computation. Then the leading segments of the edges of the object, relative to the centre of rotation U, must be determined.

Definition. For a clockwise rotation of an object about U, the leading segment of an edge E is found as follows. Consider the line L collinear with E. Compute on the line L the nearest point N to the centre of rotation U and draw an outward normal to E at the point N. Now take the semi-infinite half-line of L that lies to the left of the outward normal at N, and form the intersection of it with E. The resulting segment of E is the leading segment of E. Note that the leading segment of an edge may consist of all or part of the edge, or be null. The leading segments for a specific triangle and points of rotation are shown in figure III.2.

Lemma. For a clockwise rotation of an object about U, the leading segment of an edge E of the object has the property: for any point x on the interior of the leading segment, there is a

FIGURE III.2



AN_E = leading segment of edge AB.

BN_F = leading segment of edge BC.

CN_G = leading segment of edge CA.

Thus for rotation clockwise about P there are 3 leading segments.

For clockwise rotation about Q there is only 1 leading segment: BC.

This figure shows the leading segments of the triangle ABC for two points of rotation. If a clockwise rotation about P is intended, then the leading segments are AN_E, BN_F, CN_G. If an anti-clockwise rotation about P is intended, then the leading segments are N_EB, N_FC, and N_GA. If a clockwise rotation about Q is intended, there is only one leading segment: BC. If an anti-clockwise rotation about Q is intended, there are two leading segments: CA and AB.

rotation about U such that if x' is the new position of x, the arc xx' lies in the exterior of the object.

Proof. Pick a disc centre x, small enough that it intersects no other edge of the object, and so that it does not contain an end point of the leading segment. Consider a rotation so small that the new position x' of x lies within the disc centred on x. Because x lies to the left of the outward normal to E the direction of motion of x is perpendicular to Ux and points into the exterior of the shape. Thus, the arc xx' lies in the exterior of the object. QED.

In other words the leading segment of an edge E always sweeps out a new area of the TABLE in the course of a rotation. Depending on the angle of rotation and the exact overall Cartesian shape of the object, there will in general be considerable overlap between the areas swept out by each leading segment. I have ignored the problem of eliminating multiple scanning of areas of TABLE in a turn action.

As the object rotates each leading segment sweeps out a four-sided area of space. The sides distal and proximal to the point of rotation are circular arcs, the other two sides are straight lines. Hence I call this shape a doughnut slice. Note that if A, B are the original end-positions of the leading segment and A', B' are the final end-positions of the leading segment, then triangles ABU and A'B'U differ only by a rotation about U. Each doughnut slice is to be imagined as superimposed upon the TABLE grid. The doughnut slices for the rotation of an

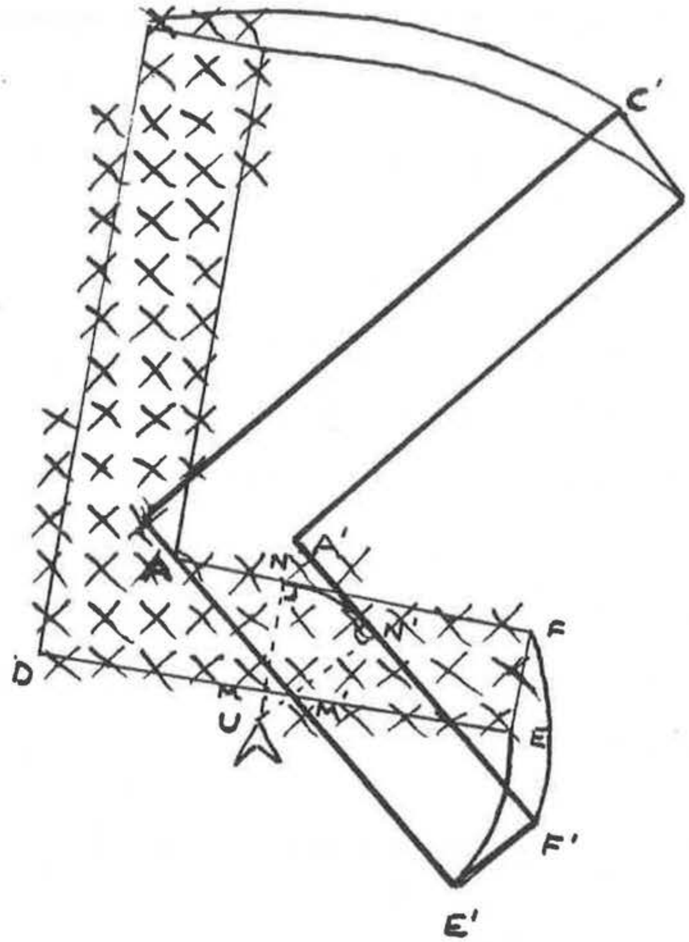
L-shaped object subjected to a particular turn action are shown in figure III.3. For each leading segment S the TABLE squares that lie within or intersect the corresponding doughnut slice are scanned. For any non-empty scanned square the minimum angle of rotation sufficient to cause a collision between the leading segment S and the square is computed. (This is not a trivial computation.) For each leading segment S, the minimum is taken over the angles computed for each obstructing square, and then the minimum of all these minimums, taken over all leading segments, gives the achieved rotation ϕ'. Finally, both Utak and the rotated object are re-projected onto the TABLE grid. That, in outline, is the simulation method used in TABLETOP.

The basic problems faced in an implementation of this simulation method are as follows.

- Tracing a line
  - -- the intended path in a slide act
  - -- the sides of a parallelogram in a pushto action
  - -- the straight and curved sides of a doughnut slice in a turn action

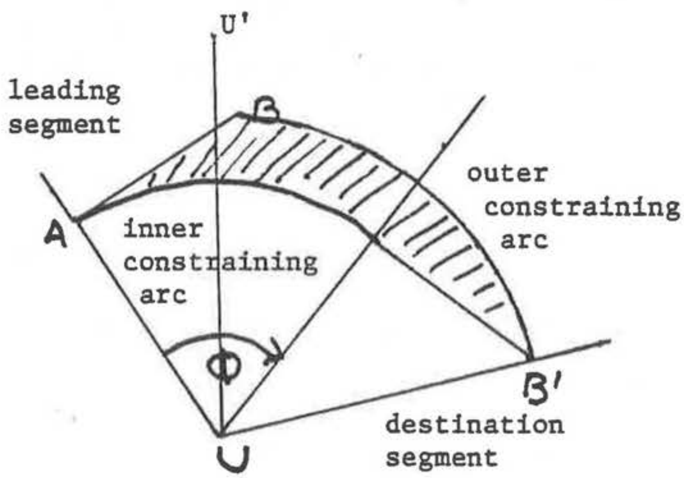
- Scanning a shape
  - -- the Cartesian representation of an object, for projecting or erasing the object's digital representation
  - -- the parallelogram swept out by a leading edge in a pushto action
  - -- the doughnut slice swept out by a leading

FIGURE III.3



Rotation about U.

An L-shaped object showing its Cartesian and digital representations. For the centre of rotation, U, there are 5 leading segments (CB, BA, AN, FE, EM) and 5 doughnut slices swept out during the action (BCC'B', ABB'A', NAA'N', EFF'E', MEE'M'). N is the point on AF closest to U, M is the point on ED closest to U.



Features of the doughnut slice generated by rotating the leading segment AB about the centre of rotation U by angle Ø.

segment in a turn action

▪Computing minimum distance from a leading edge to (a subpart of) an obstructing square

▪Computing minimum angle from a leading segment to (a subpart of) an obstructing square.

III.1.2 The algorithms used in the simulation

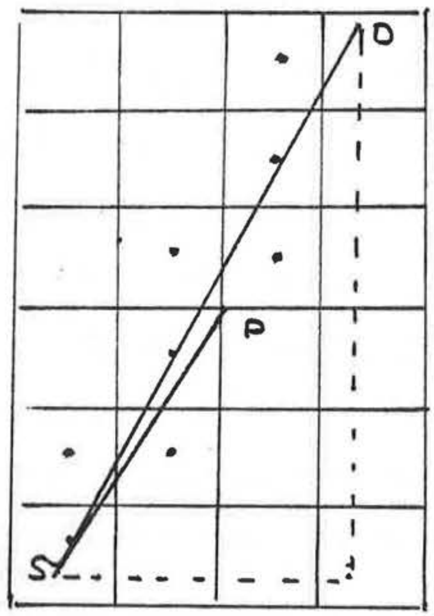
In this subsection I describe the algorithms used to solve the problems specified in the previous subsection. There are two line-tracing algorithms, one for straight lines and one for circular arcs. I first describe how to trace a straight line. As a prerequisite for this one has to know the squares of the TABLE grid containing the initial and terminal points of the straight line in order to initialize and terminate the tracing process correctly. This seemingly innocuous requirement is a little tricky to program because of the special cases that can occur such as alignment of the line with the axes and coincidence of the line with the grid lines. The code for tracing has to handle four cases, one for each quadrant of the direction of the line; I will only describe the northeast quadrant case. Let the current square be the square currently under consideration in the line-tracing procedure. The next current square can either be one up, one right, or diagonally up and right from the current square. This choice is made by computing whether the northeast corner of the current square is left of, right of, or on the line. A comparison of the slopes of SP and SD, involving two multiplications and one comparison, suffices (figure III.4).

A circular arc is traced in a similar manner. If the arc traverses more than one quadrant it is broken into subarcs each traversing all or part of a single quadrant. When the arc or subarc traverses the northwest quadrant almost the same procedure is used as for the case of a line whose direction is in the northeast quadrant (figure III.4). As for the line case, the next current square is either one up, one right, or one diagonally up and right from the current square, and this choice is made by computing whether the northeast corner of the current square is inside, outside, or on the arc. This requires two multiplications, an addition, and a comparison.

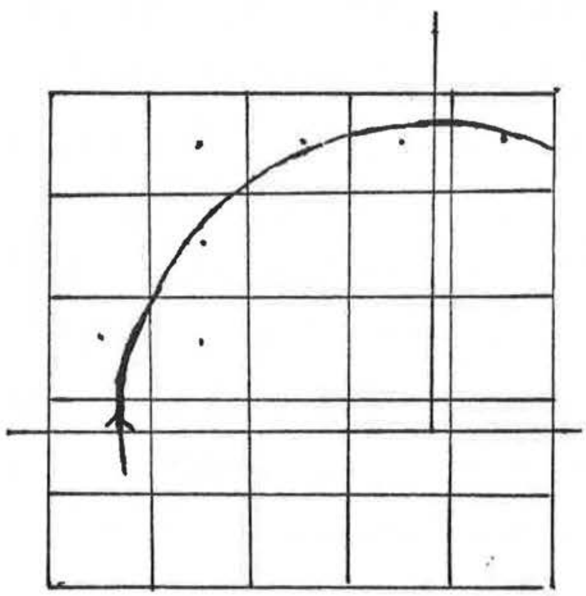
There are several operations which may be applied to the squares that a line passes through. The square coordinates may be added to a scan table for a scanning routine or, if the square is non-empty and hence represents an obstruction, an intersection computation may be executed and the line-tracing procedure abandoned.

For the purpose of projecting the Cartesian representation of a concave object into its digital representation on the TABLE (briefly, <u>drawing</u> the object), and erasing it later, the Cartesian representation is decomposed into convex subparts. This is done manually when the object is first specified. When

FIGURE III.4



Tracing a straight line in the north-east quadrant.



Tracing an arc in the north-west quadrant.

the object is drawn or erased each convex subpart is drawn or erased separately.

The digital representation of a convex (subpart of an) object is constructed row by row. First the edges in the Cartesian representation are traced, and the coordinates of the squares encountered are used to update a <u>scan table</u> that records the coordinates of the squares at the left and right extremities of each row. Since the size of the scan table is sufficient to cover only the vertical extent of the convex shape, an offset is also stored that specifies the row of TABLE that corresponds to the first pair of (left and right) scan table entries. The scan table is then used to draw the digital representation row by row. If the object is fixed the scan tables are then discarded. If the object is movable, the scan tables and offsets are stored for later use when, or if, the object is erased for a pushto or turn action. A new scan table has to be constructed for each convex subpart every time the object is redrawn.

The parallelogram swept out by a leading edge in a pushto action is scanned in almost identical fashion (see figure III.1). The edges of such a parallelogram are traced in the sequence: leading edge AB, left constraining edge AA', right constraining edge BB', destination edge A'B'. No obstructing square can occur along the edge AB. If an obstructing square is encountered while tracing the edge AA', then the minimum distance in the direction θ from the leading edge AB to the nearest part of the obstructing square that lies within the parallelogram is computed. This is taken as the new value of d, the amount of the translation. Similarly if an obstacle is encountered while tracing BB'. If either or both of these cases occur then the position of the destination edge is effectively moved closer to the original position AB. Now the destination edge, at its possibly new position, is traced and the scan table for the parallelogram is complete. The TABLE squares within the parallelogram are now scanned and if an obstructing square is found the distance from the leading edge AB in the direction θ to the nearest corner of the obstacle is computed.

The doughnut slice swept out by a leading segment in a turn action has one concave bounding line -- the inner constraining arc. However, since a shape is scanned row-by-row this is of no consequence provided the arc does not cross the vertical line through the point of rotation. If this condition holds then the doughnut slice is cut along this vertical line (line UU' in figure III.3) and each part formed is scanned separately. The edges of a doughnut slice are traced in the sequence: leading segment AN, inner constraining arc NN', outer constraining arc AA', destination segment A'N'. No obstructing square can occur along AN. If an obstructing square is encountered while tracing the arc NN' then the minimum angle of rotation about U to the nearest part of the obstructing square within the doughnut slice is computed. This is taken as the new value of ϕ, the intended rotation. Similarly if an obstacle is encountered while tracing AA'. If either or both of these cases occur then the position of the destination segment is effectively rotated back closer to the original position AN. Now the destination segment at its possibly new position is traced and the scan table for the doughnut slice is complete. The TABLE squares within the doughnut slice are scanned and collision computations carried out if any obstructing squares are found.

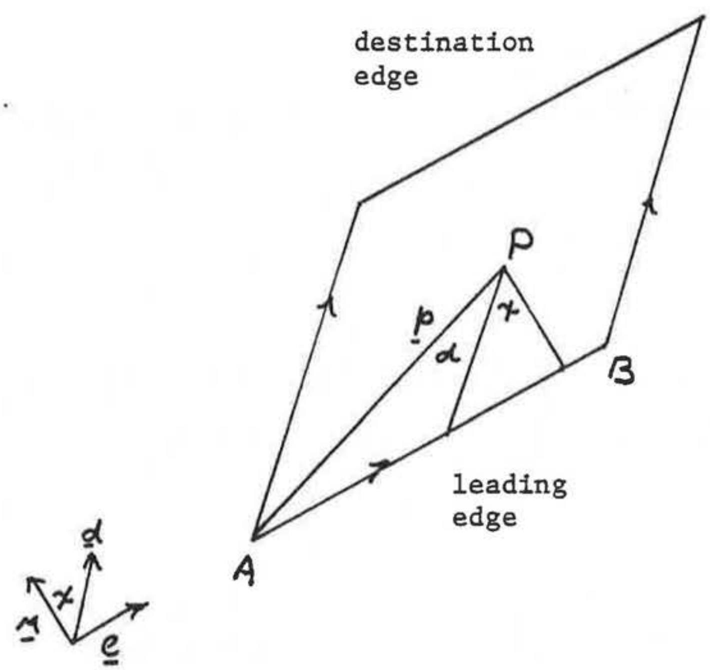
Finally I must specify the collision computations. First I describe them for a pushto action, then for a turn action.

The simplest case is this: when scanning the parallelogram swept out by a leading edge AB, an obstructing square is encountered. The amount of movement of the object before AB collides with the nearest point P of the square must be calculated (figure III.5). This is the distance to collision for this edge. The nearest point of a square is one of the corners of the square. This nearest corner depends only on the quadrant of the leading edge so a simple table lookup is used to find it. For instance, for a leading edge in the northeast quadrant the nearest corner of an obstructing square is the southeast corner. Let $\underline{n}$ be a unit vector in the direction of the outward normal to AB, let $\underline{d}$ be a unit vector in the direction of motion, and let $\underline{p}$ be the vector AP. Then the distance to collision is given by the formula

$$(\underline{p}.\underline{n}) \;/\; (\underline{d}.\underline{n}) \qquad \text{(A)}$$

If an obstructing square is encountered while tracing a

FIGURE III.5 - A formula for the distance from a leading edge to the nearest point of an obstructing square.



$\underline{e}$ = unit vector along AB.

$\underline{n}$ = unit outward normal to AB.

$\underline{d}$ = unit vector in direction of motion.

$\underline{p}$ = vector AP.

AB is a leading edge $\Longleftrightarrow |\chi| < \frac{\pi}{2} \Leftrightarrow \cos\chi > 0 \Longleftrightarrow \underline{d}.n = \cos\chi > 0$

PB = perpendicular distance from AB to P = $\underline{p}.\underline{n}$

$\frac{PB}{d'} = \cos\chi \Rightarrow d' = (\underline{p}.\underline{n}) \; / \; (\underline{d}.\underline{n})$

constraining edge of a parallelogram, more care is required to find the distance to collision. The nearest corner of the obstructing square may lie outside the parallelogram or even on the opposite side of an extension of the leading edge. When tracing the left constraining edge of the parallelogram, four cases arise (figure III.6).

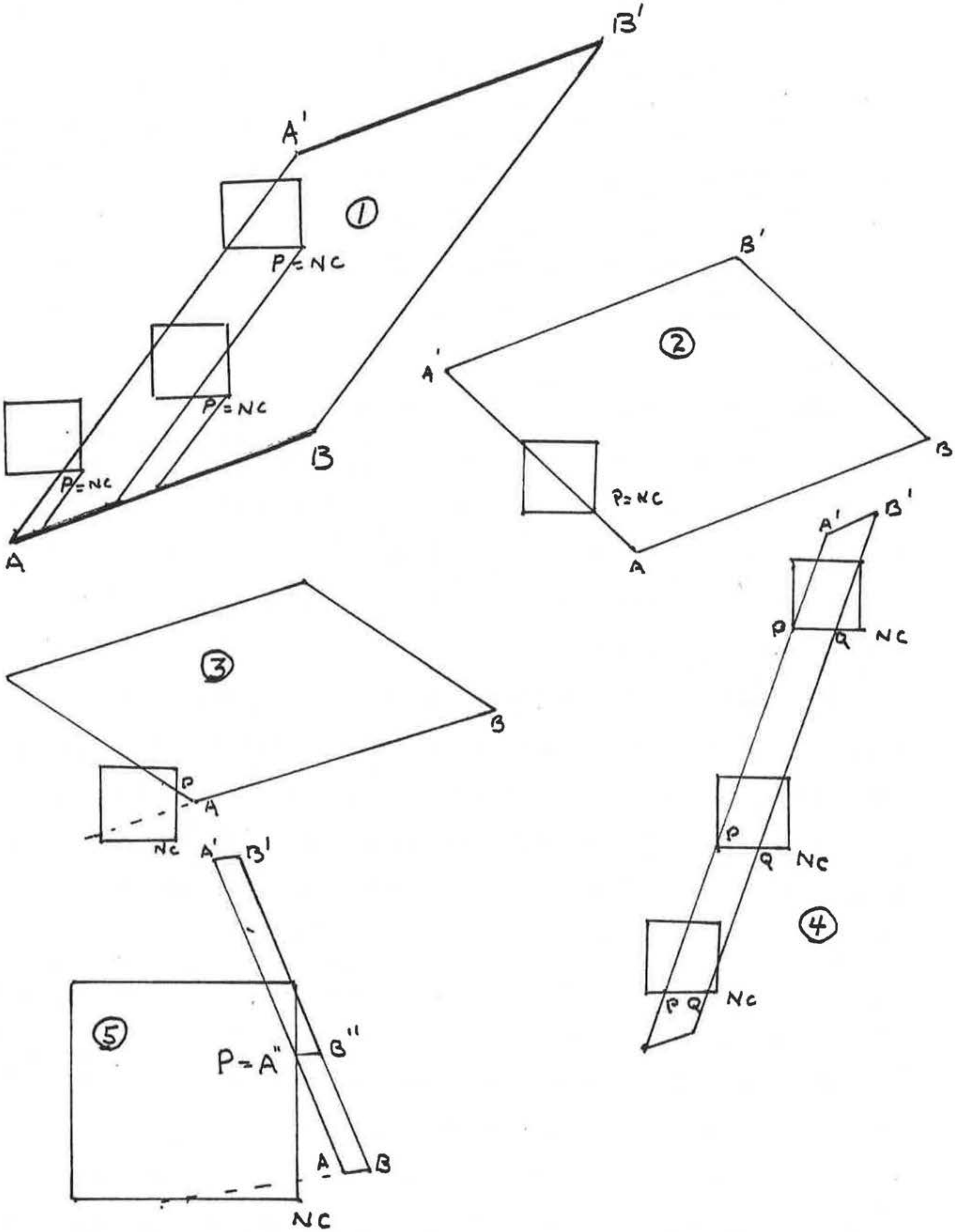
(1) The nearest corner P of the obstructing square lies between the left and right constraining edges. The distance to collision is the same as before, using equation (A).

(2) The nearest corner lies on the left constraining edge. The distance to collision is the distance from A to the nearest corner.

(3) The nearest corner lies to the left of the left constraining edge. The distance to collision is the distance from A along the left constraining edge to the point P where the left constraining edge intersects the side of the square.

(4) The nearest corner lies to the right of the right constraining edge. The distance to collision is the distance from B along the right constraining edge to the point Q where the right constraining edge intersects the

FIGURE III.6



Cases in computing distance to collision along a constraining edge.
AB = leading edge.
NC = nearest corner of obstructing square.
P = nearest point of obstructing square.

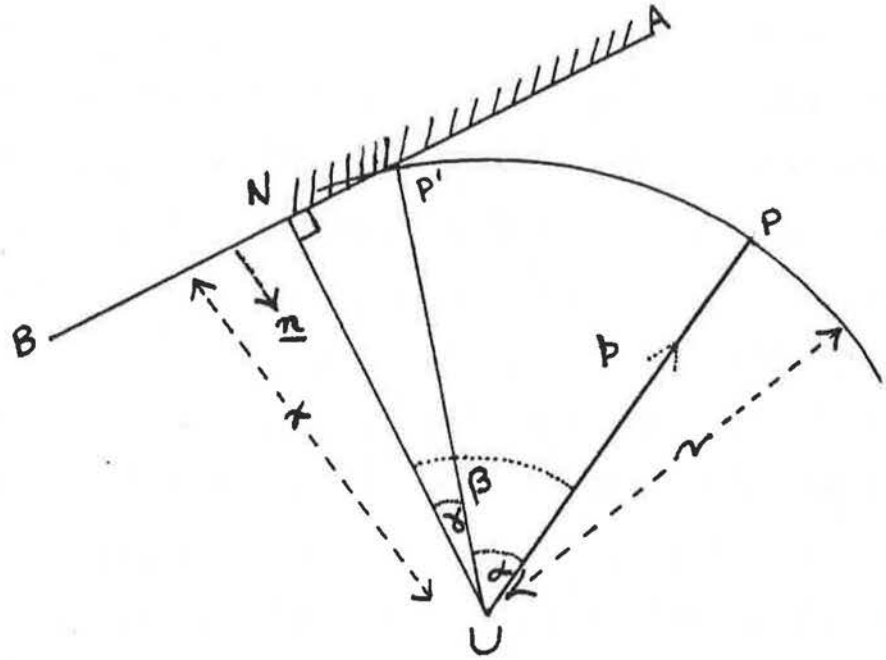
side of the square. There are really two subcases involved here depending on whether P and Q lie on the same or adjacent edges of the square. However, it is not necessary to go to the trouble of figuring out the distance BQ since this will be computed under case (3) when the right constraining edge is being traced. So it suffices to return the distance AP.

Notice that, when tracing the right constraining edge BB', case (4) [with right and left transposed] could not occur (figure III.6(5)). This is because the distance AP would have been returned from an occurrence of case (3) when tracing the left constraining edge, and so the right constraining edge would only be traced as far as B''.

There are four cases when tracing the left constraining edge, three cases when tracing the right constraining edge, and these are all repeated for each of the other three quadrants in which the leading edge may lie. These 28 cases can be handled by one 2 x 4 x 4 decision table with one row of four null entries.

Now I describe the collision computations for a turn action. Suppose that an obstructing square is encountered when scanning the doughnut slice swept out by a leading segment. The amount of rotation of the object before the leading segment AN of an edge AB collides with some point of the square must be calculated (figure III.7). This is the <u>angle to collision</u> for this edge. The point of the square at which the collision

FIGURE III.7



$$\alpha = \beta - \gamma$$

$$\cos\beta = \left(\frac{\underline{p}}{r}\right)\cdot(-\underline{n}) = -(\underline{p}\cdot\underline{n})/r$$

$$\cos\gamma = x/r$$

$$\alpha = \cos^{-1}\left[-(\underline{p}\cdot\underline{n})/r\right] - \cos^{-1}\left[x/r\right]$$

AB = object's edge.

AN = leading segment.

U = centre of rotation.

P = obstructing point.

P' = point which coincides with P at the nearest moment of collision.

r = UP = UP'.

α = angle to collision.

$\underline{n}$ = unit outward normal to AB.

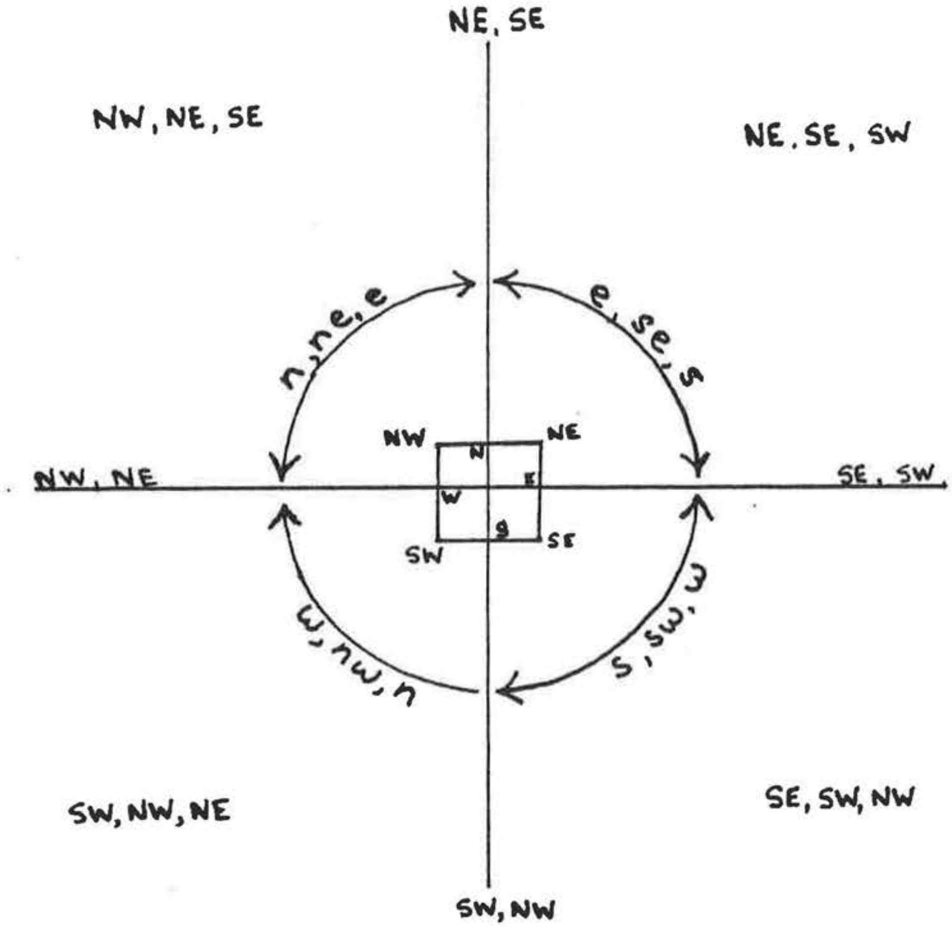
x = distance UN.

FIGURE III.7. The diagram shows the point P', on the edge AB, which coincides with the point P at the moment of collision. The formula shows how to compute PUP', the angle to collision.

occurs is the <u>collision point</u>. I also call this the <u>collision corner</u> since it is clear that the collision point must be a corner of the obstructing square. Unfortunately it is not trivial to determine which corner of the square is the collision corner. For instance, as the obstructing square varies over the squares within a doughnut slice the collision corner varies too. If the rotation is clockwise and the obstructing square is moved clockwise then the collision corner of the obstructing square moves in a clockwise direction relative to the centre of the obstructing square. It is possible to specify sets of candidate collision corners as a function of the position of the centre of rotation U relative to the obstructing square. The set of candidate collision corners for a specific leading segment contains either two or three corners. Suppose axes are taken at the centre of the obstructing square and aligned with the grid lines. If U is on one of the axes there are two candidate collision corners and if U is in a quadrant between the axes there are three candidate collision corners (figure III.8). For instance, if U lies in the southwest quadrant the candidate collision corners are the southwest, northwest, and northeast corners, and if U lies on the south vertical axis the candidate collision corners are the southwest and northwest corners. Figure III.9 shows occurrences of each of the candidate collision corners, for U in the southwest quadrant.

The angle to collision with an obstructing square is determined by finding the angle to collision with each of the

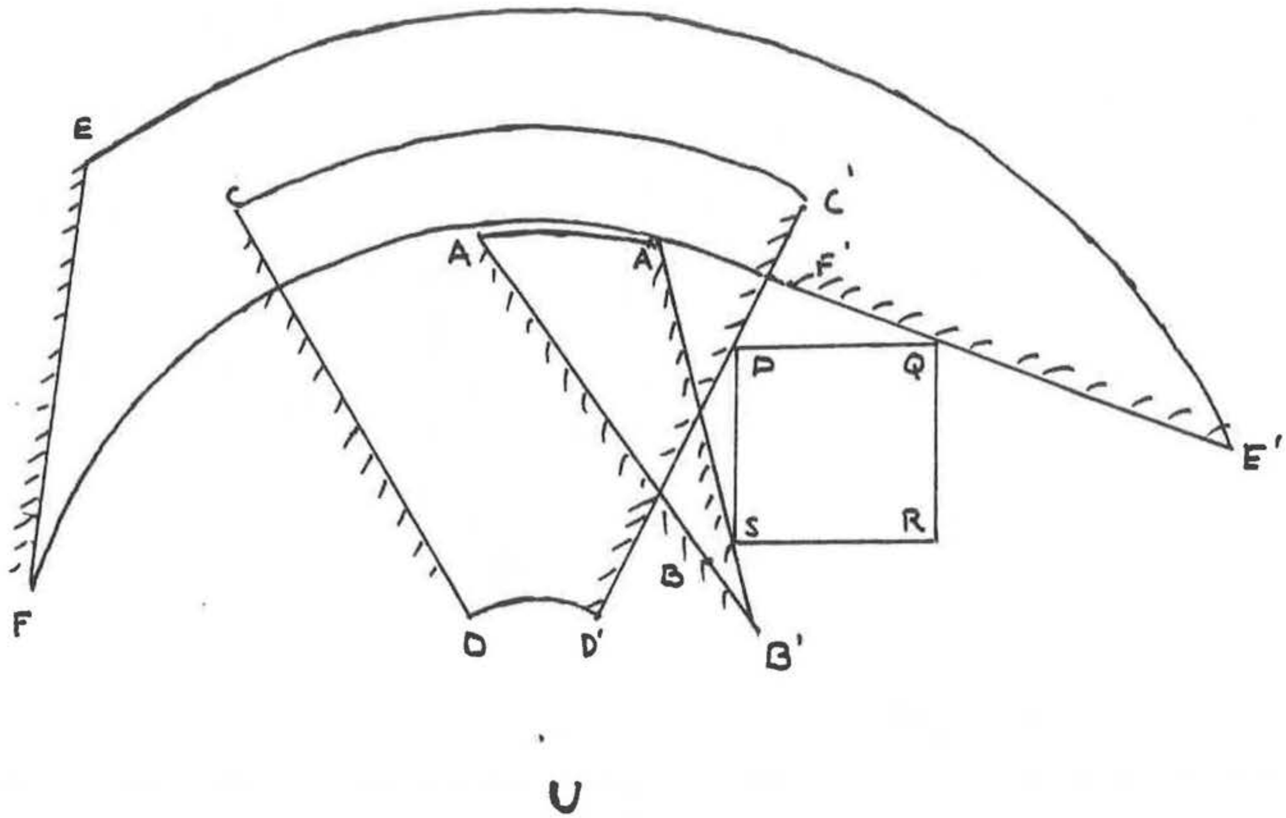
FIGURE III.8



This shows, for a clockwise rotation, how the set of candidate collision corners of an obstructing square varies as a function of the position of the centre of rotation relative to the axes of the square. This is the set of collision corners that must be considered if the obstructing square is encountered during the scan of the doughnut slice.

Along the arcs are shown the edges or corner that may be involved if the obstructing square is encountered when tracing a constraining arc.

FIGURE III.9



For a clockwise rotation about the point U in the southwest quadrant relative to the axes of an obstructing square, this shows how each of the candidate collision corners could actually occur. Leading segment AB collides with the southwest corner, CD collides with the northwest corner, and EF collides with the northeast corner. A'B', C'D', and E'F' are the positions of AB, CD, and EF, respectively, at the moment of collision.

collision corners separately and taking the minimum of the three (or two) angles. The collision corner of the obstructing square is the corner with the smallest angle-to-collision.

Given a leading segment AN of an edge AB being rotated about a point U, the angle to collision with a point P is found as follows. Let $\underline{n}$ be the unit vector in the direction of the outward normal to AB, let x be the perpendicular distance from U to AB, let r be the radius UP, let $\underline{p}$ be the vector $\underline{UP}$, let P' be the point on AN which coincides with P at the moment when the collision occurs, and let alpha be the angle to collision (figure III.8). Then the following holds.

$$\text{alpha} = \angle PUN - \angle P'UN$$

$$\cos(\angle PUN) = (\underline{p}/r)\cdot(-\underline{n}) = -(\underline{p}\cdot\underline{n})/r$$

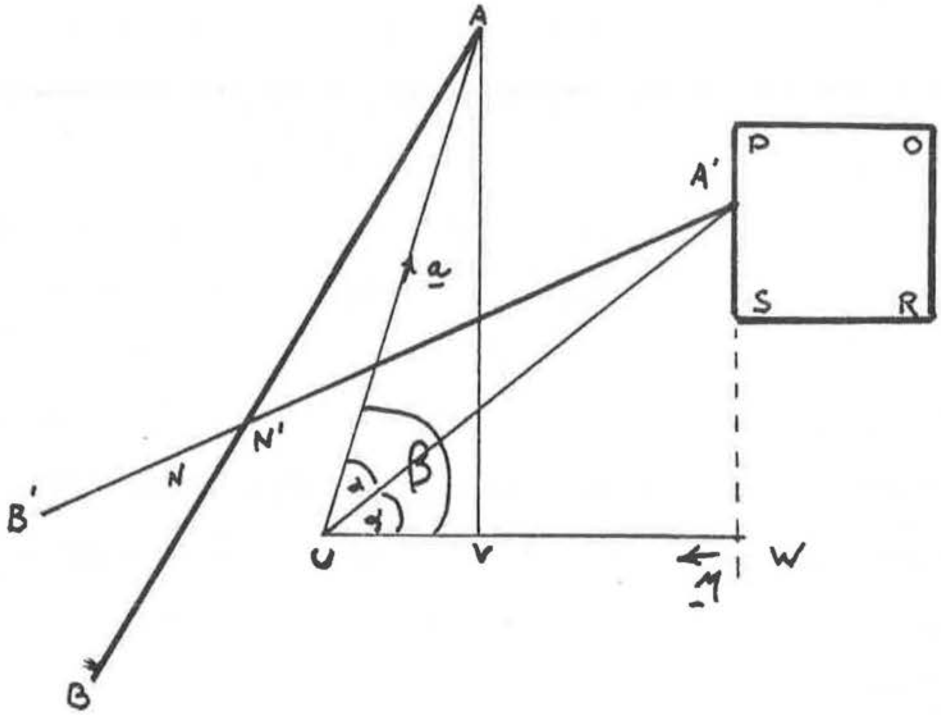
$$\cos(\angle P'UN) = x/r$$

$$\text{alpha} = \arccos[-(\underline{p}\cdot\underline{n})/r] - \arccos[x/r] \qquad (B)$$

If an obstructing square is encountered while tracing the inner or outer constraining arc, the collision computation is slightly simpler. Instead of rotating the candidate collision corner backwards to where its arc intersects the leading segment as in the usual case, here one has to rotate the endpoint of the leading segment forwards to where its arc intersects a side of the obstructing square. The computation is simpler because the sides of the square are aligned with the coordinate axes.

I describe this collision computation only for an example involving the outer constraining arc (figure III.10). Suppose the point of rotation U lies in the southwest quadrant relative

FIGURE III.10



AB = object's edge.

AN = leading segment.

U = centre of rotation.

A' = point of collision with obstructing square.

r = UA = UA'.

$\underline{a}$ = vector UA.

α = angle to collision = ∠A UA'.

UW = perpendicular from U to collision side of square.

$\underline{n}$ = unit outward normal from collision side.

x = distance UW

$$\alpha = \beta - \gamma$$

$$\cos\beta = \left(\frac{\underline{a}}{r}\right)\cdot(-n) = -(\underline{a}\cdot n)/r = -\frac{UV}{r}$$

$$\cos\gamma = \frac{UW}{r}$$

$$\alpha = \cos^{-1}\left(-(\underline{a}\cdot\underline{n})/r\right) - \cos^{-1}(x/r)$$

$$= \cos^{-1}\left(-\frac{UV}{r}\right) - \cos^{-1}\frac{UW}{r}).$$

FIGURE III.10 - The diagram shows the intersection A' of an outer constraining arc with an obstructing square. The formula shows how to compute the angle of collision.

to the axes of symmetry of the square, and the endpoint A of a leading segment AN collides with a side of the square. The side involved is the collision side, and is specified by the arc tracing routine. In the example shown the collision side is the west side of the square. Instead of dropping a perpendicular from U to AN, for this computation one drops a perpendicular from U to the collision side, meeting it at W. Let V be the perpendicular projection of A onto UW. Then alpha, the angle to collision, is given by

alpha = arccos[-UV/r] - arcos[UW/r] (C)

The arc tracing routine may, instead, specify that the arc enters the obstructing square at the corner P. The coords of P are known so the angle to collision is then simply given by

alpha = 2 * arcos[AP/(2*r)] (D)

When tracing the constraining arcs of a doughnut slice at most two obstructing squares are encountered, since an arc-tracing routine is abandoned as soon as an obstacle is found. For such an obstructing square there are four distinct relative positions of the centre of rotation U, and for each of these there are three different ways in which the arc may intersect with the square. Each of these twelve cases reduces to one or other of the computations specified by equations (C) and (D). A 4 x 3 decision table specifies the correct procedure.

\+ \+ \+

To sum up this section so far, I have guided you through a collection of algorithms sufficient to solve the basic problems involved in an implementation of TABLETOP. These include algorithms for scanning an object's shape, for scanning the parallelograms and doughnut slices swept out in the course of pushto and turn actions respectively, algorithms for tracing straight lines and circular arcs, and algorithms for computing the exact distance to collision, or angle to collision, for these two basic actions.

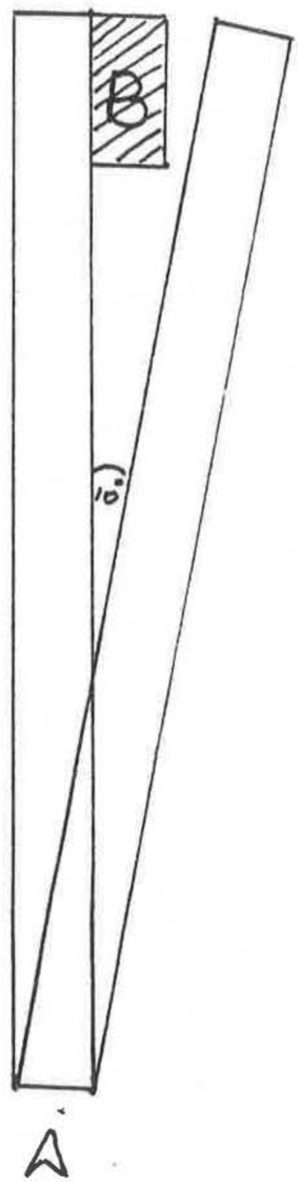
The algorithms given above for the turn action in TABLETOP requires inverse cosines and extensive case analysis. The former are not computationally cheap, the latter requires careful and time-consuming coding. Consequently I used a quicker, but dirtier and computationally more expensive, method of implementation. This is the second method referred to on page 92. Namely, when an object is to be rotated the turn procedure actually implemented does the following. The digital representation is erased, the Cartesian representation rotated by 10°, and its new projection onto TABLE scanned for obstructing squares (without actually drawing the digital representation). This is repeated until an obstructing square is encountered or the intended angle is achieved. If an obstructing square is found, a binary search is carried out over the last sub-angle of rotation until the achieved position is located to within 2° accuracy.

This method leaves open the possibility that some small obstacle may be jumped over between the 10° test positions. This has not yet happened in practice, partly because only very simple TABLETOP environments have been used that do not contain small isolated obstacles, and partly because the size of the movable objects has not been sufficiently large for an obstacle to be missed in a 10° rotation. The simplest configuration in which such an incident could occur would have Utak grasping the narrow edge of a 1 x 14 movable object or "stick", an obstacle whose digital representation occupies a single square of TABLE at a distance of about 13.5 from Utak, and Utak executing a turn action of more than 10° towards the obstacle (figure III.11).

An important implementation problem arises in practice. When many pushto and turn actions are applied to a movable object, the Cartesian representation of the object becomes deformed due to cumulative floating point inaccuracies. Thus what was originally a square may at some later time look like the end view of a squashed cardboard box.

To overcome this problem three Cartesian style representations are used for an object, not just one. When the object is first specified a <u>base representation</u> is set up. This is its original Cartesian representation. After any arbitrary sequence of actions the current Cartesian representation can always be represented as one rotation and one translation applied to the base representation. Since rotations are relatively uncommon, and are computationally expensive, a <u>rotated base representation</u> is also used. This consists of the

FIGURE III.11



This shows the simplest kind of situation in which a potential obstacle (marked 'B') could be missed by the algorithm actually implemented. Utak is holding a long stick and when he executes a turn action to the right of at least 10°, the obstacle is missed by the algorithm.

base representation rotated by the angle between it and the current Cartesian representation. During a sequence of translations between two rotations, the Cartesian representation at the end of each translation is derived from the rotated base representation that was computed at the last rotation. When a rotation occurs a new rotated base representation is computed directly from the base representation. The current Cartesian representation at this point is then obtained by one translation from the new rotated base representation. With this scheme implemented the Cartesian representation of a movable object cannot deform, however many pushto and turn actions are applied to the object.

One final problem remains to be considered.

III.1.2.1 The overlay problem

A problem involving the interaction of Cartesian and digital representations crops up occasionally when Utak is holding and moving an object. I refer to the system formed by Utak and a held object as a sum object. When he first grasps an object his digital representation is necessarily outside and contiguous with the object's digital representation. At that time, the Euclidean distance d between his Cartesian position U and the nearest point N on the object's Cartesian representation must have a value strictly between 0 and 2*root2. The distance d remains invariant over all further movements of the sum object until Utak executes a letgo action followed by a slide action. When d < root2 it is clearly possible to position the sum object such that the points U and N in the Cartesian representations lie in the same square of TABLE. Consequently it is possible for the digital representation of Utak to coincide with a square in the held object's digital representation. This is of no concern while Utak continues to hold the object since it does not affect the computations involved in pushto and turn actions. The problem arises if this situation occurs immediately before Utak lets go of the object. Suppose he lets go and then tries to execute a slide action. The slide routine finds that Utak's starting point lies within a square belonging to the digital representation of an obstacle, and therefore fails! If there are empty TABLE squares next to Utak's square then the following procedure handles the problem.

1. Let the value of BUGSQUARE be the coordinates of the TABLE square currently occupied by Utak, let OLDBUGSQUARE be the value of BUGSQUARE at Utak's previous position, let BUGMARK be the colour assigned to Utak's digital representation on TABLE, and let OVERLAY be the overlying colour of the BUGSQUARE. The value of OVERLAY is the colour empty except when the BUGSQUARE is within the held object's digital representation.

2. After a pushto or turn action, first draw the grasped object, then set

   OVERLAY = COLOUR-OF(BUGSQUARE)
   COLOUR-OF(BUGSQUARE) = BUGMARK

3. If Utak is about to execute a slide and if OVERLAY $\neq$ empty, then do:

   COLOUR-OF(BUGSQUARE) = empty
   OLDBUGSQUARE = BUGSQUARE

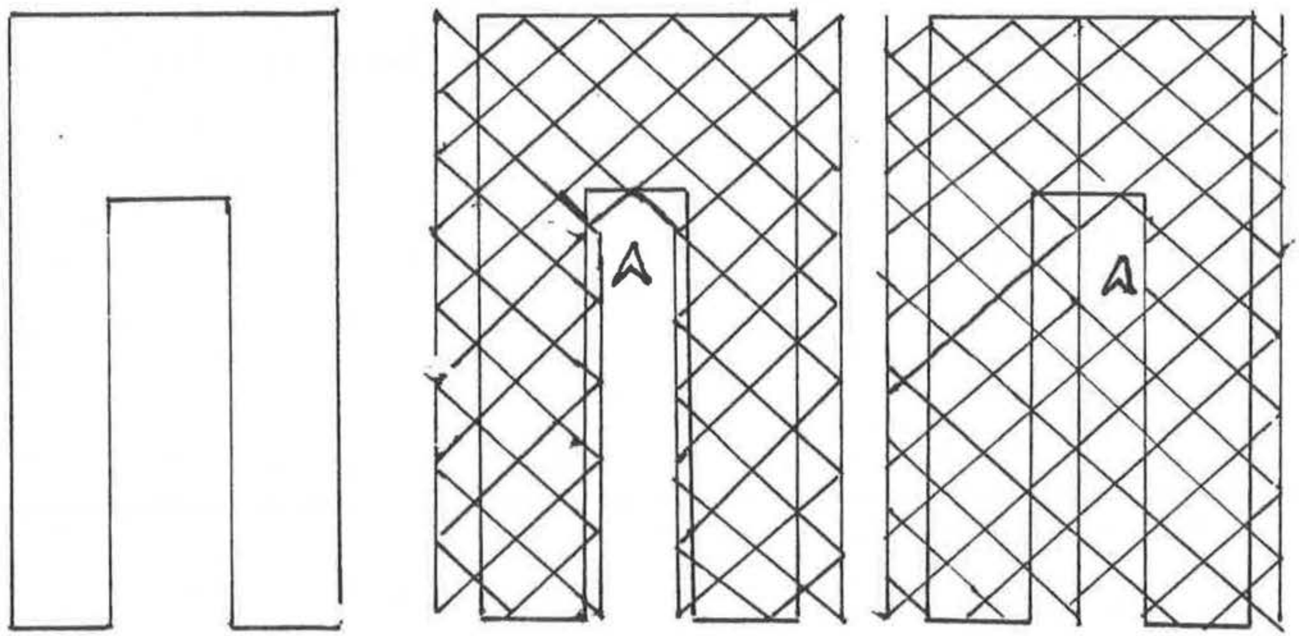
   Compute the result of the slide.

If achieved position still lies within OLDBUGSQUARE
then COLOUR-OF(OLDBUGSQUARE) = BUGMARK
else COLOUR-OF(OLDBUGSQUARE) = OVERLAY

This procedure has proved sufficient in practice but does not solve the problem in general. In fact the digital representation of Utak can lie arbitrarily far within the digital representation of the held object. This can arise if there is a long straight "canal" of width strictly between 1 and 2 units in the object's Cartesian representation. Let the canal have length n+1. In one position of the object the canal may lie astride a column of squares. Then Utak could slide to the head of the canal and grasp the object there. In another position of the object the canal may not straddle any whole TABLE squares, so that the canal does not appear in the digital representation. Now suppose that Utak has grasped the object in the first type of position and has let it go in the second type of position. When he wants to execute a slide, Utak is hopelessly trapped with n squares of the object's digital representation between him and the empty TABLE squares outside. This is illustrated in figure III.12.

One solution that immediately springs to mind and can be easily implemented within the current TABLETOP philosophy is the following. Before a grasp action can be executed the 20 closest squares to Utak's square must be empty and some square in the ring of 24 next closest squares must belong to the object's digital representation. This is shown in figure III.13. This ensures that no point of the subsequent held object lies within

FIGURE III.12



(a) (b) (c)

FIGURE III.12 - (a) The Cartesian representation of an object, with a canal of width 1.5.

(b) The object positioned so that the canal is open in the digital representation. Utak is able to slide to the head of the canal and grasp the object there.

(c) The object positioned so that the canal is closed. If Utak now lets go the object, no slide action can get Utak beyond the borders of his current digital representation. He is trapped.

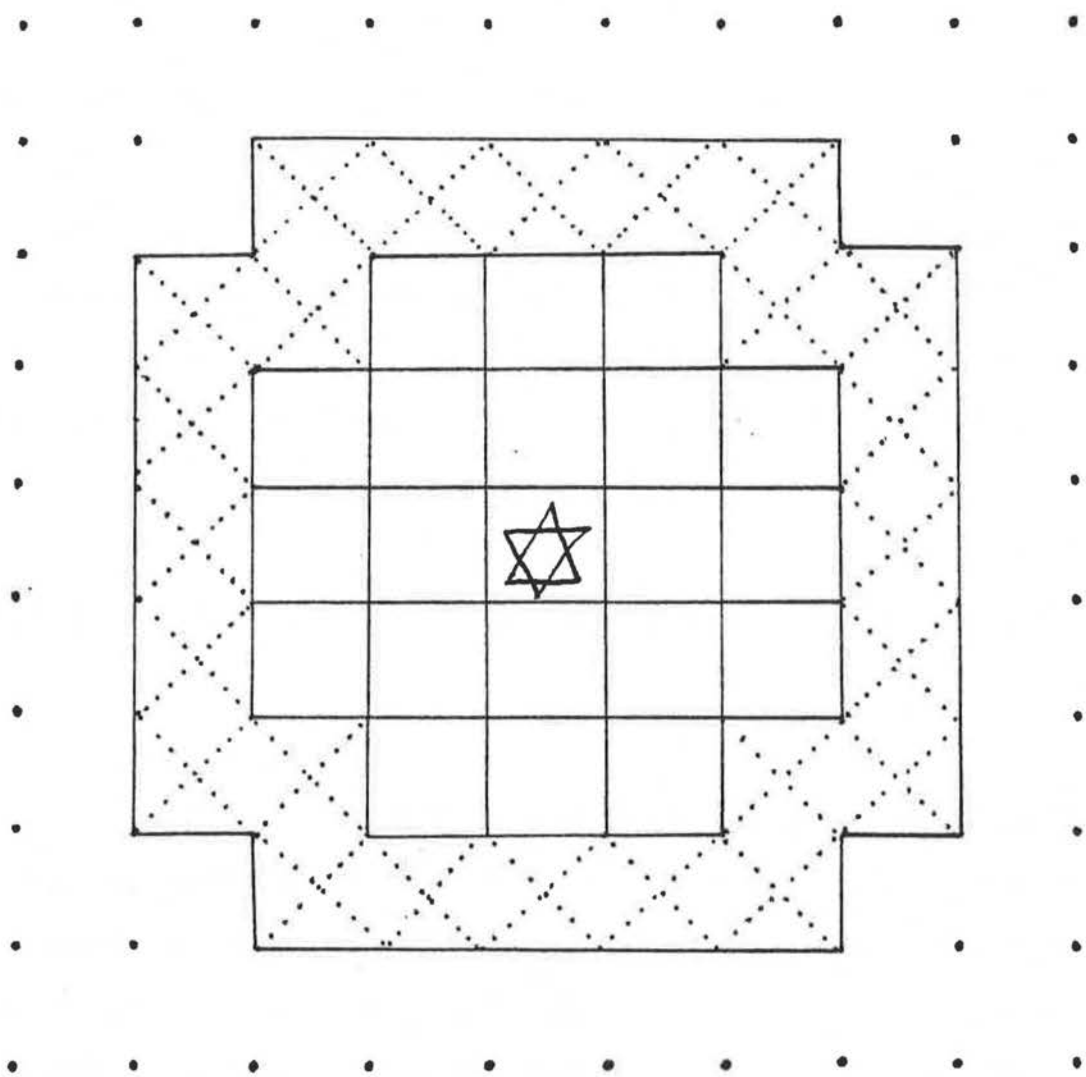


FIGURE III.13 - One solution to the overlay problem is shown here. Before Utak can successfully grasp an object, two conditions must hold. (1) The 20 squares of TABLE closest to Utak must be empty. (2) At least one of the 24 squares forming a ring around the 20 closest squares must belong to the object's digital representation.

root2 units of Utak's position, and thus that the above "overlay" problem cannot arise.

Another solution would require that, in the Cartesian representation of Utak and the object to be held, Utak was not within root2 units of any edge of the object. This however would require closest edge calculations in Cartesian coordinates, a type of calculation that I wish to try and avoid as much as possible.

Now I shall show some examples of the TABLETOP program in action.

<u>III.1.3 An example of TABLETOP performance</u>

The following pages, figure III.14, show excerpts from a session with TABLETOP recorded during UBC's Open House 1979, slightly edited. It shows the state of the TABLE array as a human user solved the L-shaped object problem. The first snapshot includes a statement of the task, which is read by the user, not Utak! The subsequent snapshots show:

- an action command issued by the user
- the resulting state of the TABLE array, with the BUGMARK '*' showing Utak's digital representation
- the contents of the nine TABLE squares in the 3 x 3 array centred on Utak. (Thus the centre square shows the contents of the OVERLAY variable.)
- the typed response from TABLETOP.

{I have added comments within braces like this.}

```
>   TRANSLATE A BY (-10 0) I. E. MOVE IT UP BY 10 UNITS

B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B   B B B B B B B B B B B B B B B B B B B B B B B B       B
B                                                         B
B         A                                               B
B         A                                               B
B   *     A                                               B
B         A                                               B
B         A                                               B
B         A A A                                           B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B

...
...
...

>   OK
```

FIGURE III.14

(GRASP)

```
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B   B B B B B B B B B B B B B B B B B B B B B B B B       B
B         *                                               B
B         A                                               B
B         A                                               B
B         A                                               B
B         A                                               B
B         A                                               B
B         A A A                                           B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B

BBB
...
.A.
```

>   GOT IT!

(PUSHTO 6.0 N)

```
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B   B B B B B B B B B B B B B B B B B B B B B B B B * A   B
B                                                   A A   B
B                                         A A A A A A A   B
B                                         A A A A A A A   B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B

...
BAA
.AA

>   HIT!
```

```
(PUSHTO 1.0 W)

B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                   A     B
B                                                 * A A   B
B                                                   A A A B
B   B B B B B B B B B B B B B B B B B B B B B B B B A A A B
B                                               A A A A   B
B                                             A A A A     B
B                                             A A A       B
B                                             A A         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B

..A
.AA
..A

>   HIT!
```

(PUSHTO 5.0 N)

```
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                 A A A A B
B                                                 * A A A B
B                                                     A A B
B                                                     A A B
B                                                     A A B
B                                                     A A B
B                                                     A A B
B                                                         B
B   B B B B B B B B B B B B B B B B B B B B B B B B       B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B                                                         B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B

.AA
.AA
...

>   OK
```

III.2 The simulated organism Utak and his tasks

III.2.1 Design considerations

Now that I have described the simulated environment, what kind of organism should be built to live there, and what kind of tasks should the organism be required to execute? These two questions are closely linked since, for example, you cannot expect a colour blind human to respond to traffic lights correctly unless other cues such as light position are available. More precisely, what kind of actions should the organism be capable of executing and what kind of sensory input should the organism receive from its environment? The kinds of input/output allowed to the organism, and the kind of task he is required to solve, both mould to a certain extent the design of the mediating mechanism, or organism-controller, that lies between input and output.

Asking these questions of design immediately raises many factual and methodological questions. How animal-like should the organism be? If so, what animal? If not, by what criteria are the design decisions to be made? If the simulated organism is to be like some specific animal, exactly what sensory input and motor output messages does that animal's brain receive and send?

Since I am interested in the principles of behaviour of animals and humans, the organism should definitely be animal-like. The subsequent questions about sensory-motor I/O are, unfortunately, virtually unanswerable; in no case are all the relevant facts for one interesting animal known. One might, instead, fall back upon the gross behaviour of an animal, but again, in almost no case are all the relevant facts known!

This situation has recently improved. M.J.Wells has collected a wealth of information about the physiology and behaviour of the octopus [Wells,1978], and E.R.Kandel has published a comparable collection of information about an even simpler creature, the marine snail Aplysia [Kandel,1978]. With a view to receptor design, I have perused various known facts about receptors in mammals, including visual receptor densities, receptive field sizes, and magnification factors in the brain.

One has to resort to criteria of elegance, naturalness, and finally, in the AI approach, on the criterion of computational feasibility. This is in contrast to most other experimental sciences, where the final arbiter is experimental feasibility.

My attitude was well put by the philosopher Daniel Dennett when he wrote [Dennett,1978,p.104]

> " one does not want to get bogged down with technical problems in modeling the cognitive eccentricities of turtles if the point of the exercise is to uncover *very* general, *very* abstract principles that will apply as well to the cognitive organization of the most sophisticated human beings. So why not then *make up* a whole cognitive creature, a Martian three-wheeled iguana, say, and an environmental niche for it to cope with? I think such a project *could* teach us a great deal about the deep principles of human cognitive psychology, but if it could not, I am quite sure that most of current A.I. modeling of familiar human mini-tasks could not

either. "

Utak is my three-wheeled Martian iguana and TABLETOP is his niche.

III.2.2 The sensory-motor capabilities of Utak

Utak can move in a straight line, he can grasp a movable object, and he can push, turn, and letgo a held object. He has five motor outputs, only one of which is active at once. Thus the design of his motor output immediately contradicts one of the obvious features of the design of natural motor output. This is the fact that exactly one output has to be active to execute an action, whereas large numbers of output lines are active when a biological system acts. Indeed, an alert biological system could be described as a pattern transducer that transforms a never-ending series of input patterns on millions of input lines into another never-ending series of patterns on a similarly large number of output lines. This, of course, is very different from a typical present-day computer which channels all information through a bottleneck formed by the single high-speed CPU. It seems unlikely that we can get close to understanding the computations occurring in biological systems if we allow the "bottleneck" design of current computers to influence our thinking.

The visual sensory input is somewhat more realistic, consisting of 160 input lines from 160 retinal receptors. Utak gets a bird's eye view directly down on his immediate environment as though his eye were on a stalk. The retinal fields are arranged as follows: 64 in a central fovea, 48 fields each 4 times the size of a foveal field in an intermediate zone surrounding the fovea, and 48 fields each 16 times the size of a foveal field in an outer, peripheral, zone. Each retinal cell registers a 3-bit graylevel, or integer in the range 0-7, that reflects the ratio of object to total area in the part of the tabletop covered by the cell's field (figure I.3). The colour of an object is ignored. A set of 160 graylevels constitutes one retinal impression. Object colours can be sensed via a tactile impression, which consists of the colours of the 8 adjacent squares to Utak.

The Utak simulation computes a new retinal and tactile impression each time Utak comes to a halt. To do so it superimposes the array of retinal fields on TABLE, centred on Utak's digital representation, and computes the graylevels directly from the TABLE array. An action by Utak will be reflected by a change in the retinal impression only if the digital representation of Utak or of an object held by him changes. More sophisticated visual sensory systems are easy to propose, but this one is computationally cheap and has sufficed so far.

The concepts of speed, mass, acceleration, and momentum have not been implemented. As the reader will see, the attempt to design an organism-controller for this simple world raises an ample range of interesting and fundamental problems in perception and action without the addition of these extra features. Nonetheless, the distance between successive retinal impressions may be viewed as a measure of speed if one assumes that retinal impressions are received at a constant rate.

III.2.3 Examples of Utak's sensory-motor experience

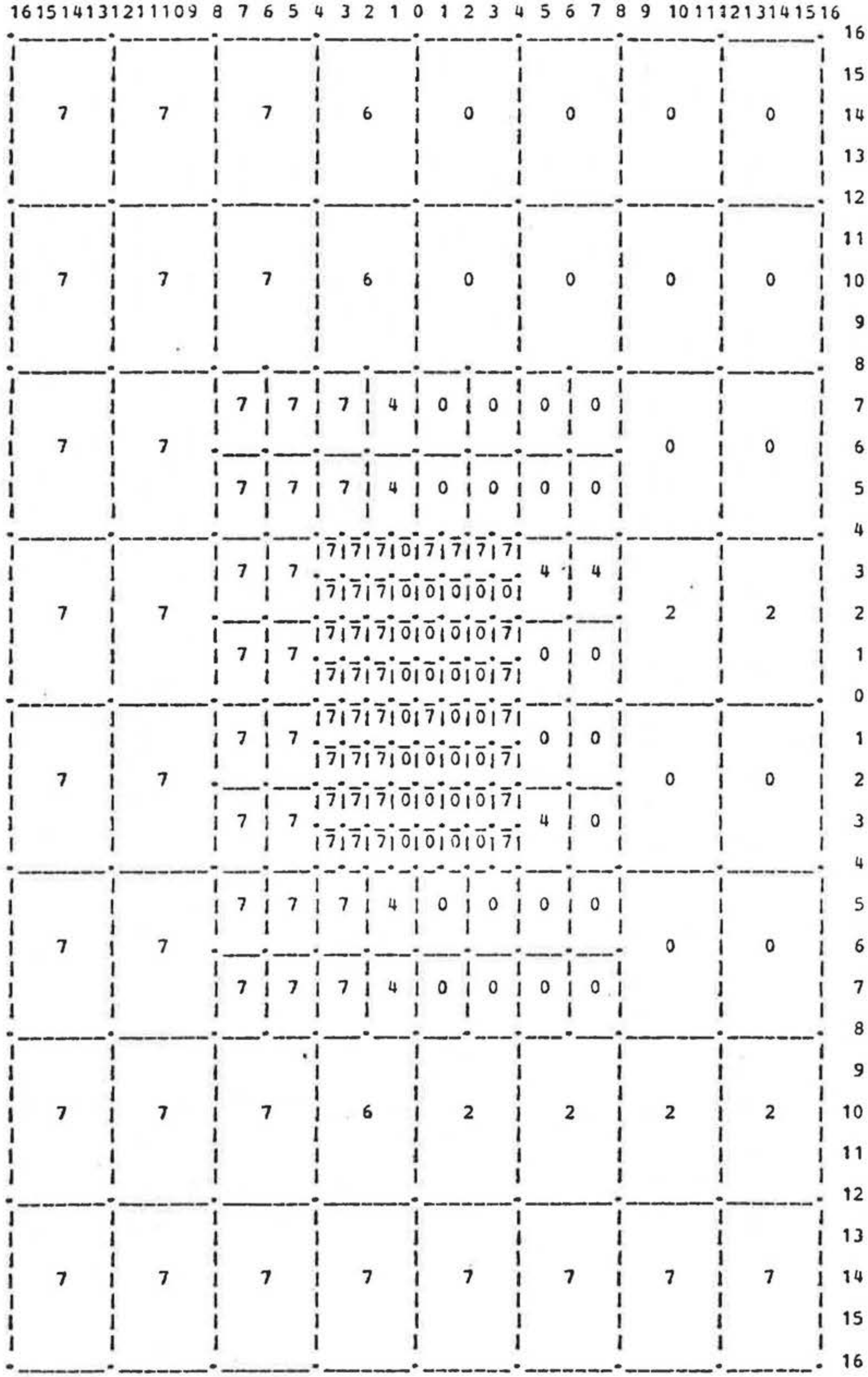
Figure III.15 shows the retinal and tactile impressions received by Utak immediately prior to the first few actions in the performance of section III.1.3.

III.2.4 Examples of tasks for Utak

Here I present a list of tasks, in English, which I would expect a competent organism-controller for Utak to be able to handle. Their method of presentation to Utak's organism-controller will be described in section IV.2.

"Go to the northeast corner"
"Go to the next room"
"Go to the square"
"Go round the square and return"
"Push the square into the northeast corner"
"Push the square into the next room"
"Push the brick through the door"
"Push the brick around the corner"
"Push the L-shaped object into the next room"

FIGURE III.15



```
. . .
. * .
. . .

* (ROLL 5.0 NE)
>     (4.24164 0.785398)
* (THLOOK)
```

```
161514131211109 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 10111213141516
.-------.-------.-------.-------.-------.-------.-------.-------.  16
|       |       |       |       |       |       |       |       |  15
|   7   |   7   |   7   |   5   |   4   |   4   |   4   |   4   |  14
|       |       |       |       |       |       |       |       |  13
.-------.-------.-------.-------.-------.-------.-------.-------.  12
|       |       |       |       |       |       |       |       |  11
|   7   |   7   |   7   |   2   |   0   |   0   |   0   |   0   |  10
|       |       |       |       |       |       |       |       |   9
.-------.-------.---.---.---.---.---.---.---.---.-------.-------.   8
|       |       | 7 | 7 | 4 | 0 | 0 | 0 | 0 | 0 |       |       |   7
|   7   |   7   .---.---.---.---.---.---.---.---.   0   |   0   |   6
|       |       | 7 | 7 | 4 | 0 | 0 | 0 | 0 | 0 |       |       |   5
.-------.-------.---.---.-.-.-.-.-.-.-.-.---.---.-------.-------.   4
|       |       |   |   |7|0|0|0|0|0|0|0|   |   |       |       |
|       |       | 7 | 7 .-.-.-.-.-.-.-.-. 0 | 0 |       |       |   3
|       |       |   |   |7|0|0|0|0|0|0|0|   |   |       |       |
|   7   |   7   .---.---.-.-.-.-.-.-.-.-.---.---.   2   |   2   |   2
|       |       |   |   |7|0|0|0|0|0|0|0|   |   |       |       |
|       |       | 7 | 7 .-.-.-.-.-.-.-.-. 4 | 4 |       |       |   1
|       |       |   |   |7|0|7|7|7|7|7|7|   |   |       |       |
.-------.-------.---.---.-.-.-.-.-.-.-.-.---.---.-------.-------.   0
|       |       |   |   |7|0|0|0|7|0|0|0|   |   |       |       |
|       |       | 7 | 7 .-.-.-.-.-.-.-.-. 0 | 0 |       |       |   1
|       |       |   |   |7|0|0|0|0|7|0|0|   |   |       |       |
|   7   |   7   .---.---.-.-.-.-.-.-.-.-.---.---.   0   |   0   |   2
|       |       |   |   |7|0|0|0|0|7|0|0|   |   |       |       |
|       |       | 7 | 7 .-.-.-.-.-.-.-.-. 0 | 0 |       |       |   3
|       |       |   |   |7|0|0|0|0|7|0|0|   |   |       |       |
.-------.-------.---.---.-.-.-.-.-.-.-.-.---.---.-------.-------.   4
|       |       | 7 | 7 | 4 | 0 | 4 | 0 | 0 | 0 |       |       |   5
|   7   |   7   .---.---.---.---.---.---.---.---.   0   |   0   |   6
|       |       | 7 | 7 | 4 | 0 | 2 | 4 | 0 | 0 |       |       |   7
.-------.-------.---.---.---.---.---.---.---.---.-------.-------.   8
|       |       |       |       |       |       |       |       |   9
|   7   |   7   |   7   |   2   |   0   |   0   |   0   |   0   |  10
|       |       |       |       |       |       |       |       |  11
.-------.-------.-------.-------.-------.-------.-------.-------.  12
|       |       |       |       |       |       |       |       |  13
|   7   |   7   |   7   |   5   |   4   |   4   |   4   |   4   |  14
|       |       |       |       |       |       |       |       |  15
.-------.-------.-------.-------.-------.-------.-------.-------.  16


B B B
- * -
- - A

>    NIL
* (ROLL 1.0 E)
>    (1. 1.570796)
* (THLOOK)
```

When a compass direction appears in the task statement this refers to Utak's own local orientation system, which need not coincide with the TABLETOP orientation. It is initialized when the first retinal impression is received. Whatever direction he facing at that time becomes north in his orientation system. I do not claim to have a system or even the outlines of a system that can handle all these tasks; I am presenting this list here to show the ultimate design goals for a robot-controller for Utak.

III.3 An extension and two generalizations of TABLETOP

The purpose of this section is to discuss issues that arise from the design of TABLETOP. These are not germane to the main argument of my thesis but are of some interest in their own right. They are also relevant to questions in automatic assembly.

The first issue concerns exactness. How accurate is the TABLETOP simulation, and how, if at all, could it be extended to achieve exactness? For the moment I will assume that the referent for the terms "accuracy" and "exactness" is the usual Cartesian representation of shapes, using real numbers represented to a limited precision in some computer. The motion of an object may, in the worst case behaviour of TABLETOP, be halted if a point of the moving object comes within root2 of a potential obstruction, whether or not a collision would occur in the Cartesian representation. This can occur if an edge of the obstacle intersects a TABLE square arbitrarily close to one corner and if the path of Utak, or a point of a held object, intersects the same TABLE square but arbitrarily close to the diagonally opposite corner. The TABLETOP collision point may be arbitrarily far from the Cartesian collision point, or even worse, the TABLETOP collision point may not correspond to a Cartesian collision point. These cases can arise if the line of approach of Utak, or an object, makes a near-zero angle with a potential obstructing edge.

How can this state of affairs be remedied? In describing the possible cures I restrict attention to the case of a moving point (Utak) approaching an object.

The first cure is this. In the course of projecting objects onto the TABLE array, whenever a TABLE square is entered by an edge a pointer back to the traversing edge is stored at that square. If a square is entered by several edges a pointer is stored for each edge. Then each time the path of Utak encounters an obstructing square on the TABLE, the Cartesian representation(s) of the edge(s) which caused this square to be marked as an obstruction is (are) retrieved. Then the Cartesian coordinates of the collision point, if any, are computed by any standard line intersection algorithm.

The second cure is based on the idea of repeatedly projecting Cartesian representations onto the TABLE at higher and higher scales of magnification, each time re-determining the obstructing squares. The process halts when the collision point, if any, has been found to within the required accuracy. I call this the focus method. To explain it I need some terminology.

A C-representation (R,C) is a collection R of Cartesian representations of one or more distinct objects, using some coordinate system C. A coordinate system C is obtained by the application of a sequence of rotation, translation, and scaling operations to some fixed initial coordinate system. Let a window W on a C-representation (R,C) be a rectangle of any size or orientation. Say that a C-representation (R,C) has been tabled with respect to a given window W if

a) All parts of R that lie wholly outside W are removed and any line segment of R that intersects an edge of W is replaced by a line segment that terminates at that edge. In computer graphics terms, R is clipped.

b) The coordinate system C is transformed into a new system by

- rotating it into alignment with the edges of W,
- moving the origin to the centre of W,
- applying scale factors, one in each coordinate direction, so that the sides of W coincide with the sides of TABLE.

c) The C-representation (R,C) is projected onto TABLE.

Suppose the coordinate system C' is the result of applying to a coordinate system C a series of several rotation, translation, and scaling operations as in b) above. Such a series can always be reduced to one rotation, one translation, and two scalings. Let Q be a unit square in C'. Let Q' be the rectangle in C that is obtained by applying to Q, in reverse order, the inverses of the operations in the series. Then the resolution of the coordinate system C' is defined to be the maximum of the lengths of the sides of Q in the original system C.

I can now describe more precisely the focus method for finding as accurately as desired the collision point, if any, between Utak's intended path and an obstacle. Let R be the collection R0 of the original Cartesian representations of all the objects and the verge in the environment, let C be the original coordinate system C0, and let W coincide with W0, the sides of TABLE. Let ϵ be the resolution required.

CP1. Table the C-representation (R,C) with respect to the window W. Let (R',C') be the new C-representation.

CP2. Find all potential obstructing TABLE squares along Utak's intended path. If there are none then exit with the message "no collision found". Otherwise, if the resolution of C' is less than ϵ, then compute the point of collision of the intended path with the first obstructing square encountered. Find the coordinates of this point in the original coordinate system C0 and exit with these coordinates for the point of collision. Otherwise, continue.

CP3. Take the smallest rectangle W' that is aligned with the direction of motion of Utak and that contains all the potential obstructing squares found in step CP2. Let (R,C)=(R',C'), the window W=W', and go to step CP1.

A magnification always occurs at step CP1 if the new window W' is smaller than the previous window W. The rectangle W' will always be aligned with the coordinate axes except, possibly, the first time that CP3 is executed. Thus a rotation is executed in step CP1 at most once.

This rotation, permitted by allowing the window W' in CP3 not to be aligned with the axes, is necessary to handle one special type of case. Namely, the case where the potential obstructing squares form a diagonal across the TABLE array, as can happen if an edge and Utak's path both extend approximately corner-to-corner and both lie approximately parallel to each other. A window aligned with the axes would not, in this case, be smaller than the current window, and consequently no magnification could occur. As a result of this single rotation, the intended path of Utak is always parallel to one of the axes, say the vertical.

Suppose the TABLE grid has n unit squares along each side. Then the rectangle W' of CP3 will have horizontal width one in every execution of CP3 after the first, since all the squares intersected by a vertical line lie in a single column. The horizontal scale factor in CP1 will therefore be n in every execution of CP1 after the second. Since n is an integer greater than 1, a horizontal magnification occurs in every execution of CP1 after the second.

This may seem like a computationally expensive method to use. One reason I have sketched it out is because, if the C-representation consists only of convex shapes, the projection onto TABLE can be done very fast with some simple parallel hardware. If such hardware was available, this magnification method might become feasible. Another reason is that this same method, in simpler form, can be used to solve linear programming problems in any dimension. A final reason is that an extension of the focus method is used in SHAPE, the spatial planner in the organism-controller of Utak.

The parallel hardware required consists of one component for each TABLE square. Given a line L, its coordinates are broadcast to every component. Each component computes whether its corresponding TABLE square lies to the right of, to the left of, or is intersected by, the line L. Let this computation take one time unit. Then the projection of a convex shape S bounded by m lines takes m time units, one for each line, plus the time for one extra AND operation by each component. The AND operation is this. If a TABLE square lies to the right of, or is intersected by, every line of the shape S, then the corresponding component signals "inside S"; otherwise the component signals "outside S".

One other question naturally arises. Can the design of TABLETOP be generalized to three or higher dimensions? The answer is yes, provided the projection of an n-dimensional convex polytope onto a generalized TABLE array can be computed. The generalized TABLE consists of an n-dimensional array of unit hyper-cubes. The scan-table technique used in TABLETOP for projecting a convex polygon onto the TABLE does not easily generalize to n dimensions. To apply the scan-table technique it must be easy to compute which hyper-cubes a hyper-plane passes through. In three dimensions this is the problem of determining, for each face of a polyhedron, which unit cubes the face intersects. When the face is oblique to all the coordinate axes there is no simple algorithm corresponding to the line-tracing algorithm in two dimensions. Thus the scan-table technique does not seem to generalize. However, the method sketched above for computing in parallel the projection of a convex polygon onto TABLE generalizes easily. For each hyper-plane one simply computes in parallel, for each unit hyper-cube of the n-dimensional array, whether the hyper-cube lies to the left of, to the right of, or is intersected by, the hyper-plane. One final AND operation for each hyper-cube completes the computation.

This method can also be used to compute the hypercubes swept out by a leading hyper-plane in the course of a translation, or to compute the segment of a hyper-sphere (generalization of a doughnut slice) swept out by (a part of) a hyper-plane in the course of a rotation. In this latter case one has to compute whether a hyper-cube is inside, outside, or intersected by the surface of a hyper-sphere. Thus the basic TABLETOP method can be generalized to higher dimensions.

\+ \+ \+ \+ \+

In this chapter I have described the design of the TABLETOP simulation system, and gone into sufficient detail that the essential problems to be handled in an implementation are clear. I have also shown how this design could be extended to obtain more accurate collision points, and how the two dimensional tabletop could be generalized to three or more dimensions. In particular I introduced the focus method for obtaining more accurate collision points. This will reappear later in the design of SHAPE, the spatial planner.

In conclusion, considerable programming effort is required to simulate an environment as simple as a tabletop. The advantages of such a system are that some of the problems associated with real world sloppiness are avoided, no additional hardware is required, and the sensory-motor experience of a prototype organism-controller is easily reproducible. In the next chapter I describe a class of algorithms fundamental to the functioning of the organism-controller.

CHAPTER IV

TOWARDS THE DESIGN OF A ROBOT-CONTROLLER

The purpose of this chapter is to present an approach to the design and implementation of a robot-controller for Utak. As described in the introductory chapter, any such design seems to require two parts, a data part called the world model or cognitive map and a process part called the action cycle. This latter consists of a loop containing the three subprocesses of perception, planning, and action.

The chapter is structured as follows. In the first section I present an analogy that is useful in approaching the problem. In the second section I present the parts required of any robot-controller and in the third I present a scenario of the behaviour which any complete robot-controller for Utak should display to be acceptable. In the fourth section I describe the progress made in one approach to the design and implementation of a robot-controller, and in the last section I describe an alternative approach to this problem.

IV.1 An analogy

I start with a thumbnail sketch of the well-known analogy between the scientific method and the process of perception since this was the starting point for my design of the organism-controller. Consider an experimenter investigating some phenomenon. He or she wants to understand the phenomenon and proposes an hypothesis. To test its validity this hypothesis is used to predict what will be observed if certain actions are done - an experiment. He or she carries out the experiment and makes observations. If the observations are almost exactly as predicted then the experimenter's degree of confidence or belief in the hypothesis increases. One then says that the hypothesis <u>explains</u> the observations. If the observations are not as predicted but the hypothesis can easily be modified to accommodate the observations, e.g. by changing a parameter, then the degree of confidence remains as before but in an improved hypothesis. If the observations are not as predicted and cannot be accommodated by simple parameter adjustment then the experimenter makes a structural change, if possible, to enable the hypothesis to accommodate them. If there is still an observation which the experimenter cannot currently explain by an hypothesis then it is noted but otherwise ignored. With this new hypothesis, or old hypothesis with higher confidence, the experimenter designs another experiment and makes more observations, accommodates the hypothesis to these, and so on. Eventually the hypothesis will, in principle, be so well adjusted to the observations that the hypothesis will be generally accepted as a useful description of one aspect of Nature. The hypothesis will be consistent with all the observations so far, or in other words may be regarded as the truth at that particular time and place.

Note the pragmatic nature of science: whatever theory is most satisfactory at a particular time is used as a basis for action. For instance, Newtonian mechanics was an acceptable and immensely successful theory even though it was known that it could not explain the precession of the perihelion of Mercury.

Now return to Utak in his simulated world. At all times Utak maintains an hypothesis about the world, called the <u>world model</u>. Each part of the world model has an associated degree of confidence, and these degrees of confidence may vary from time to time. In general the degree of confidence associated with a particular part will increase with time; only the occurrence of some quite unusual event will cause it to decrease. The experimenter is Utak; the phenomenon to be explained is the external world; the observations, or pieces of evidence, are provided by the series of sensory impressions impinging on Utak; and any hypothesis or world model must be consistent, as far as possible, with the series of sensory impressions. At the very least, the current world model must be consistent with the most recent sensory impression.

In this analogy, <u>perception</u> is the act of accommodating the world model to explain the current sensory impression, while maintaining consistency as far as possible with previous sensory impressions. When there is no externally imposed task, <u>planning</u> is deciding on an experiment to gather more evidence to corroborate the current world model. <u>Acting</u> is simply carrying out the planned actions. Each act will be executed with a speed dependent on the degrees of confidence associated with the parts of the current world model most relevant to this particular act. If all parts of the hypothesis have been corroborated to a high degree of confidence then it may seem to Utak that he knows the whole world, even though in actual fact there may be considerable difference between his hypothesis and the current state of TABLETOP. In other words he may still be mistaken even when he seems to know otherwise.

If there is an externally imposed task which Utak must perform then provided the current hypothesis has some overall minimal degree of belief, Utak will construct a plan to accomplish the task on the basis of this hypothesis. Any sensory impression received while carrying out this plan must be accommodated by the hypothesis. This continual cycle of perception, planning, and acting is, of course, the action cycle. When Utak is given a task he first assumes a default world model, then opens his eye and receives the first retinal impression. He interprets this first sensory impression and modifies the default world model to accommodate it. Then he interprets the task statement in terms of this world model, and passes control to the planner which makes an initial plan to accomplish the task. An act is decided on by examining the first few actions of the plan, and executed. Then another retinal impression is received, interpreted and accommodated by the world model, the plan modified if necessary, and the next act produced and executed.

This analogy with science constrasts with a naive Map-in-the-Head design. In this, spatial knowledge resides in a structure isomorphic to a printed map and spatial reasoning occurs when the "mind's eye" examines this structure. Such a proposal begs answers to many questions, the most important of which is, perhaps: Who draws the map? This is answered by the analogy with the scientific method: By a process of hypothesis assumption and modification, using partial evidence presented through the senses.

IV.2 The parts of an organism-controller

Any complete organism-controller for Utak must contain at least the following program steps. These can be stated here without specifying data structures or processes. All that is needed is a world model, a way to receive a retinal impression, and an action effector.

INITIALIZATION STEPS

1. Set the current world model equal to some default world model.
2. Receive the first retinal impression.
3. Analyze the retinal impression into regions and borders.
4. Interpret the regions in the retinal impression and identify the image of Utak in the retinal impression.

5. Modify the default world model to be consistent with the interpreted retinal impression.

6. Accept a task and interpret it in terms of the world model. This may require substantial modification of the world model, for instance the addition of an object if one is mentioned in the task but no object is "visible" in the current retinal impression.

PLAN

7. Construct a plan to achieve the task, using the spatial planner.

THE ACTION CYCLE

ACT

8. Test whether the task is complete. If so, STOP.

9. Decide on the next action to take, by examining the initial portions of the plan and the degrees of confidence associated with those parts of the world model close to the planned actions.

10. Execute the next action and receive the next retinal impression.

PERCEIVE

11. Interpret the new retinal impression on the basis of the current world model, and modify the world model as necessary to make it consistent with the current retinal impression.

PLAN

12. Is the plan still viable? If so go to 8.

13. Otherwise, re-compute all or part of the plan, as in step 7, and go to 8.

The parts of the action cycle correlate with figure I.6 as follows.

Steps 8, 9, & 10 are carried out by ACT.

Step 11, "perceive", is carried out by ACCOM.

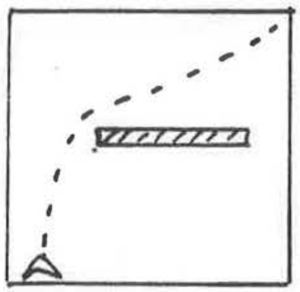
Steps 12 & 13 are carried out by SPLAN.

A task statement as required in step 6 is assumed to be presented as two parameterized world models, a starting and a goal world model. For example the world models corresponding to the task "Push the square object into the next room" will have two rooms, a square object, and a connecting doorway, the only difference between the start and goal world models being in the position of the square object. The problem in step 6 is to reconcile the currently assumed default world model with the world model implied by the task statement. I make this assumption to circumvent the handling of natural language input.

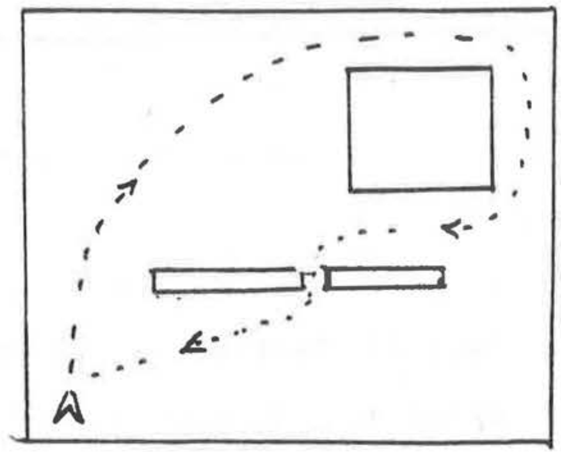
IV.3 The goal behaviour for an organism-controller

The intended meaning of some of the task statements of III.2.4 (page 134) is illustrated in figure IV.1. This shows the intended situations before and after each task but does not indicate how the world model of Utak might change while executing a task. A detailed scenario of the intended behaviour of Utak as he carries out one task appears in figure IV.2. It

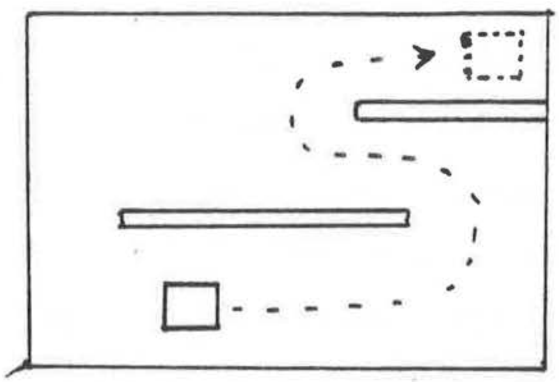
FIGURE IV.1 The meaning of task statements.



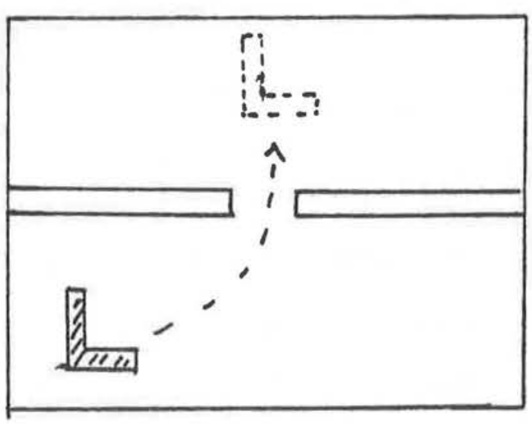
"GO TO NORTH EAST CORNER"



"GO ROUND SQUARE AND RETURN"



"PUSH SQUARE OBJECT TO NORTH EAST CORNER"



"PUSH L-SHAPED OBJECT INTO NEXT ROOM"

starts with a summary on the first page (a), showing how the many pages of this figure relate to executions of the action cycle.

In the initial TABLETOP situation S-1 (b), Utak is located in the area of the southwest corner, a square object lies in the northeast corner, and two thin horizontal obstacles separated by a small gap are in the east half of the tabletop. The actual tabletop or "room" dimensions are 40X40.

The default world model assumed by Utak before opening his eye is a square room of dimensions 36X36 centred on him (c). The first retinal impression is shown in (d) with the preferred line-segment interpretation superimposed on it. When this interpretation is reconciled with the default world model, WM-1 is obtained (e), containing a single small object object1. Then the task statement is received (f), which really consists of two subtasks. Since there already is a square object, object1, in the current world model WM-1, this is immediately identified as the square object of the task statement. Thus no modification to WM-1 is required by the task statement. If the statement had included, for instance, "go between the square and the L-shape" then an L-shaped object would have had to be hypothesized in a position beyond the area covered by the retina. A path is planned in the world model and a first action decided on (g). The size of the action, or equivalently the distance travelled between retinal impressions, is proportional to the confidence with which the structure of the world model is known. This

FIGURE IV.2(a) Executions of the action cycle.

(b) Initial situation S-1.
(c) Default world model WM-0
(d) First retinal impression RTI-1
(e) First world model WM-1
(f) Task statement
(g) First plan, PLAN-1, and first act ACT-1

(h) New situation S-2
(i) Next retinal impression RTI-2
(j) World model WM-2, plan PLAN-2, act ACT-2

(k) Situation S-3
(l) Retinal impression RTI-3
(m) World model WM-3, completely new plan PLAN-3, act ACT-3

(n) Situation S-4
(p) Retinal impression RTI-4
(q) World model WM-4, plan PLAN-4, act ACT-4

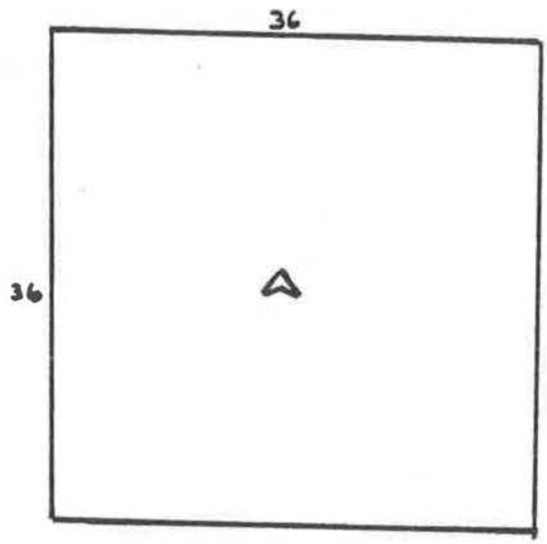
(r) Situation S-5
(s) Retinal impression RTI-5
(t) World model WM-5, plan PLAN-5, act ACT-5

(u) Situation S-6
(v) Retinal impression RTI-6
(w) World model WM-6, plan PLAN-6, act ACT-6

```
>    GO ROUND THE SQUARE OBJECT TO THE SE CORNER
>
* (TABLETOP)
* B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                               C C C C             B
* B                                                               C C C C             B
* B                                                               C C C C             B
* B                                                               C C C C             B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                               B B B B B B B B B B B B     B B B B B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                     *                                                             B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B                                                                                   B
* B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
```

FIGURE IV.2 (b) Situation S-1

FIGURE IV.2 (c) The default world model WM-0.



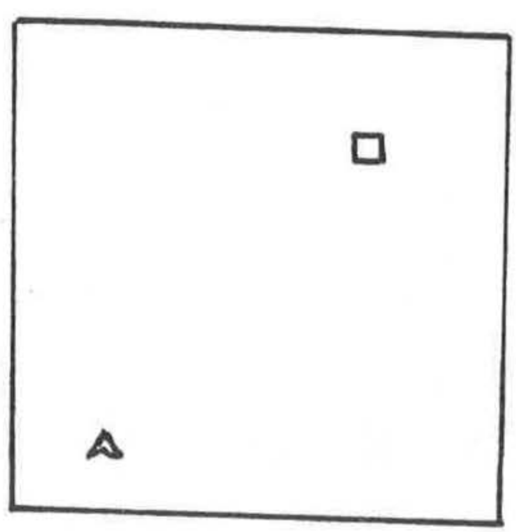
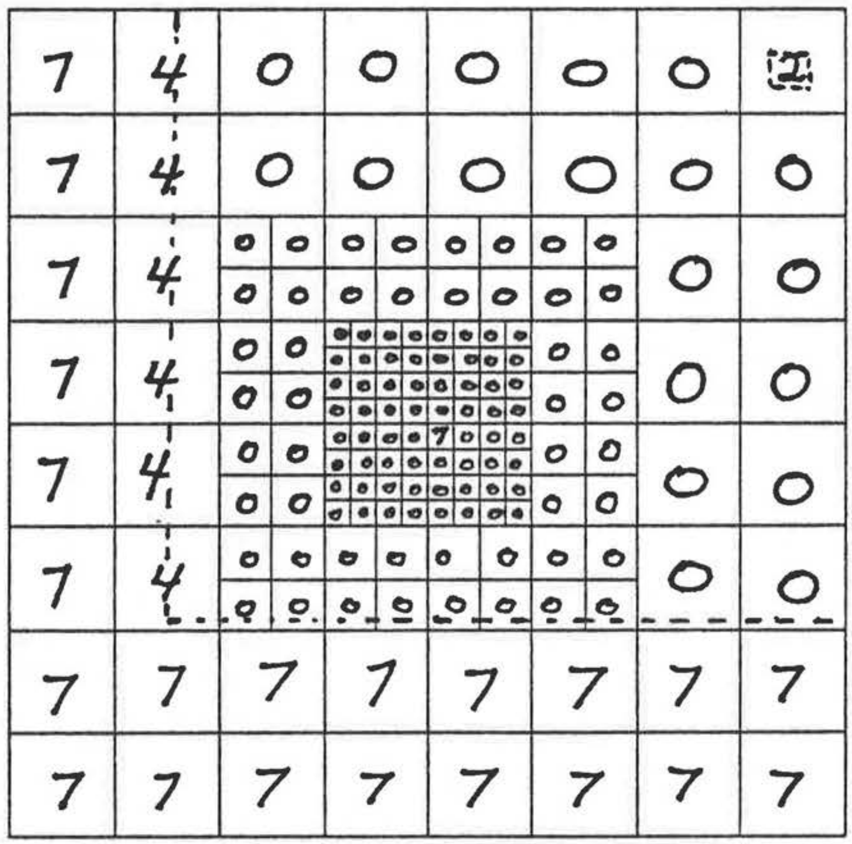


FIGURE IV.2 (e) The world model WM-1 after the first retinal impression RTI-1 received.

FIGURE IV.2 (d) Retinal impression RTI-1.



This shows the gray levels received by Utak's retina from the TABLETOP world. The central '7' is the image of Utak. The overlaid dashed lines show the line-segment interpretation.

FIGURE IV.2 (f) Task statement "Go round the square object to the south-east corner".

This may be translated as:

"Find a square object."
"Keeping the square object on your right, to to a point near the square object on the side away from your current position."
"Still keeping the square object on your right, go from this point to the south-east corner."

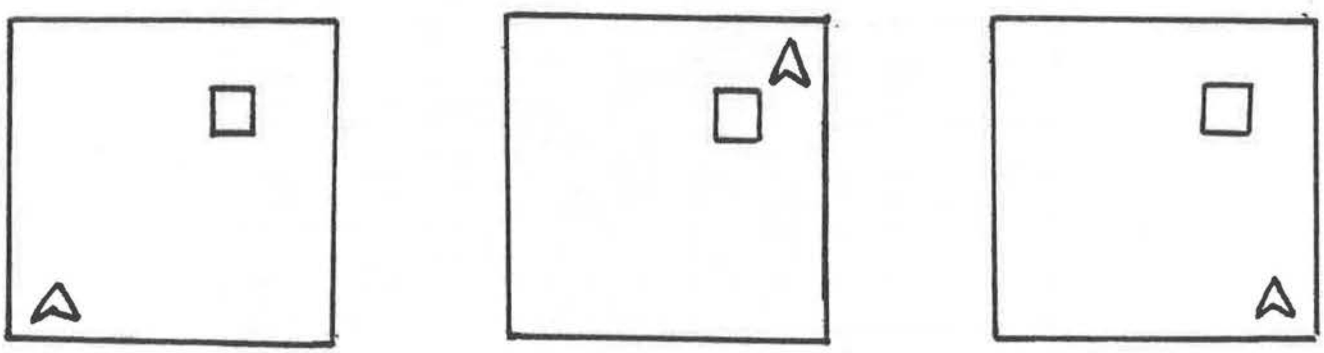
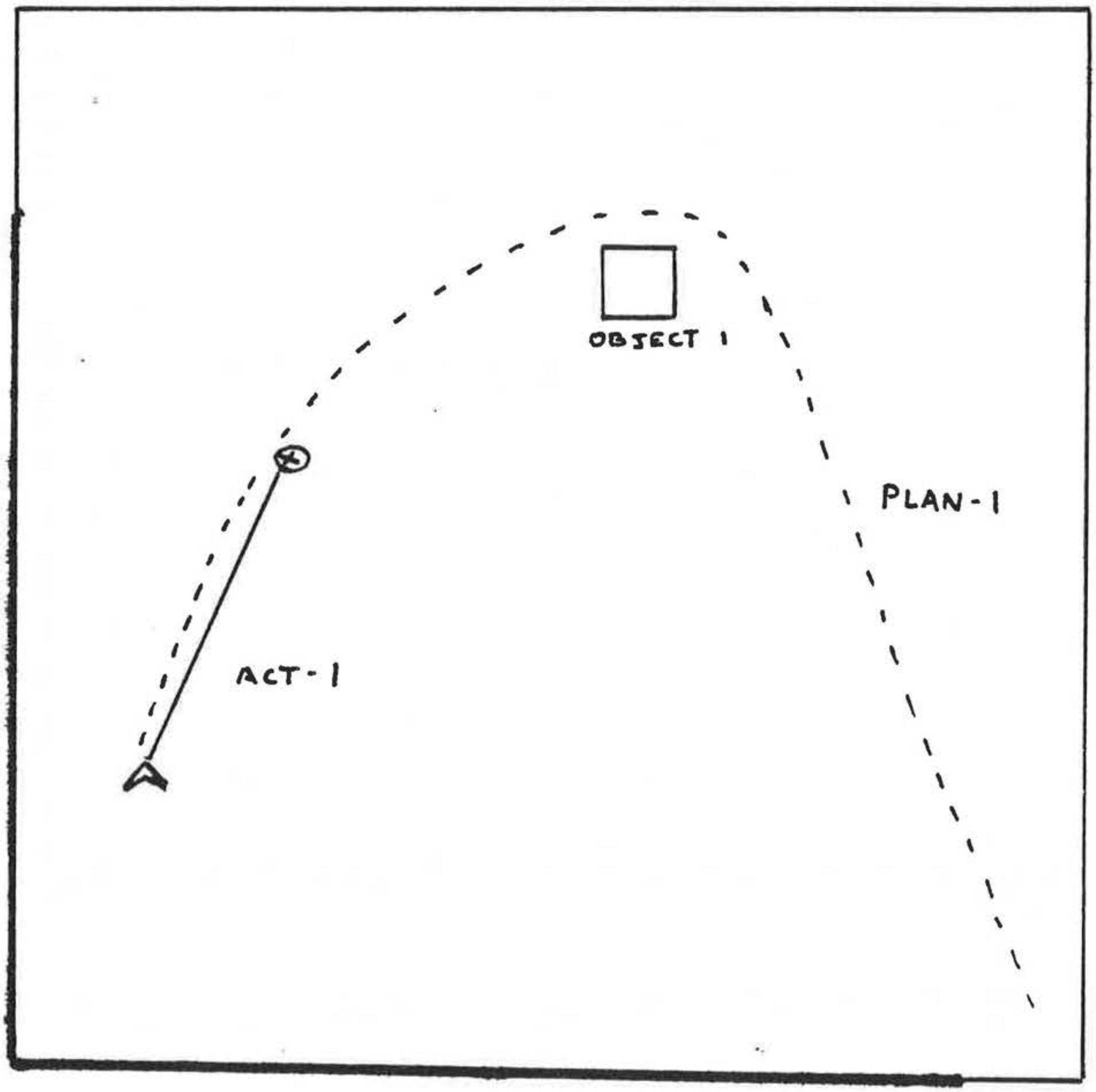


FIGURE IV.2 (g) PLAN-1, ACT-1, superimposed on WM-1.

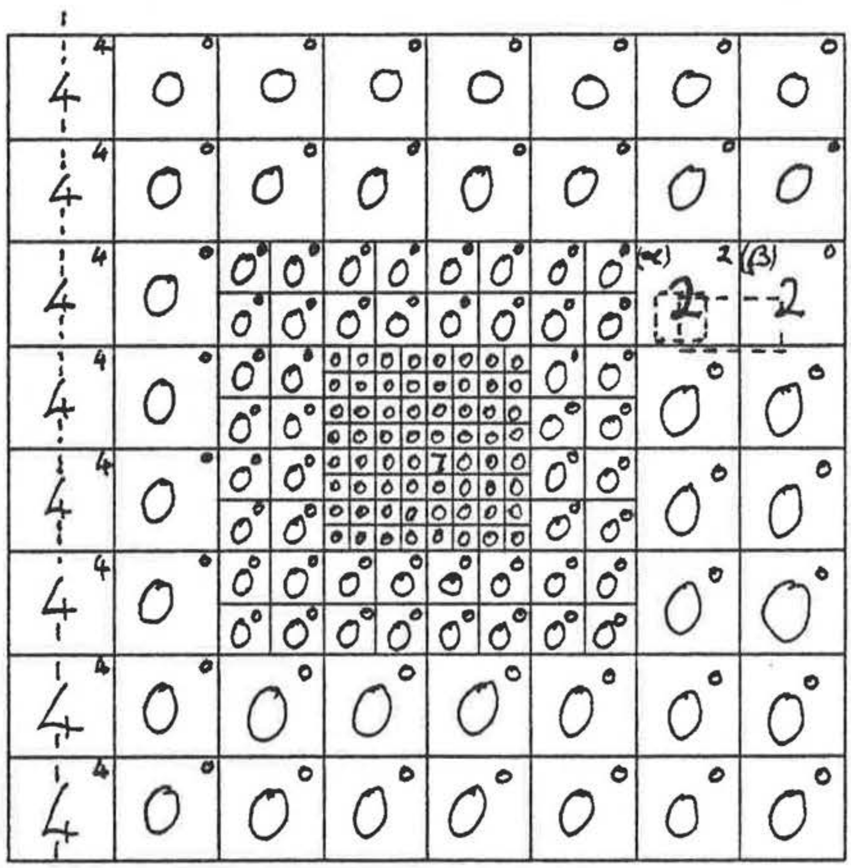


The thickened lines correspond to the line segment interpretation of RTI-1. The other segments forming the boundary of the tabletop are hypothesized.

```
* (ROLL 9.0 E) (TABLETOP) (THLOOK)
* B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                             C C C C             B
* B                                                             C C C C             B
* B                                                             C C C C             B
* B                                                             C C C C             B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                             B B B B B B B B B B B B     B B B B B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                             *                                                   B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
```

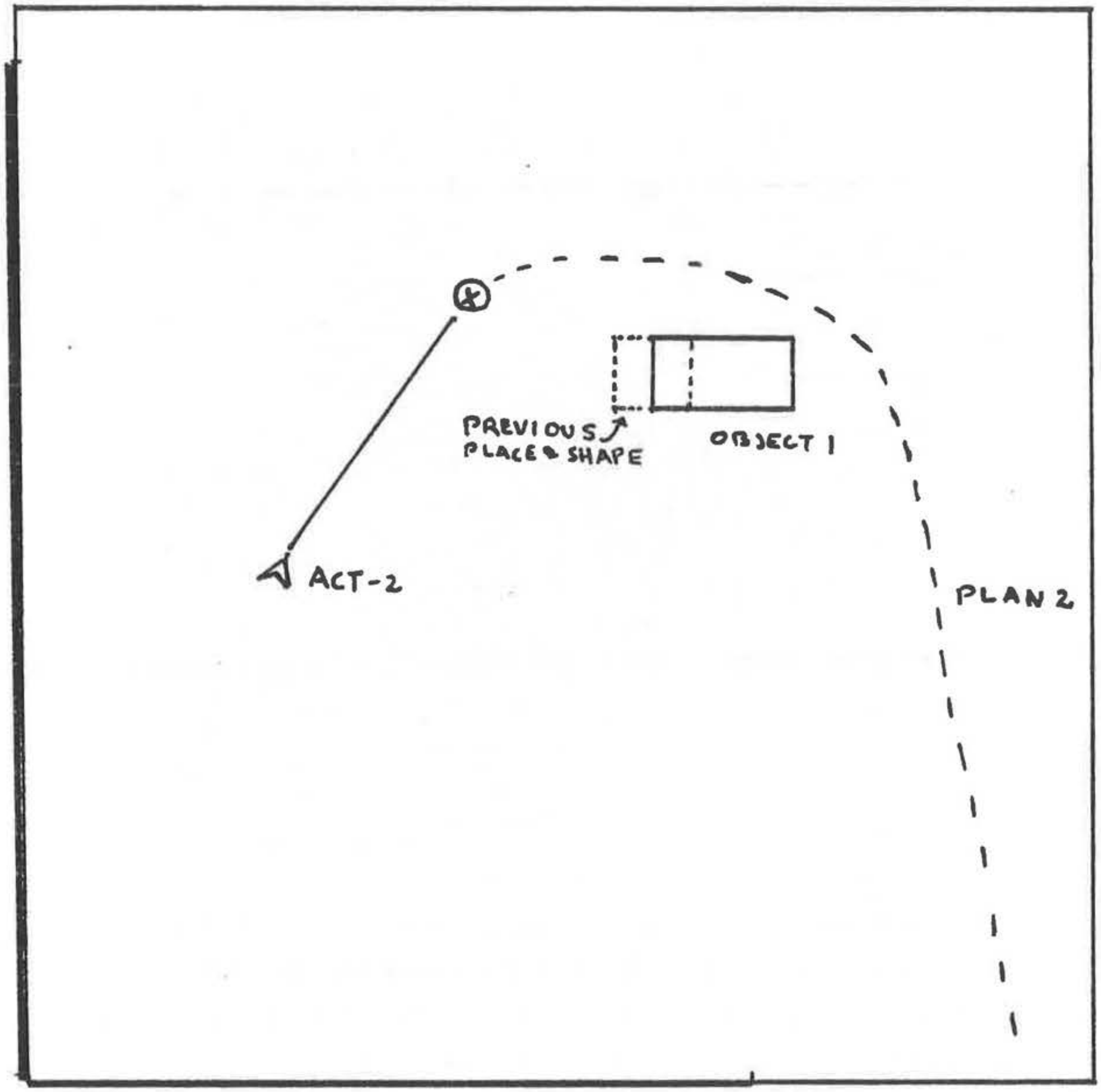
FIGURE IV.2(h) Situation S-2

FIGURE IV.2 (i) Retinal impression RTI-2



The predicted graylevels are inscribed in the top right hand corner of all but the foveal retinal fields. The predicted line segment interpretation is shown by the overlaid dashed lines at the left hand side and at retinal field (α). The only difference between predicted and actual graylevels occurs at (β); this results in a new line segment interpretation overlaying both fields (α) and (β).

FIGURE IV.2 (j) World model WM-2, with PLAN-2 and ACT-2.



The position and shape of object 1 have been updated - it is no longer a square object. The previous plan, PLAN-1, had to be modified slightly.

* (ROLL 8.0 NE) (TABLETOP) (TALJOK)

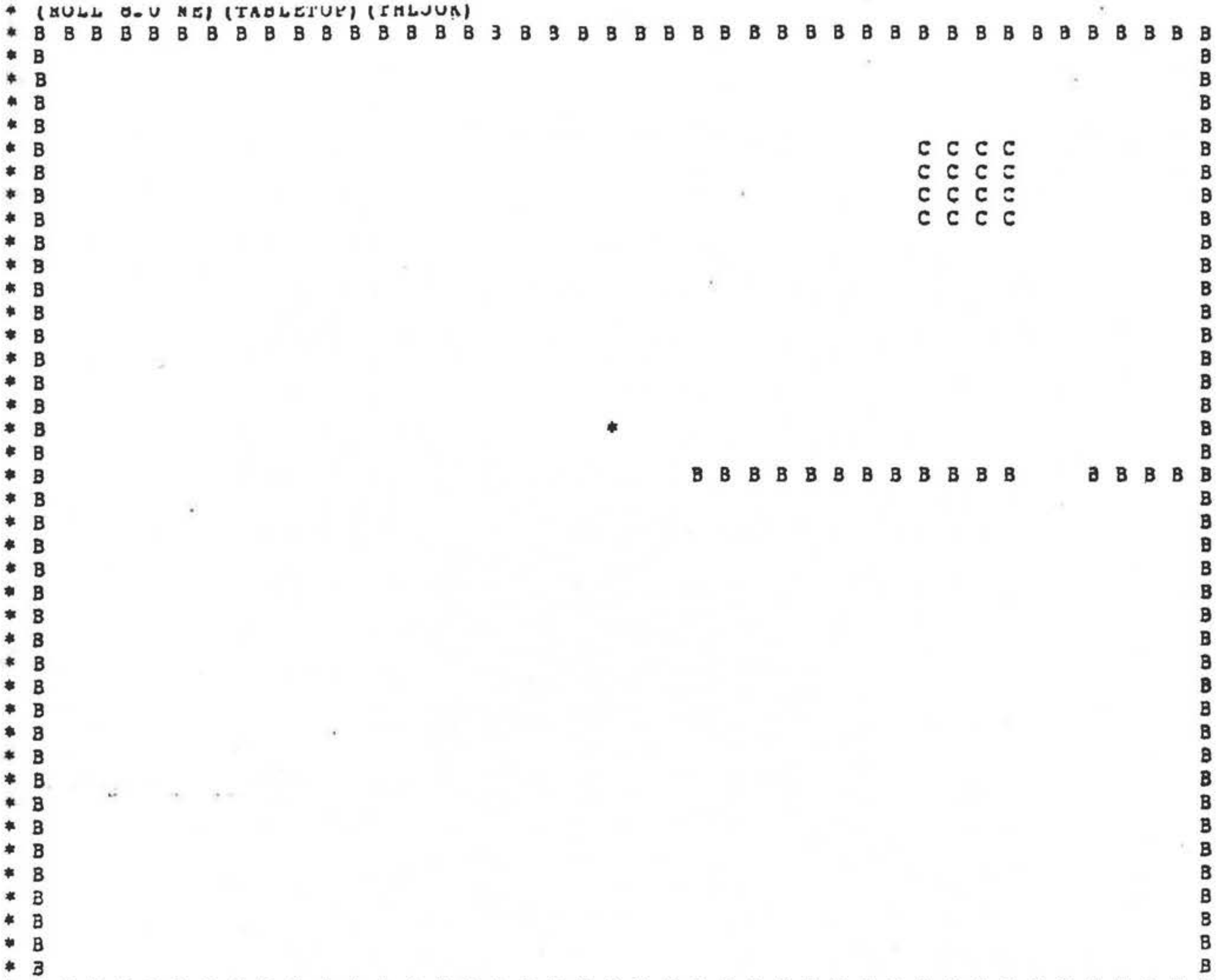
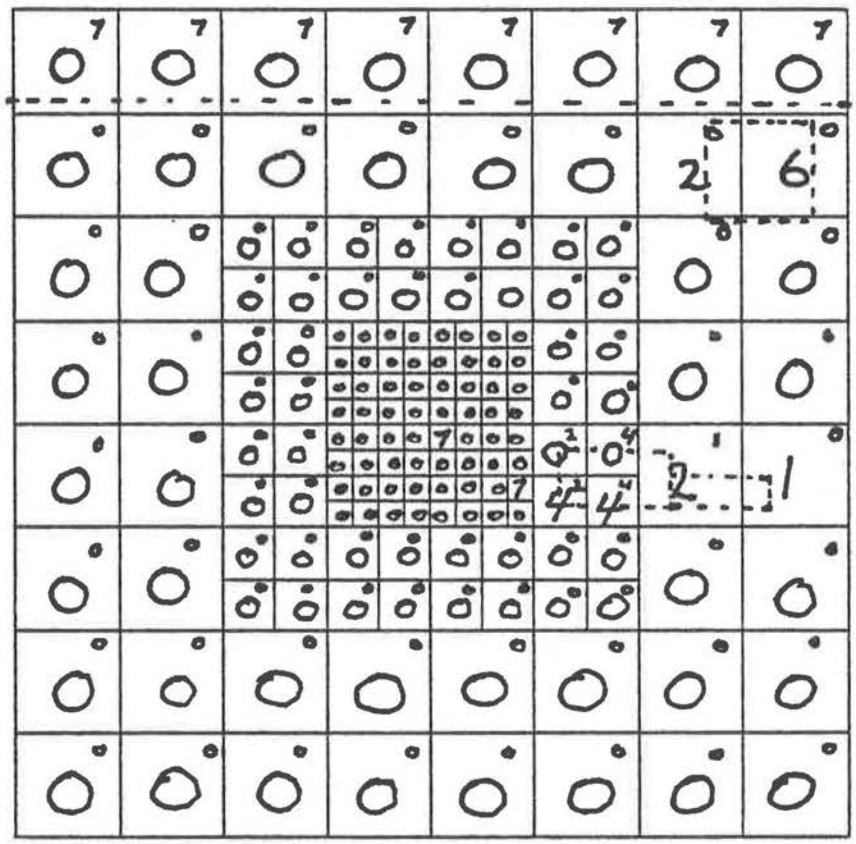


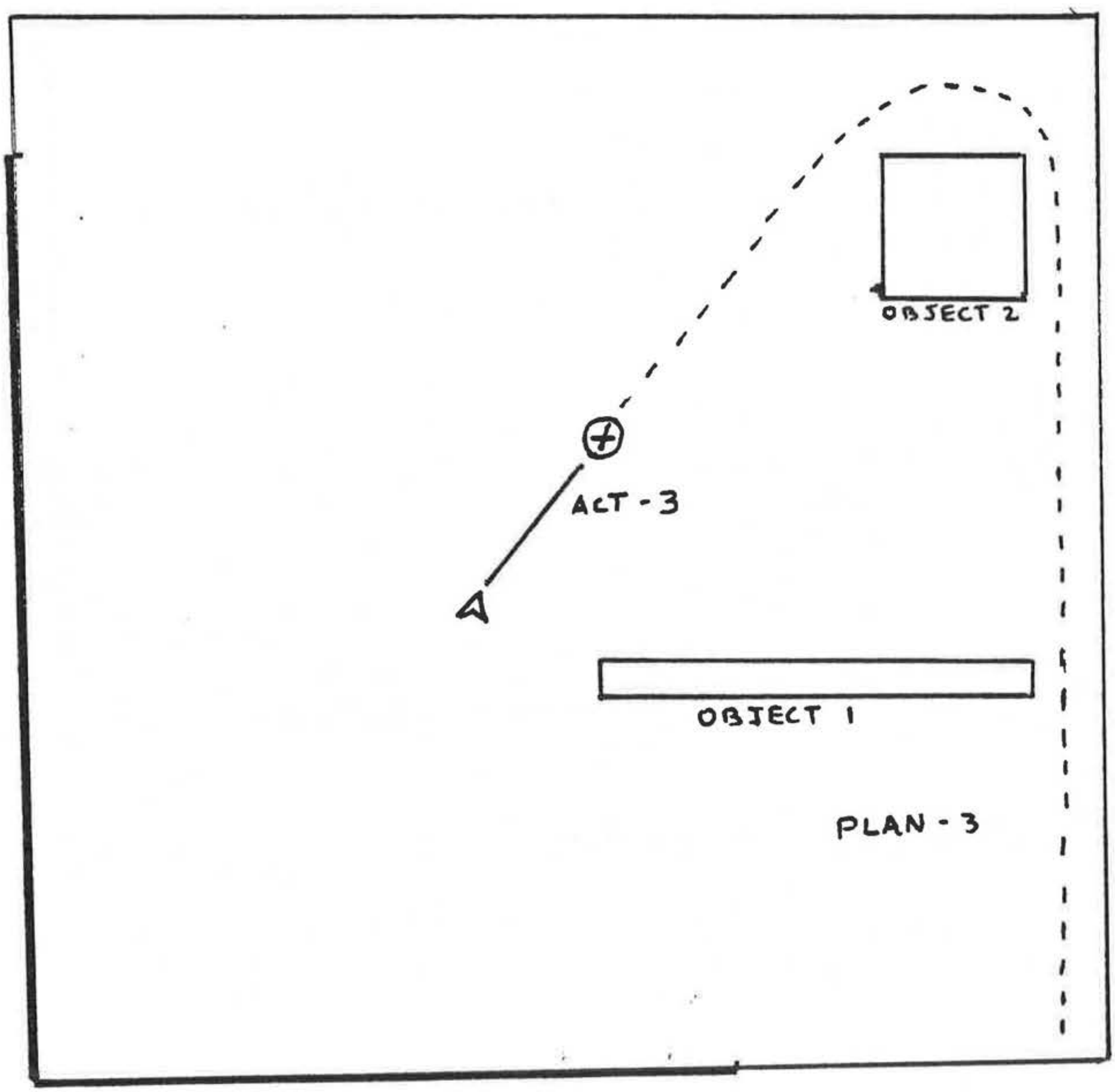
FIGURE IV.2 (k) Situation S-3

FIGURE. IV.2 (1) Retinal impression RTI-3.



The actual gray levels differ in several places from the predicted gray levels (in the RH corner of each field). The row of O's across the top forces the north verge out by four units. The 2 and 6 in the top RH corner forces the introduction of a new object. Finally, the several differences at middle right force another change in the shape of object 1.

FIGURE IV.2 (m) World model WM-3, with PLAN-3 and ACT-3.

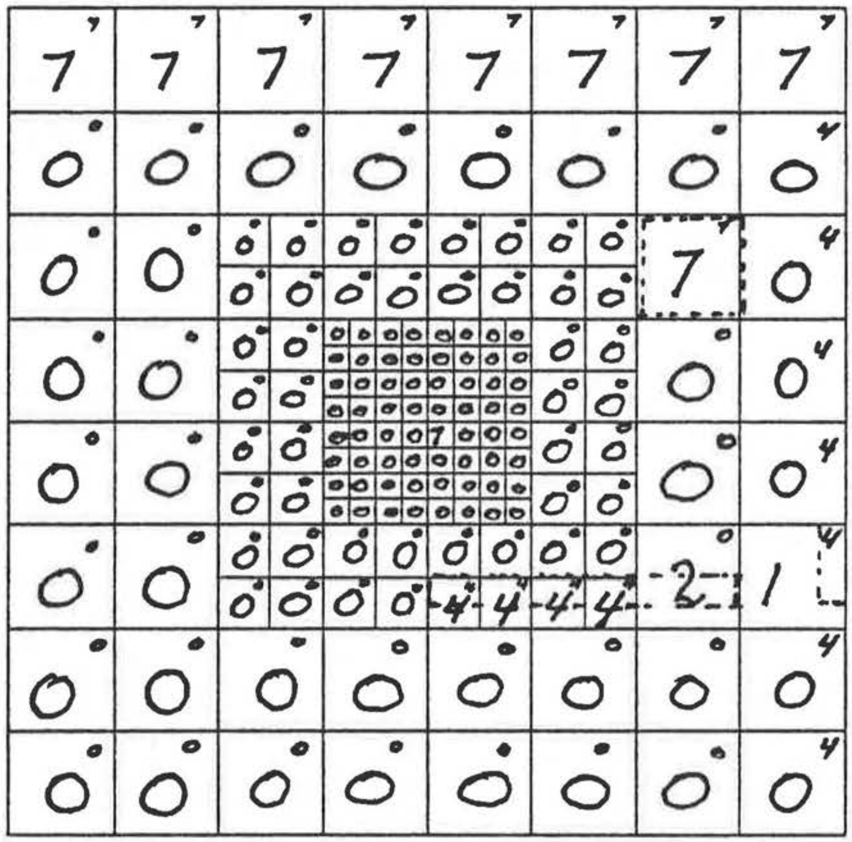


The north verge moved out by four units; object 2 is new and is square while the old object 1 can no longer be considered square; therefore, object 2 is assumed to be the square object of the task.

```
* (ROLL 3.0 E) (TABLETOP) (THLOOK)
>    (3. 1.570796)
* B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                           C C C C             B
* B                                                           C C C C             B
* B                                                           C C C C             B
* B                                                           C C C C             B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                           *                                   B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                           B B B B B B B B B B B B     B B B B B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B                                                                               B
* B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
```

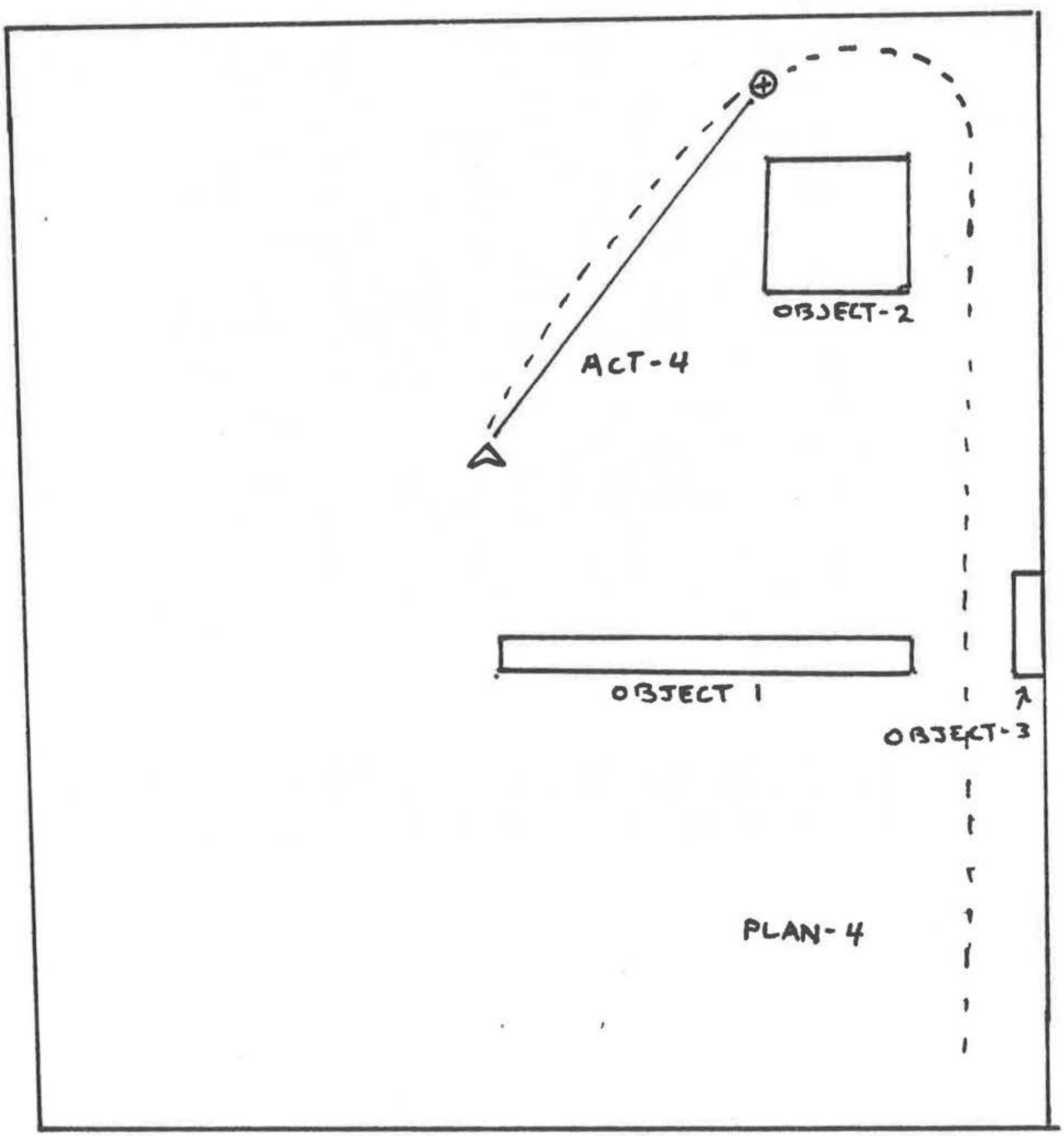
FIGURE IV.2 (n) Situation S- 4

FIGURE IV.2 (p) Retinal impression RTI-4.



The gray level differences on the right-hand side force the east verge to be moved out by two units, and a new object, object 3, to be introduced.

FIGURE IV.2 (q) World model WM-4, with PLAN-4 and ACT-4.

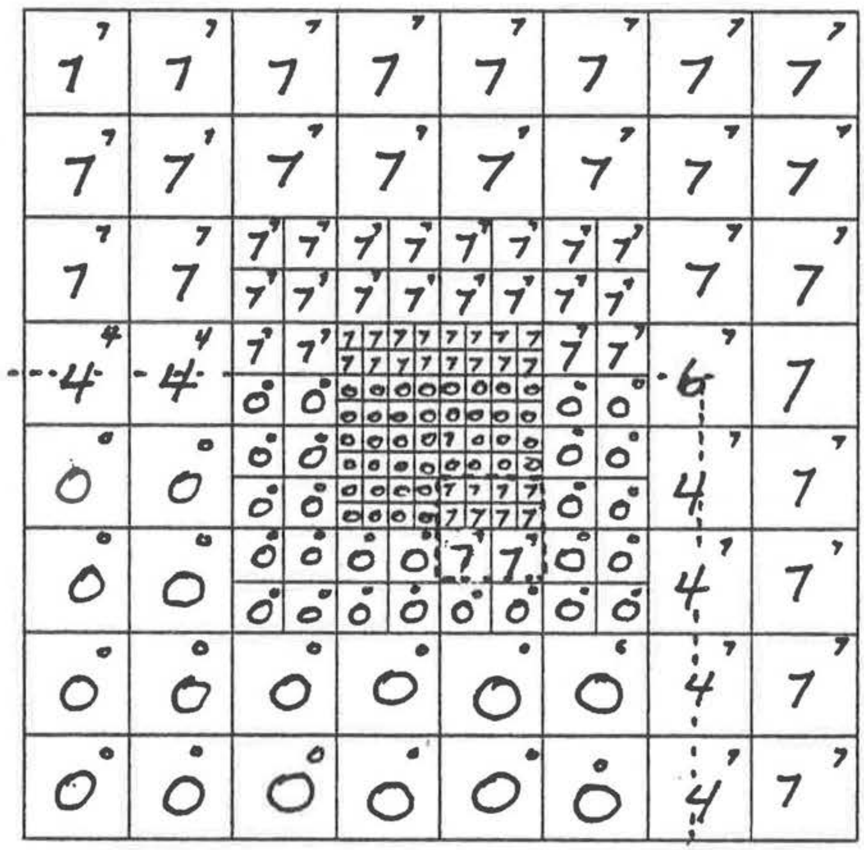


The east verge moved out by two units; new object 3 assumed.

```
* (ROLL 1.0 S)
>     (1. 3.141593)
* (TABLETOP)
* B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
* B                                                                                 B
* B                                                                                 B
* B                                                             *                   B
* B                                                                                 B
* B                                                             C C C C             B
* B                                                             C C C C             B
* B                                                             C C C C             B
* B                                                             C C C C             B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                             B B B B B B B B B B B B     B B B B B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B                                                                                 B
* B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
```

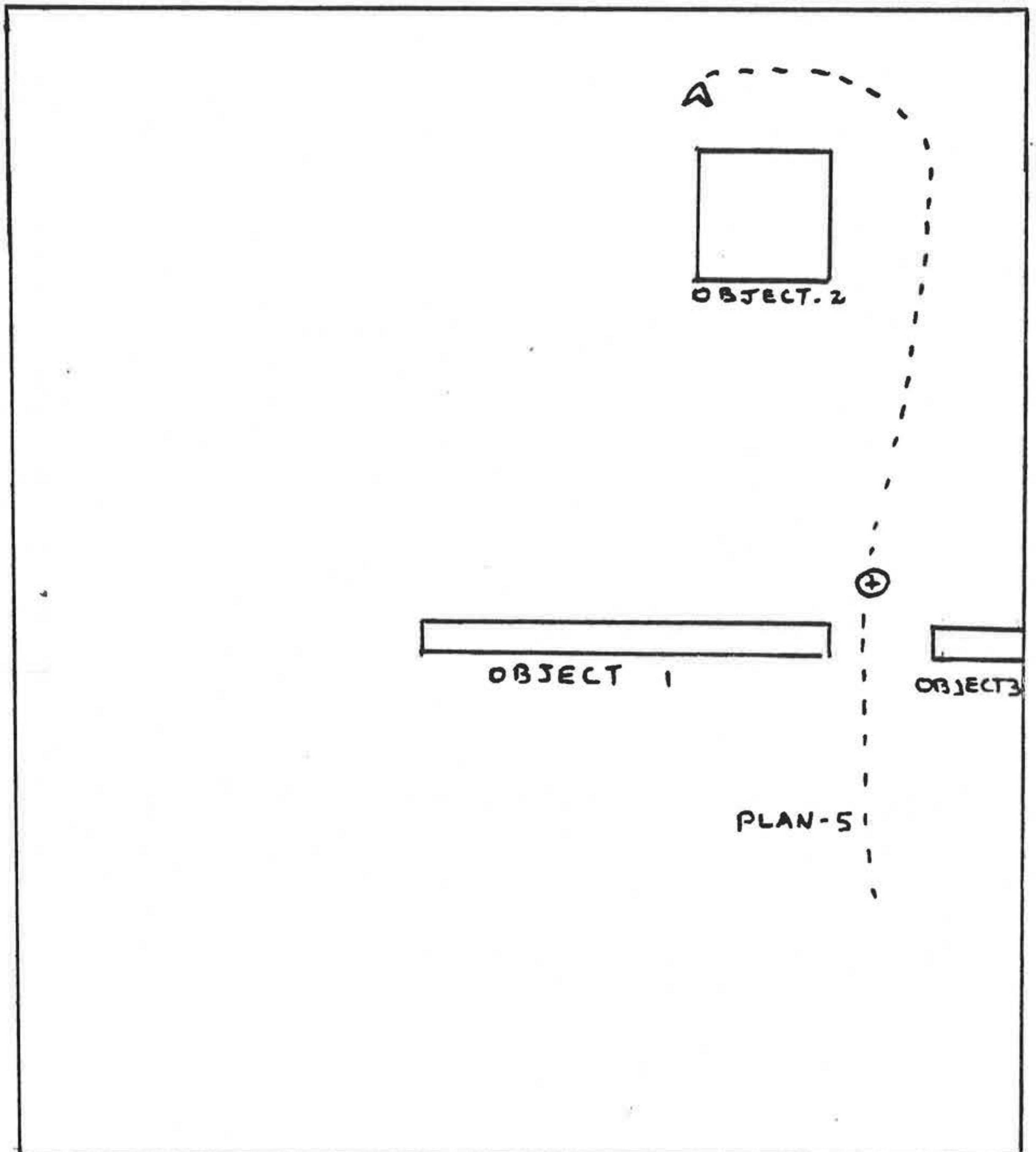
FIGURE IV.2 (f) Situation S- 5

FIGURE IV.2 (s) Retinal impression RTI-5.

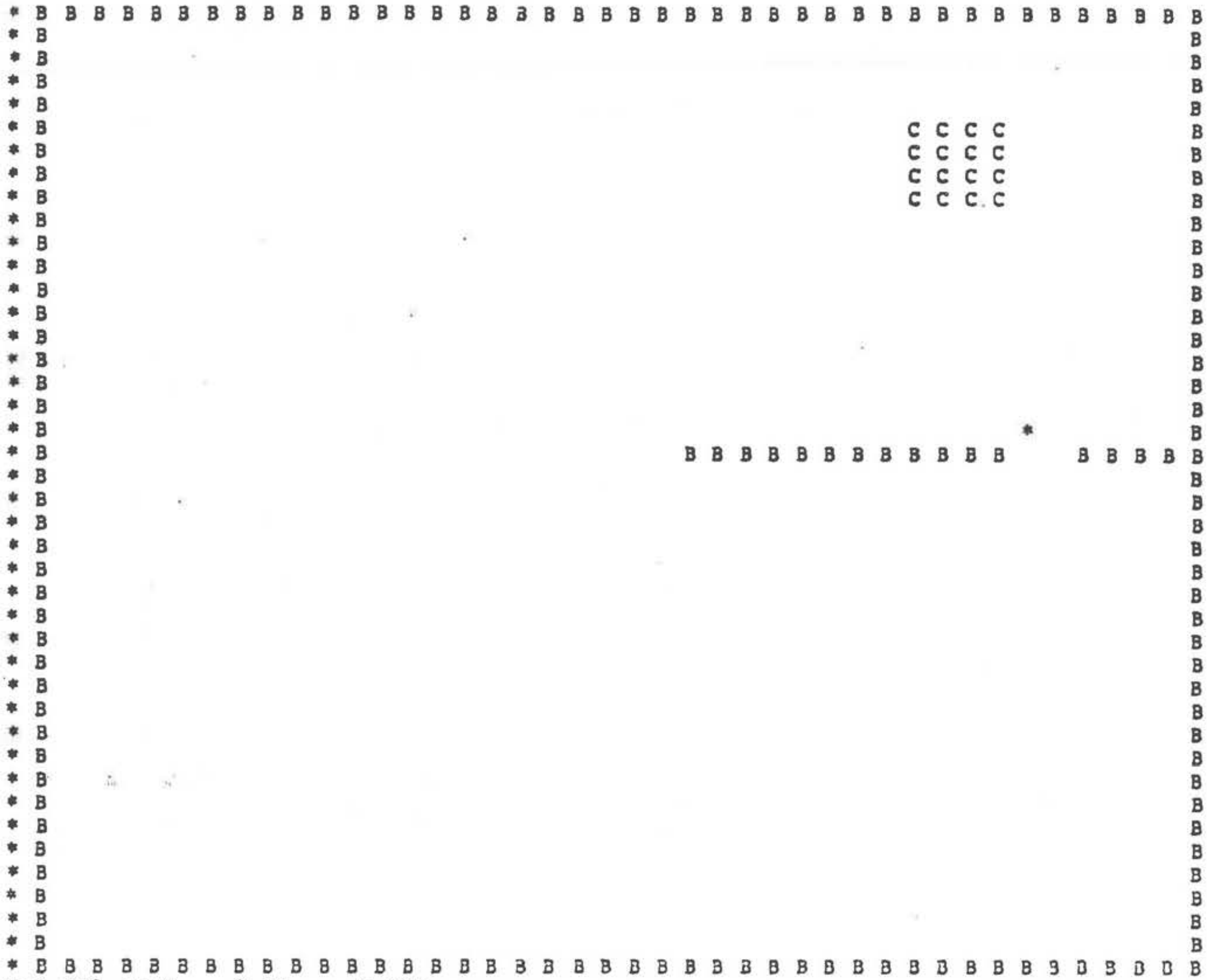


Once again, the gray level differences on the right-hand side force the east verge out by two units and a change in the position and shape of object 3. The gray levels for object 2 (just below centre) were exactly as predicted.

FIGURE IV.2 (t) World model WM-5, with PLAN-5



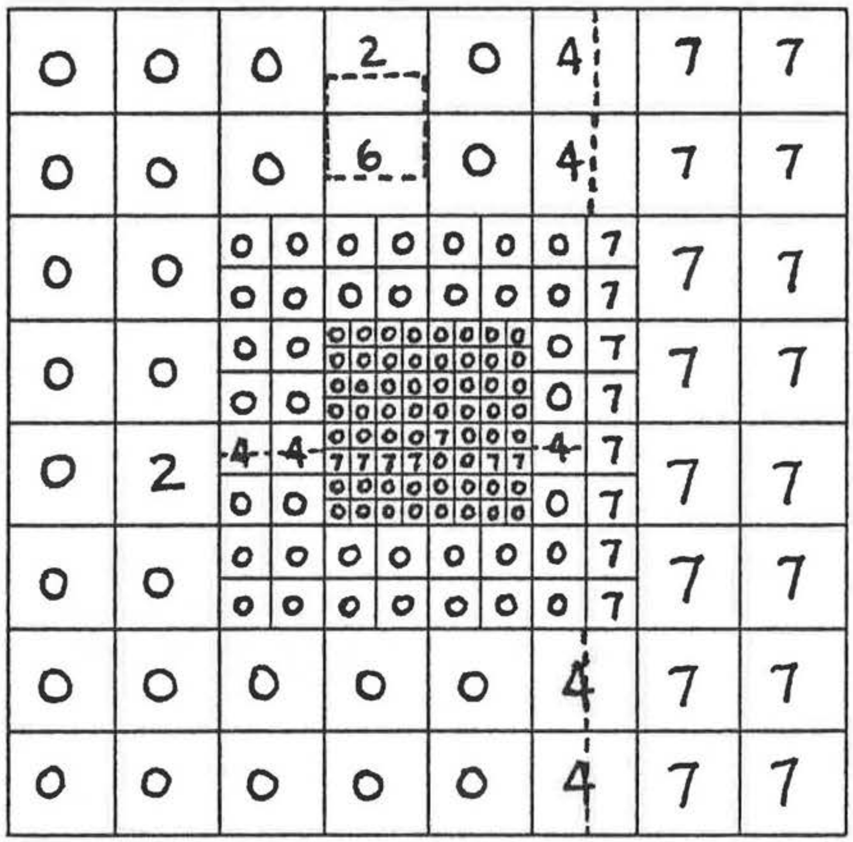
The east verge moved out by two units; shape of objects changed.



T=0.077 DR=0 $.20, $.28T

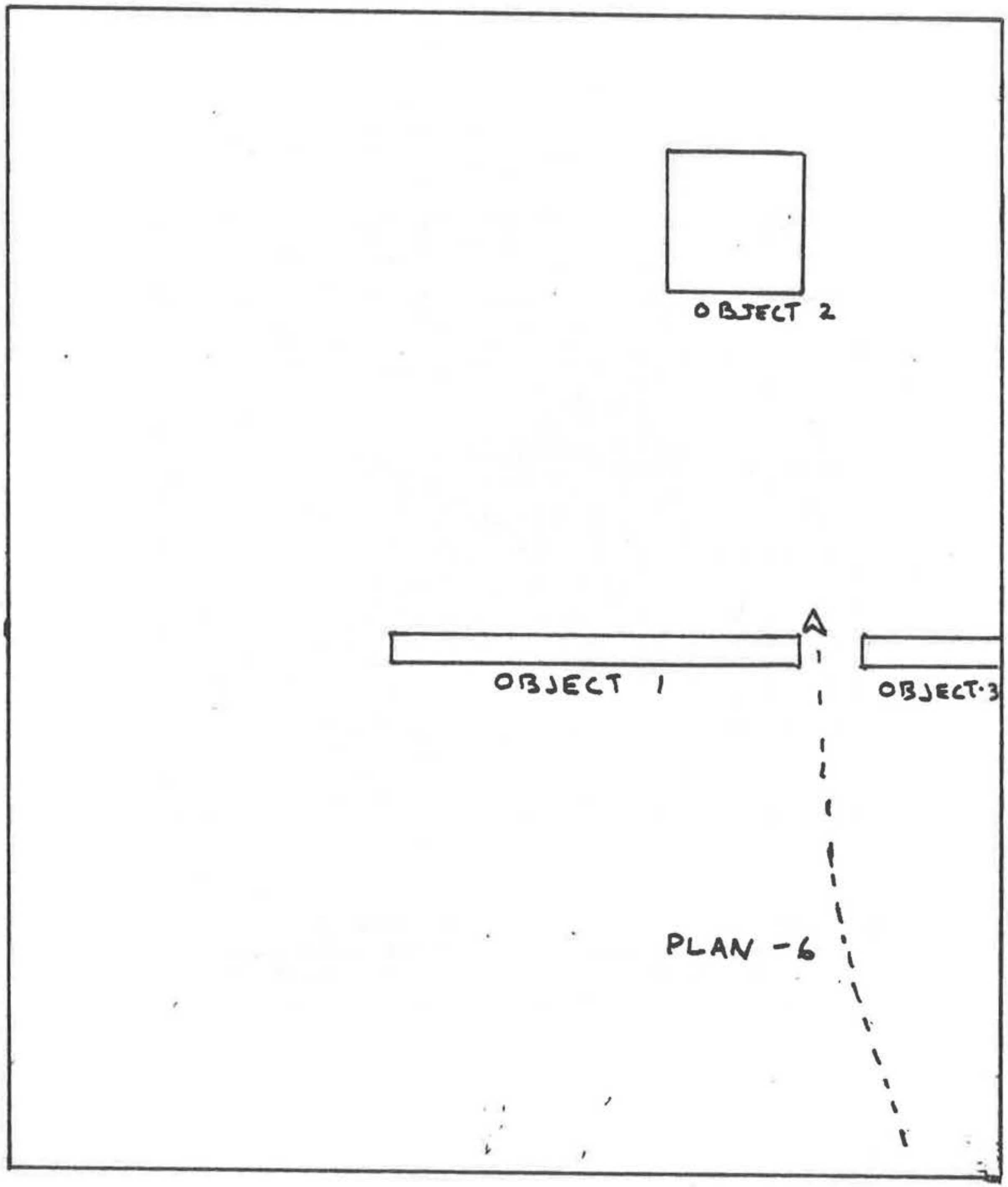
FIGURE IV.2 (u) Situation S-6

FIGURE IV.2 (v) Retinal impression RTI-6.



The predicted gray levels are not shown here because only one difference between predicted and actual occurs: in the Fovea at the left hand end of object 3. This forces a very small change in object 3.

FIGURE IV.2 (w) World model WM-6.



The shape of object 3 changed very slightly.

distance can also be regarded as a measure of speed with which Utak travels, if one makes the assumption that retinal impressions are received at a constant rate.

The TABLETOP situation S-2 after the execution of the first action is shown in (h) and the new retinal impression, RTI-2, received (i). The two neighbouring graylevel values of 2 in the upper right hand side are not as predicted by WM-1; consequently the position and shape of object1 must be changed to be consistent with RTI-2, while maintaining consistency with RTI-1. As a result of this change object1 is no longer square and there is some doubt as to whether it should still be identified with the square object of the task statement. There is, however, no other visible, potentially square, object with which the task's square object could be identifed, short of hypothesizing one in an arbitrary position beyond the area that is or has been covered by the retina. Thus the known object1 is retained, for the moment, as the task's square object. The new world model is WM-2 (j). The original plan remains unchanged and another action is decided on.

The new situation S-3 is shown in (k) and the next retinal impression, RTI-3, obtained (l). This differs considerably from the predicted retinal impression. New line-segment interpretations are derived and the world model WM-2 modified accordingly to obtain WM-3 (m). A new square object, object2, has been found and the shape of the supposed square object object1 turns out to be far from square. Thus the new object2 is assumed to be the square object of the task statement. Also, the position of the edge of the "room" is not as expected and has to be moved out by four units. The planner is called and a new plan from the current position of Utak going around object2 and back to the southeast corner is constructed. Since there is a gap between object1 and the side of the room, and since a path going via the gap should be shorter than going on the other side of object1, the new plan goes through this gap.

Another action is decided on and executed, resulting in situation S-4 (n) and retinal impression RTI-4 (p). The graylevels of zero along the right hand side where fours were predicted results in the east side of the room being moved out by two units. The graylevel of one is interpreted as a small object object3 against the east side. From the evidence presented in RTI-4 alone it would be consistent to assume a small thin object not quite extending to the edge of the retina; this interpretation is not consistent with the evidence from RTI-3 (which is embedded in the line-segment interpretation obtained from it), so instead object3 is assumed. The new world model is WM-4 (q); the plan remains unchanged, and an action decided on and executed.

The new situation is S-5 (r), with new retinal impression RTI-5 (s). Again this is not quite as predicted; the east side of the room has to be moved out by two and the position and shape of object3 changed. In the accommodated world model WM-5 (t), the plan remains the same. The next situation S-6 (u) and retinal impression RTI-6 are obtained after several intermediate actions. RTI-6 (v) shows clearly the gap between object1 and object3 which was only deduced from previous retinal impressions and the current world model WM-6 (w) correctly reflects situation S-6. Thus in this example no more accommodation can occur, so the remainder of the plan will be executed correctly.

IV.4 A first approach to implementation

The idea is to take seriously the analogy between science and perception. The world model consists of a collection of statements about the shape of the verge, the positions of objects on the tabletop, and the shapes of the objects. Each statement is to have an associated degree of confidence based on evidence collected from a series of retinal impressions. This degree of confidence is to be used in the accommodation process, to help determine to what extent old statements should be modified when new evidence is received. The statements give positions of end-points of lines and other spatial facts in terms of a Cartesian coordinate system centred on Utak. A spatial problem is to be solved by projecting all or part of the world model onto a screen -- Utak's map-in-the-head -- then solving the problem there and translating the solution -- the plan -- back into the Cartesian coordinate system. At all times there is a world model and a plan, even though initially these may be simple defaults.

The implementation of these ideas was approached in the order given by the list of program parts in IV.2. Step 6, reconciling a world model with a task statement, was initially ignored on the basis that I could get by with simple path-finding problems that did not require reconciliation of two world models. Steps 1-5 were accomplished and then step 7, the implementation of a spatial planner, was tackled. I found that current techniques for path-finding were not really satisfactory for my purposes and that an approach based on the use of the skeleton of a shape promised to be useful. This work is described in the next chapter.

First I will describe the world model.

IV.4.1 Definition of the world model

The data structure for a world model consists of a tree of _nodes_ linked by _relations_. The root of the tree is a node corresponding to Utak, called $org, which has one son, $floor, corresponding to the floorspace. The sons of $floor correspond to the isolated objects on the tabletop. The node N corresponding to an object may in turn have sons if the shape of the object is complex and is best described in a hierarchical manner.

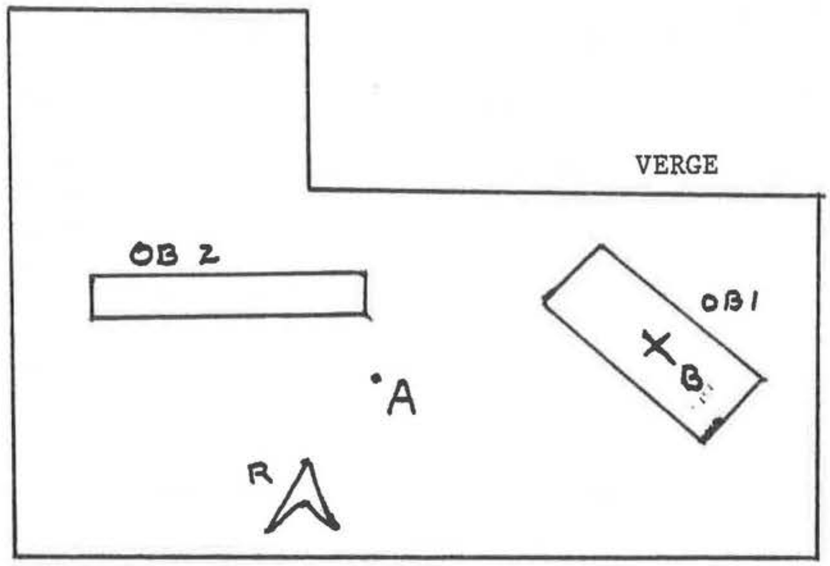
A node corresponds to a shape on the tabletop and is a list consisting of the containing rectangle (the smallest rectangle aligned with the axes that contains the shape), the actual boundary shape as a circular list of straight-line segments, the shapes of any holes if present, and a sublist consisting of the relations between this node and its descendants in the tree, if any. Each node has its own coordinate system, and a relation between two objects specifies the translation and rotation required to get from the coordinate system of one node to the other. A relation is simply a list (node1 node2 offset) where the offset is a triple consisting of the x- and y-translation and the rotation to get from node1's coordinate system to node2's. In the implementation so far, the rotation has always been zero.

By specifying the world model in this manner it is easy to modify the world model to reflect the motion of Utak or the motion of an object caused by the motion of Utak. All that has to be done is to modify one relation. This tree representation can also be used to specify a complicated shape in increasing levels of detail. Figure IV.3 shows the world model tree for a tabletop situation.

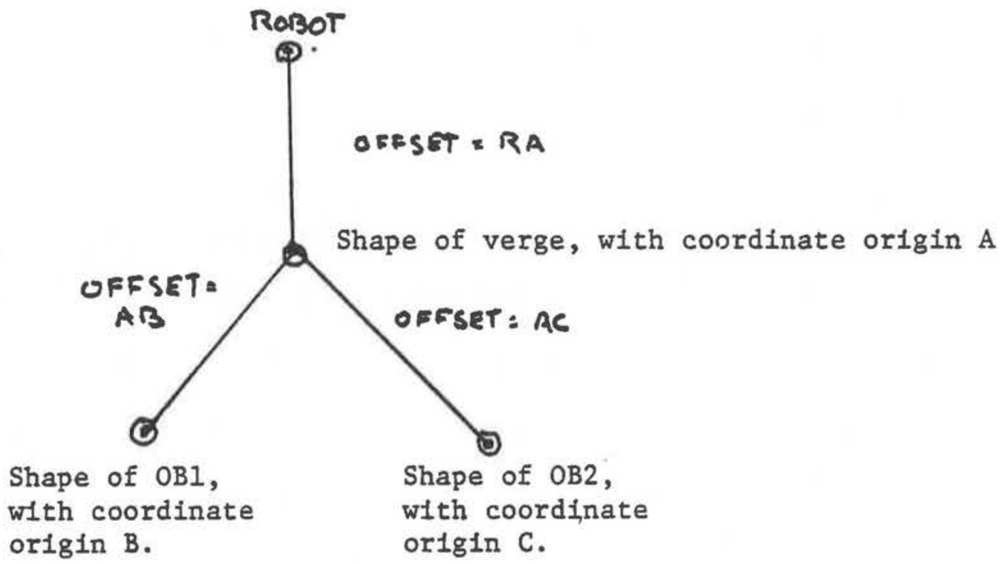
IV.4.2 Perception: accommodation to the first retinal impression

When a retinal impression has been received it must be analyzed to find those regions which represent space and those which represent objects or verge. Then the object regions must be distinguished from the verge regions in an interpretation stage so that finally the world model can be modified to accommodate (explain) them. The next three sub-sections describe these operations (steps 3,4,5 of IV.2).

FIGURE IV.3



(a) A world model, drawn on the Euclidean plane.



(b) Tree corresponding to the world model of (a).

IV.4.2.1 Edge and region finding

A region in the retinal impression is a connected set of retinal cells where all the cells have a zero graylevel, or else all have a non-zero graylevel. A connected set on the retina is defined using edge-adjacency. Two cells are edge adjacent if they have an edge or part of an edge in common. Thus diagonally adjacent cells are not edge adjacent. A region is connected if, for any two retinal cells in the region, there is a chain of edge-adjacent retinal cells which starts at one of the given cells and finishes at the other. The border between two regions has a direction associated with it, such that a turtle, crawling along the border in this direction, would always find the non-zero region on its right. Thus the (outer) boundary of a region with a non-zero graylevel would be traversed by the turtle in a clockwise direction and any other boundary (i.e. a hole) of the region would be traversed in a counter-clockwise direction.

The basic building block for finding the edges of a region is the "inter-cell-edge" (ICE) which is created between any pair of retinal cells where the graylevel of one is zero and the graylevel of the other is non-zero. A first scan of the retinal impression produces all the connected regions and marks the position of each ICE with a data structure linked to each member cell of the ICE.

The next operation links the ICEs of one region into one or more disjoint circuits. Since more information is needed for later grouping operations, when an ICE is being traversed several extra pieces of information are added: the direction of traverse (N E S W), the graylevels of the cells of the ICE, and the sizes of these cells, the result being called a chunk. The last two items are needed later as evidence for the segments of the world model. Some ICEs lie along the edge of the retina and special chunks have to be used.

The linking is done by taking one ICE associated with the region in question and iteratively finding its successors using the geometry of the retinal impression until a closed circuit is formed. If an ICE associated with the region still remains unlinked in a circuit then another circuit is started. This is continued until all the ICEs are exhausted.

Next, a grouping operator traverses each chunk-circuit and groups consecutive chunks having the same compass-direction into a segment. A segment is a list

(pt1 pt2 dirn len type evidence)

where pt1, pt2, dirn, len, specify the endpts, direction, and total length respectively of a straight line segment. "Type" is EXTERIOR or INTERIOR depending on whether the segment coincides with an edge of the retina or is interior to the retina. "Evidence" is a list of pairs (graylevels cell-sizes) derived from the component chunks, shortened by grouping identical pairs together. The meaning of "evidence" here is how well the line segment was defined by the graylevels in the retinal impression. It is intended to provide restrictions on how much the parameters of this line segment could be changed to accommodate subsequent retinal impressions without losing consistency with the current retinal impression. The end result of this operation is a segment-circuit.

Suppose a turtle traverses an arbitrary closed circuit of straight lines that nowhere crosses itself and counts 1 for every right-handed 90° turn and counts -1 for every left-handed 90° turn. At the end of the trip the total will be either 4 for a clockwise traverse or -4 for a counter-clockwise traverse. This is also known as the Total Turtle Trip theorem. In one final operation, each segment-circuit is traversed once and the total turtle turns computed. At the same time, the maximum number of consecutive segments that coincide with an edge of the retina is found. This is called "max-resegs" below.

The final output of the edge and region finding stage is a list of regions, where each region has on its p-list the grl-class, number of squares, and a border-list of one or more borders. Each border is a p-list with properties name, segment-circuit, turtle-turns, max-resegs, and their values.

IV.4.2.2 Interpreting the first retinal impression

In the fovea of the eye a retinal cell with zero graylevel corresponds to floorspace, and a cell with non-zero graylevel, necessarily 7, corresponds to either an object, the verge, or to Utak himself. In the peripheral parts of the eye a retinal cell with zero graylevel may correspond to an area of the tabletop which is not entirely floorspace, and similarly one with a graylevel of 7 may correspond to an area of the tabletop which is not entirely object or boundary.

In any one retinal impression the regions of zero graylevel are interpreted as floorspace. Since all floorspace is connected, if two or more disconnected regions of zero graylevel appear in the retinal impression the interpretation must provide that these are connected. If all the floorspace regions have a segment coincident with an edge of the retina then no action needs to be done; it is assumed that they are connected outside the area of the tabletop covered by the retinal impression. If one of the two or more floorspace regions is completely surrounded in the retinal impression by a non-zero region then a passage must be hypothesized between the surrounded floorspace region and the nearest neighbouring floorspace region. The natural place to hypothesize such a passage, in order to keep its length to a minimum, is the line of shortest distance between two floorspace regions. This line of shortest distance can easily be found from the skeleton of the complement of the floorspace regions; this will be described in section V.4.2.

The non-zero regions are interpreted as either isolated objects or verge. This is straightforward in two cases. If a non-zero region completely surrounds a zero region, and is not itself surrounded by a zero region, then this is interpreted as the verge of the tabletop. If a zero region completely surrounds a non-zero region then this latter region is interpreted as an isolated object with a high degree of certainty.

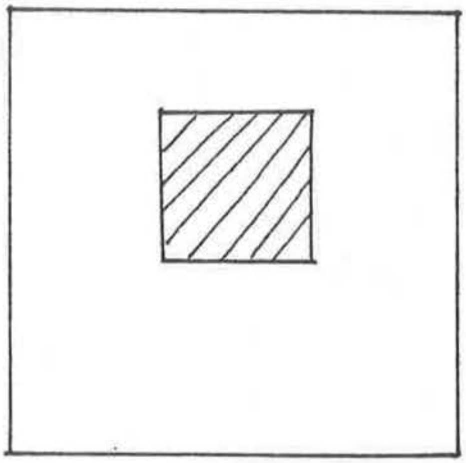
If neither of these two cases holds then the max-resegs of the outer boundary of the non-zero region (the unique border with turtle-turns=4) is examined. Remember that "max-resegs" records the maximum number of consecutive segments of the border of the region that coincide with edges of the retina. If its value is zero, one or two then the region is interpreted as an isolated object, otherwise as part of the verge. Figure IV.4 illustrates this interpretation scheme. Any interpretation made in this way is subject to revision or a "double-take" when subsequent retinal impressions are received. A region initially interpreted as an isolated object may later turn out to be more correctly interpreted as boundary, and vice versa. Figure IV.5 illustrates a possible double take.

IV.4.2.3 Accommodating the default world model to the first retinal impression

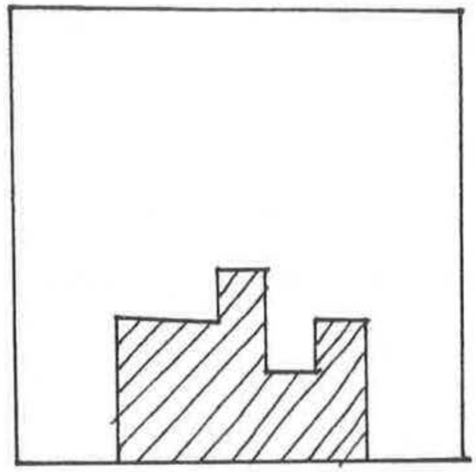
The default world model with which Utak "wakes up" is the simplest possible: a square floorspace centred on his position, and containing no isolated objects. After the first retinal impression has been received and interpreted, this world model must be modified (accommodated) to be consistent with this retinal impression.

First the interpreted retinal impression is examined for what restrictions, if any, the retinal impression places on the

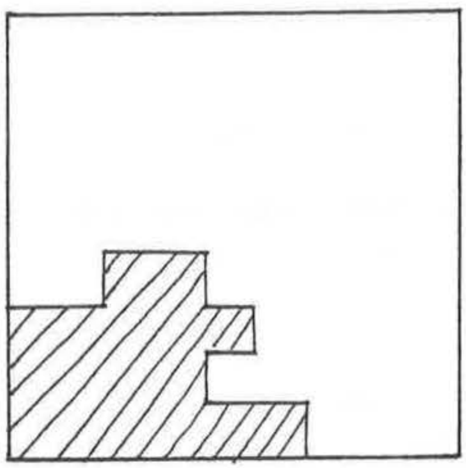
FIGURE IV.4 Examples of region interpretation.



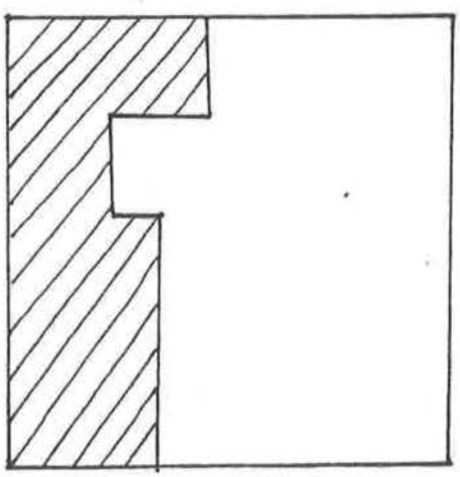
Max-resegs = 0
ISOLATED OBJECT



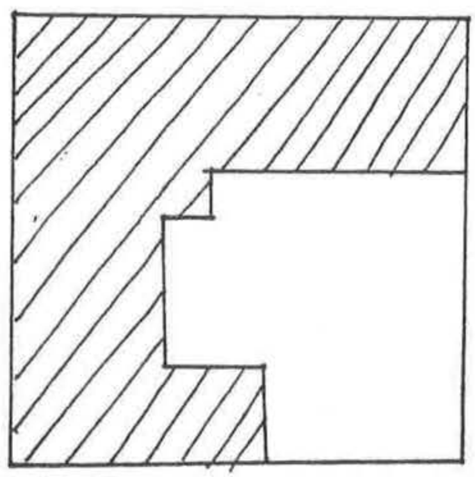
Max-resegs = 1
ISOLATED OBJECT



Max-resegs = 2
ISOLATED OBJECT

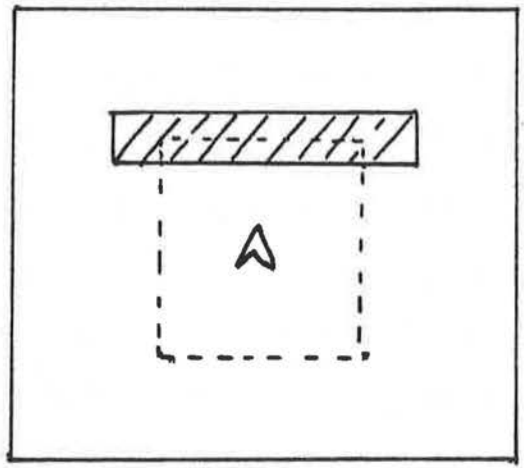


Max-resegs = 3
Verge

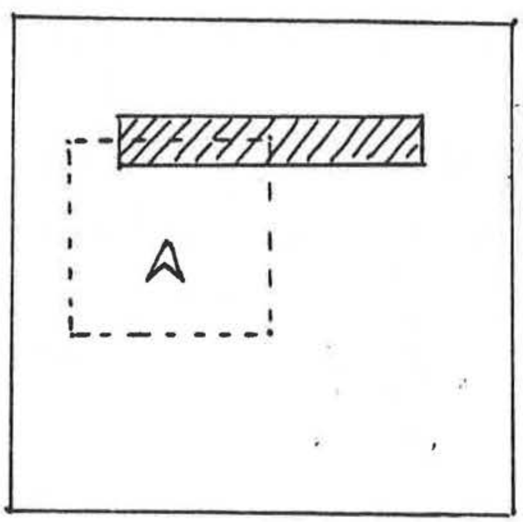


Max -resegs = 4
Verge

FIGURE IV.5 A doubletake.



In this initial position the robot thinks it is close to the north edge of its environment.



What the robot previously thought was verge is re-interpreted as part of an isolated object.

actual dimensions of the containing rectangle of the $floor of the world model. One restriction is computed from the retinal impression for each side of the $floor's containing rectangle, and consists of a value and a type. The type may be INTERIOR or EXTERIOR. If INTERIOR, the restriction is defined by a segment of the boundary of a floorspace region which is interior to the retinal impression; consequently the position of this side of the containing rectangle of $floor must be given by the value of this restriction. If the type is EXTERIOR, the restriction is defined by a segment of the boundary of a floorspace region which coincides with an edge of the retinal impression; consequently the position of this side of the containing rectangle of $floor must lie at or beyond (i.e. further N, E, S, or W than) the value of this restriction. The default $org-$floor offset, initially (0,0,0), and the sides of the containing rectangle of the default $floor, are compared with these retinal restrictions and modified as necessary to satisfy them.

Now that the coordinate origin and containing rectangle of the $floor is fixed, other parts of the world model can be computed. The next item is to derive the actual shape of $floor. If none of the segments of boundaries of floorspace regions coincide with a retinal edge, or in other words the whole of $floor appeared within the retinal impression, then the boundaries of the floorspace regions are taken as the shape of $floor. Otherwise, only parts of the shape of $floor appeared within the retinal impression. These are the sequences of segments of boundaries of regions interpreted as verge, which lie strictly within the current retinal impression. These segment sequences have to be linked up into one continuous segment circuit by creating hypothetical extensions to existing segments and by adding new hypothetical segments. First the end segments of each sequence are extended to the containing rectangle, with the extensions marked as "HYPO" in the evidence lists attached to these segments. Second, hypothetical segments along the edges of the containing rectangle are introduced between each consecutive pair of segment sequences, also marked "HYPO". The shape of $floor is then complete.

Lastly, a node has to be created for each isolated-object region of the retinal impression, and added to the world model. Given an isolated-object region, its containing rectangle, the offset of its coordinate system from $floor, and its shape are all computed. The default world model has now been accommodated to the first retinal impression.

IV.4.3 Perception: accommodation to subsequent retinal impressions

Once a world model has been constructed that correctly interprets ("explains" in the analogy with scientific method) the retinal impressions received so far, it is used to facilitate the accommodation of subsequent retinal impressions. An overall view of this accommodation process will now be described. Given the decided-on action, a predicted world model is constructed, and from this a predicted retinal impression is produced by projecting the predicted world model onto an array structure topolcgically identical with a retinal impression. This retinal impression differs from a normal retinal impression in that every retinal cell has a pointer back to the segment(s) of the world model which caused it to have its assigned graylevel. After the action is executed the actual retinal impression is compared with the predicted retinal impression and the differences between the two used to indicate what line segments and what parts of the world model must be modified to accommodate the new retinal impression.

Given an action, the predicted world model is constructed as follows. If the action is SLIDE($\underline{v}$) then the \$org-\$floor relation ($\underline{r}$,θ) is replaced by ($\underline{r}$-$\underline{v}$,θ). If the action is PUSHTO($\underline{v}$) and the object held is objecti, then the \$org-\$floor relation is replaced by ($\underline{r}$-$\underline{v}$,θ) and the \$floor-\$objecti relation ($\underline{s}$,ϕ) is replaced by ($\underline{s}$+$\underline{v}$,ϕ). If the action is TURN(ψ) then the \$org-\$floor relation becomes ($\underline{r}$,θ-ψ) and if Utak is holding \$objecti then the \$floor-\$objecti relation becomes ($\underline{s}$,ϕ+ψ).

From the predicted world model a predicted retinal impression is computed as follows. Each line segment in the shape of the floor, verge, or an object of the predicted world model is traced across the retinal fields of the eye and a pointer is set from each retinal field intersected back to the intersecting segment. Any retinal cell not intersected by a segment of the border of an object is labelled by the name of the object which encloses it, if any, and given a graylevel of 7, else it is assumed to be part of $floor and given a graylevel of zero.

The next step is to compare the graylevels of the predicted retinal impression and the actual retinal impression, cell-by-cell, and note any differences. The differences immediately focus attention on those parts of the world model that must be changed. Some differences can be accounted for by changing the parameters of line segments in the world model; others require more drastic action. For example if a non-zero region in the actual retinal impression has no overlap with any non-zero region in the predicted retinal impression, then it has to be interpreted and added to the world model as a new isolated object in just the same way as a similar region in the first retinal impression is acommodated by the initial default world model. This, in outline, is the accommodation process.

In a "realistic" simulation where an element of randomness is allowed in the effect of an action the differences between the actual and predicted retinal impressions may arise from two sources of uncertainty:

(1) New parts of the environment coming into view or old parts seen at higher resolution;

(2) The action is not as predicted.

If only (1) is allowed then any differences must be explained by modifying the world model. If only (2) is allowed the problem is to recognize what position in the world model could give rise to the actual retinal impression and thus deduce what actually happened. Since the difference between the actual and predicted effects of an action will normally be quite small the problem is one of computing the disparity between the two retinal impressions, say by trying the positions closest to the predicted position. If the difference is too great for a disparity to be found then it becomes a pure recognition problem to be approached, say, by matching features. If both (1) and (2) occur together then the problem is to find a match in the world model for as much as possible of the retinal impression, thus proposing a new position, and then modifying the world model to explain the remaining unmatched parts of the retinal impression. Although (2) is easily incorporated in my system, I have not attempted to handle this possiblity.

IV.4.4 Accommodation, another approach

The problem is, given the current retinal impression, to modify the current world model as necessary to be consistent with it. This can be broken down into three processes. The first process interprets the retinal impression, regarded as a structured array of graylevels, as a consistent collection of line segments. The adjective 'consistent' implies that, locally, the line segments make sense or in other words form continuous lines. Moreover straight lines are preferred to lines with many changes of direction. The next process interprets the lines formed by the line segments as the (partial) contours of the verge or of an isolated object. The third process extends these contours by rules of continuation to a complete world model. The last two processes are straight-forward, the first one is harder.

In more detail, the first process can be described as follows. In the fovea the graylevel of a retinal field is 0 or 7 according as the corresponding TABLE square is occupied or not. Trivially, a line segment is inserted between each pair of adjacent cells with differing graylevels. In the middle part of the retina the graylevels which can actually occur are 0,2,4,6,7. Each graylevel has a distinct class of potential interpretations where an interpretation is a 2X2 pattern of occupied and vacant squares. In the periphery all the graylevels 0,1,...,7 can occur and the potential interpretations of a graylevel are specified by classes of 4X4 patterns of squares. Each assignment of a line segment to a retinal field must be consistent with the assignements of neighbouring retinal fields, such that the line segments constitute a continuous line and preferably a straight line. Thus this first process is a problem of finding a consistent interpretation and can be tackled by the NC consistency algorithm of [Mackworth,1977], by the relaxation algorithms of [Zucker,1977], or the improved backtracking algorithms of [Gaschnig,1978]. The latter two processes, line segment interpretation and continuation, can be handled as before.

IV. Towards the design of a robot-controller

IV.4.5 The spatial planner

This is the heart of the system. This is where path-finding is done, where a plan for moving an object is constructed and where the initial task command is interpreted. Given an interpreted task description, the spatial planner uses the current world model as a database and produces a structured plan as output. If the current world model reflects the "real world" outside the organism sufficently closely, or at least models closely enough those parts of the world model required for this task, then a completely accurate execution of the plan will result in the successful completion of the task.

The spatial planner does not produce plans from the world model directly, but indirectly via the screen, a 2-dimensional digital array. Consider a simple pathfinding problem, "Go to the north east corner" for example. First the dimensions of a rectangular window which includes both the current position of Utak and the position of the destination are computed. Then the world model is projected through this window onto the screen and each square of the screen marked as representing space or with the name of the object ovarlaying it. Then a pathfinding algorithm is used (cf. chapter 5) to find a path from start to destination that only traverses cells representing space. If this fails then other potential paths are considered. These arise where two adjacent squares of the array are marked with names of different objects or where a single square of the screen is marked with the names of two or more objects. These positions can be used to trace possible paths between objects and, where such a position is adjacent to squares representing space, to connect up these possible paths with other paths through space. If the path between two objects is required in more detail then a small rectangular window is selected that includes only the potential passage between the objects. Then the world model is projected onto the screen again and this part of the path traced at a higher resolution. The result is included as a more detailed subpath in the previous coarser path.

IV.5 An alternative approach to implementation

The implementation sketched above uses a representation for the world model based on the use of Cartesian coordinates and then computes, via projection onto a screen, the skeleton, a graph-like representation of the world model. The edges of this graph represent routes in the environment. A representation in terms of routes has been used by Kuipers to model a person's representation of large-scale (city) space, and by Arbib and Lieblich to model a rat's representation of space. The natural suggestion, then, is to use the skeleton as the basic representation for the world model. This depends on the fact that the complete skeleton of a shape, consisting of the skeletal graph plus the quench function, can be used to recover the original shape in its entirety. The world model then consists of a graph, whose edges represent routes through the environment, together with a quench function defined on each edge. Pathfinding is done directly on this graph. One way to carry out perception is to regenerate the original shape in the predicted retinal impression and then match it with the actual retinal impression. Any shape changes forced by the actual retinal impression are reflected in changes to the skeletal graph or the quench function. Alternatively, instead of doing matching on shapes, matching can be done on graphs. That is, the skeleton of the spaces on the retina is matched against the stored skeleton and changes made as appropriate to the stored skeleton.

IV.6 Summary

The design of the whole organism-controller is of major importance for my project. I described in this chapter the first approach that I took, based on the analogy between the process of perception and the scientific method, and the progress made in implementing it. There are two outstanding problems inherent in this approach. One is defining exactly what is meant by the phrase "collecting evidence for a feature of the world model", and the other is defining a phrase such as "assigning a degree of confidence to a feature on the basis of the evidence". I approached the first problem by retaining with each feature both the graylevel values in the retina and the retinal field sizes, from which the feature parameters were derived. In line with the analogy, one can say that "the feature explains the graylevel values". The problem of defining a measure of confidence, showing how it changes with the receipt of new evidence, and using the confidence values to control how the features are allowed to change when new evidence shows that a change must be made, is an important and interesting problem that must be solved before completion of this project is possible. This problem bears comparison with the legal problem of evidence in a court of law, where the degree of confidence in a statement depends upon the evidence presented.

There is one major requirement that remains to be described, namely algorithms for pathfinding and for making plans for moving an object.

CHAPTER V

PATH-FINDING AND THE SKELETON OF A PLANAR SHAPE

The purposes of this chapter are to review path-finding algorithms, to motivate the use of the skeleton for path-finding, to introduce a new algorithm for computing the skeleton, and to apply the skeleton to path-finding.

V.1 Introduction to path-finding algorithms

The problem is this. Given a description of shapes on the Euclidean plane in terms of the Cartesian coordinates of points on the boundaries of the shapes and the lines between them, and given the coordinates of two points S and D outside all the shapes, describe a path from S to D that avoids all the shapes, if such a path exists. Further requirements are that the path should be reasonably close to being optimal, and that if an organism wanders slightly from the correct path, either due to inaccurate movement or to avoid a small obstacle, it should be easy to regain the correct path.

A path-finding algorithm was incorporated in the design of Shakey [Nilsson, 1969]. This was based on the observation that in a cluttered space an optimal path between two points consists of a sequence of line segments connecting extreme points of obstacles. Thus one starts with the extreme (convex) points on the obstacle boundaries and considers the set consisting of the lines joining the pairs of extreme points together with lines from the starting point S to the extreme points and with lines from the extreme points to the destination point D. Any line which intersects an obstacle is discarded, so that the remaining lines represent all the "lines of visibility" in the situation. A heuristic search is then used to find the optimal series of points connected by lines of visibility from S to D.

Another approach to path-finding, but still using a Cartesian representation, was used by [Thomson,1977] for the path planning module of the JPL robot. This was the approach I first took to the problem. The zero'th order approximation was the straight line SD. The first order approximation was obtained as follows. Determine those shapes intersected by SD, if any. If none, then the straight line SD is the required path. Otherwise pick the obstructing shape nearest to S, call it B say, and compute four points L1,R1,L2,R2 on the periphery of B defined as follows. L1, R1 are the leftmost, rightmost points respectively of B as seen from S, while L2, R2 are the rightmost, leftmost points respectively of B as seen from D. There are now two candidate paths at the first order level of approximation:

$$P11 = SL1 + L1L2 + L2D$$

$$P12 = SR1 + R1R2 + R2D$$

where the path segments SL1, L2D, SR1, R2D are known not to intersect object B. See figure V.1. Now recursively apply the

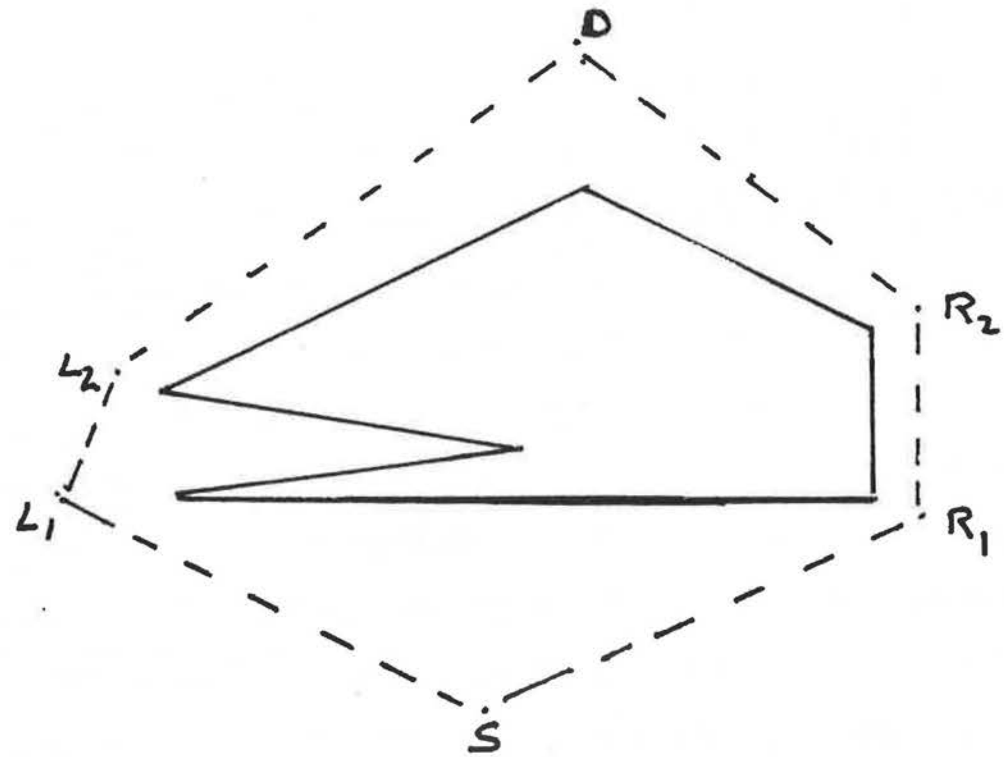


FIGURE V. 1 AND/OR APPROACH TO PATH-FINDING above procedure to each of the segments SL1, L1L2, L2D, SR1, R1R2, R2D. Eventually the procedure stops , there being only a finite number of objects and only a finite number of points on the boundary of each object, with a perhaps long list of possible paths. Each path can be evaluated in terms of total length, number of segments, total angle turned, etc., and the best one chosen. The disadvantages of this direct approach to the path finding problem are the following.

A) It involves finding the intersections of particular lines with every object and finding extreme points of those objects intersected, altogether an expensive computation in some cases.

b) There is no notion of "level of detail". All objects of whatever size are considered, and all lines which form the boundary of a shape have to be scanned for intersection finding.

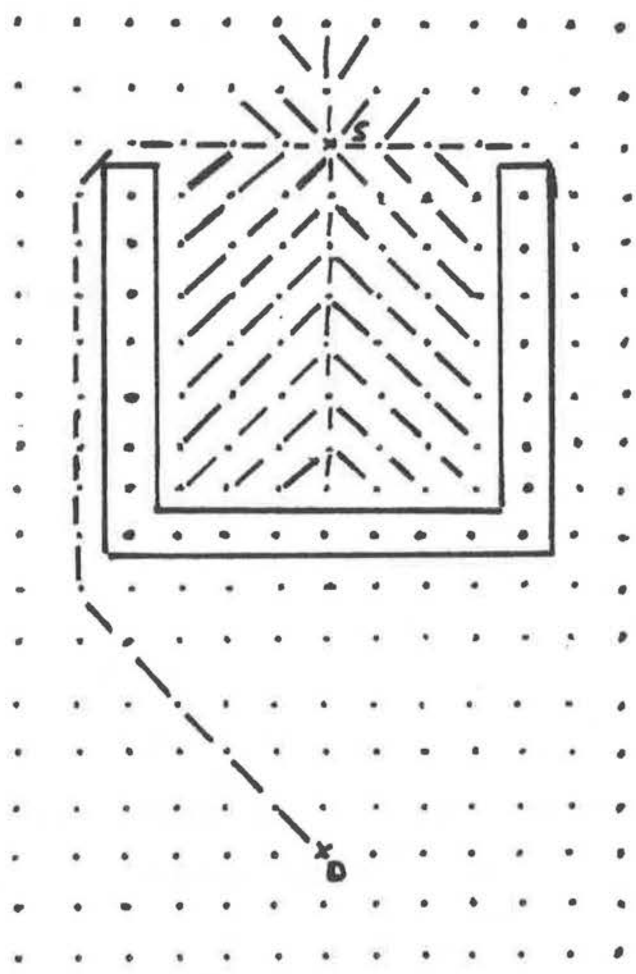
c) There is no obvious way to generalize to object moving.

An advantage of this approach is that it can produce a description of a path in "left-of object", "right-of object" terms.

Another idea is to project all the shapes onto a rectangular network of cells, the screen of chapter IV for example, and convert the path-finding problem into a pure graph-traversal problem. The screen is converted to a graph by inserting between every pair of adjacent or diagonally adjacent cells an edge of the graph, where each edge is assigned a length of root2 or one according as its endpoints are diagonally adjacent or not. It is forbidden to traverse edges leading to cells that do not represent floorspace. The start and destination points are mapped onto cells S and D of the graph. The problem can now be restated as: find the shortest path from S to D along the edges of this graph. This can be done by an application of the A* algorithm of [Hart,Nilsson, & Raphael,1967], using their function f=g+h to evaluate incomplete paths. For an incomplete path p that currently terminates at a cell n, g(p) is the sum of the lengths of the edges from S to n and h(p) is the Euclidean distance from n to D. This algorithm finds the optimal path through the graph directly, but suffers from some disadvantages. These include: no obvious way to generalize to handle object moving; no easy way to discover what objects a section of the path is passing between; no distinction between topologically distinct paths. In addition the actual search process involves a high number of "knowledge independent" choices. As an extreme example of what I mean by "knowledge independence", witness the bucket phenomenon illustrated in figure V.2. In using heuristic search to go from S to D, the search goes straight to the bottom of the bucket and gradually fills up until suddenly it overflows and rapidly advances towards D. Worse cases can also occur.

Finally the skeleton of the shape of the empty space was considered. The initial attraction was that the skeletal graph contains only the topologically distinct paths between two

FIGURE V.2 The bucket phenomenon, which occurs when the A* search algorithm is applied to a network representation of space.



positions. Two paths are <u>topologically</u> <u>distinct</u> if neither can be continuously deformed into the other. The search for a path then reduces to searching over the topologically distinct paths. The choice points in the skeleton seem, intuitively, to correspond to the choices we have to make in navigating through obstacles. In addition there is a great deal of information associated with the skeleton which can be used in other spatial problems. The optimum path, when restricted to edges of the skeletal graph, is not in general the optimum path when no such restrictions are made; however, my initial concern was to compute any reasonable path, not necessarily an optimal one.

V.2 The skeleton

[Blum,1964], who called it the Medial Axis Function, was the first to introduce the skeleton of a planar shape. Since then it has been the topic of several investigations [Calabi and Hartnett,1968; Montanari,1968,1969; Rosenfeld and Pfaltz,1966; Pfaltz and Rosenfeld,1967; and others] and has been called the distance transformation, the grassfire transformation, or the symmetric axis transformation. [Blum,1973,1974] has comprehensively analysed it and written about its potential applications to the description of shape in biology.

V.2.1 Definition and properties

In the continuous Euclidean plane several equivalent definitions of the skeleton can be given; however when these are converted to algorithms to compute the skeleton on a rectangular network if cells, it turns out that this equivalence no longer holds.

First the several equivalent definitions in the continuous Euclidean plane will be described, second Montanari's algorithm will be derived from one of these definitions, third we will see why this algorithm is unsatisfactory, and fourth we will see how to augment Montanari's algorithm to provide a satisfactory algorithm for the skeleton on a network of cells.

Definition 1. Interpret the boundary of the shape as a wavefront which propagates at uniform velocity into the interior of the shape. At certain points two or more sections of the wavefront emanating from distinct points of the boundary meet and mutually extinguish themselves; the locus of these points of extinction is the skeletal graph and the time from the initiation of the wavefront to the time of extinction is the quench function.

Definition 2. The most concise definition for mathematical purposes is probably this. Consider the set of all circles contained in the shape and partially order them by inclusion. Then the skeletal graph is the locus of the centre of maximal circles, and for each point on the graph the radius of its maximal circle gives the value of the quench function.

<u>Definition</u> <u>3</u>. At every point P of the plane define the function d to be the minimum distance from P to the shape:

$$d(P)=\min\{ euc(P,Q) \mid Q \in shape\}$$

where euc is the euclidean distance. For every point P there is at least one point Q ∈ shape such that d(P)=euc(P,Q), while for certain points P* there are at least two distinct points Q1, Q2 such that d(P*)=euc(P*,Q1)=euc(P*,Q2). Q1, Q2 are <u>contact points</u> for P*. The locus of points with two or more contact points is the skeletal graph and the value of the function d at each point of the graph is the quench function.

<u>Definition</u> <u>4</u>. Given a point P, a minimal path from P to the boundary of the shape is a straight line segment PR where R lies on the boundary. P is defined to be a point of the skeletal graph if it does not belong to a minimal path of any other point. In the mountain metaphor, this says that any point on a ridge-crest or peak does not lie on the fall-line of some higher point. This is the definition used by [Montanari,1968].

Definition 3 lends itself to the following visualization of the skeleton. Imagine the boundary of the shape as the shoreline of an island, which everywhere rises uniformly at an angle of 45° out of a calm ocean, thus forming a mountain range with peaks and ridges. It is possible for three or more ridges to meet at a junction which is not a peak. Holes in the shape are to be visualized as lagoons at ocean level. The projection of the ridges and junctions to the plane of the ocean is the skeletal graph and the height of the ridge at each point, i.e. the function d, gives the quench function. Picture yourself standing on the crest of the ridge; in your immediate vicinity there are exactly two well-defined directions in which to travel along the ridge, "forwards" or "backwards". Now stand at a junction ; three or more ridges meet there, providing three or more corresponding well-defined directions of travel. That is the content of [Calabi and Hartnett,1968, Theorem 6]. Now return to a point on the ridge crest and note that there are exactly two lines of steepest descent, or <u>fall-lines</u> in skiers parlance, from the ridge-point down to the ocean. The points where the fall-lines from a ridge point or peak enter the ocean are called <u>contact points</u>. Back at a junction again, where n ridges meet, note that there are n fall-lines to the ocean, alternating with the ridges. That is the content of [Calabi and Hartnett,1968, Theorem 7].

At any junction there can be at most one ridge which ascends away from it; all the others must be descending from it. This can be seen as follows. Suppose P is a junction where 3 ridges meet. Let the three contact points be p, q, r on the circumference of the maximal circle centre P. One may without loss of generality consider the skeleton generated by these three points alone. Consider the ridge which starts at P and passes midway between p and q. The quench function must decrease from P to the midpoint of p and q, since the distance pq is less than the diameter of the maximal circle, unless p and q lie on the ends of a diameter through P. In this latter case the quench function increases at P as one begins to move out along the ridge. Also, since pq is a diameter, neither qr or rp can be a diameter, and hence the value of the quench function as one starts out along these other two ridges is decreasing.

V.2.2 Approximating the Euclidean plane

To obtain the skeleton of a shape the euclidean distance between two arbitrary points is required. But when the shape is given as a digital image, that is to say as a rectangular network of cells where each cell is marked with a 1 (outside the shape) or a 0 (inside the shape), this network can be treated as an approximation to the euclidean plane. By connecting each cell to a number of its neighbours and measuring distance between cells by summing over all the links between them an approximation is obtained. Figure V.3 shows 4-, 8-, 16-connected cells and figure V.4 shows networks of 4-, 8-, 16-connected cells. The n directions of the links in an n-connected network will be referred to as the <u>major directions</u> of the network. The higher the connectivity the better the approximation. Using only 4-connectivity, the distance between two cells in a network can be as much as 41% out from the euclidean distance, whereas with 8-connectivity it can be 8% out

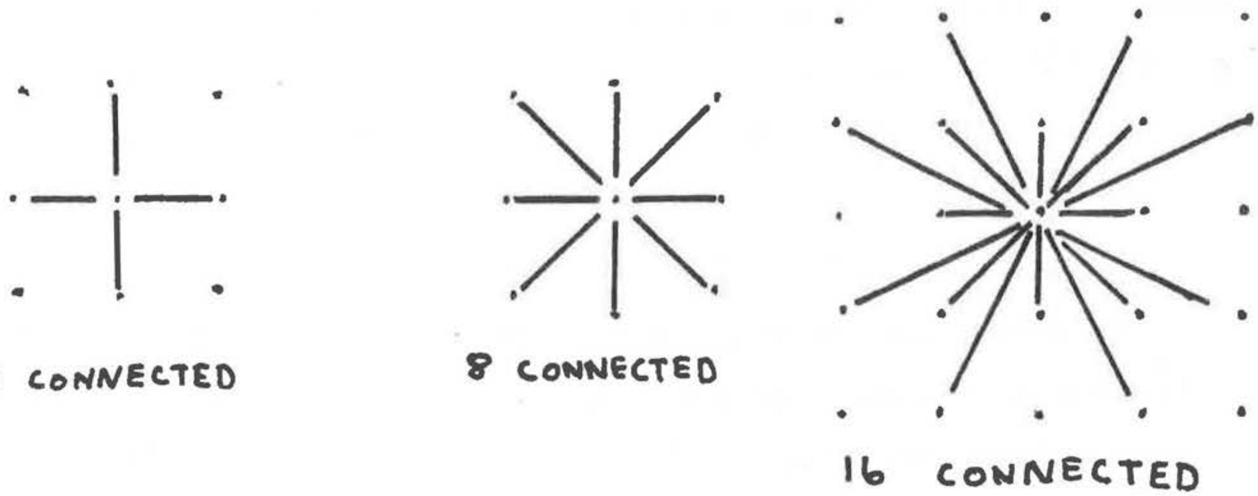
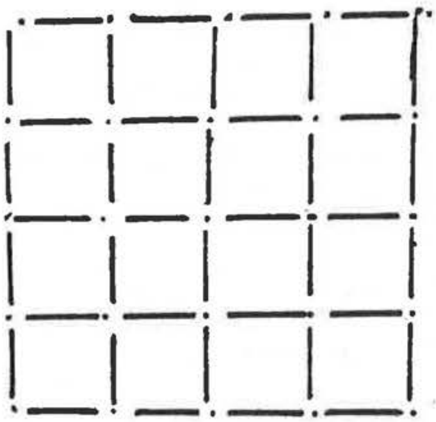
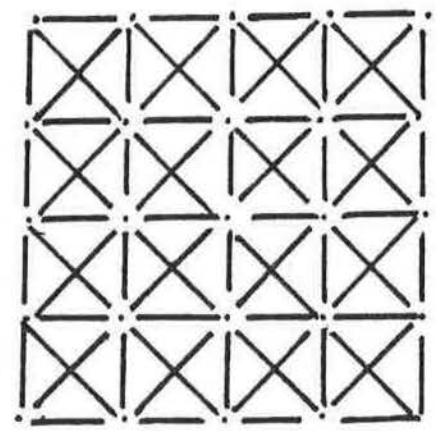


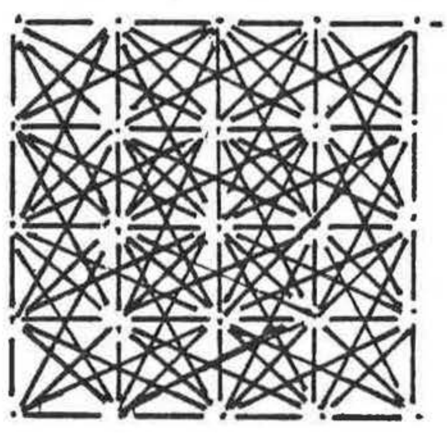
FIGURE V 3



4 - CONNECTED NETWORK



8 - CONNECTED NETWORK



16 CONNECTED NETWORK

FIGURE V. 4.

and with 16-connectivity only 2.7% out.

Given two cells in a network, if the line connecting them happens to lie parallel to one of the major directions then there is a unique shortest path between the two cells, otherwise there will be many shortest paths between them. This is illustrated in figure V.5. Note that there is always a shortest path consisting of exactly two straight line segments.

V.2.3 Montanari's algorithm

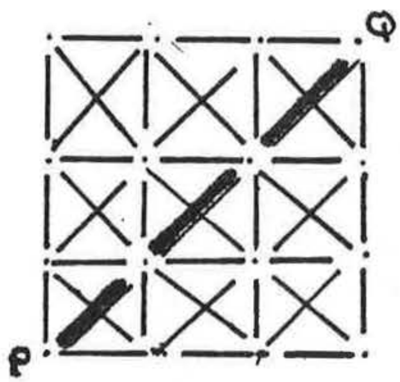
Using definition 4 above Montanari showed that the skeleton finding problem was equivalent to a certain optimal policy problem, and thus derived a two part algorithm which can be stated as follows.

a) For each cell of the network find the minimum distance to the boundary of the shape (the "height"), and find all the fall-lines (necessarily at least one) from the cell to the boundary.

b) Classify as skeleton points those cells which do not lie on a fall-line descending from any other cell.

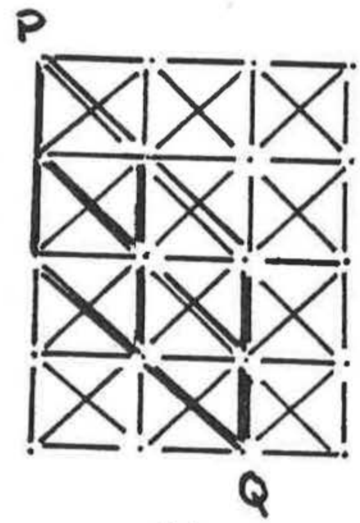
The distance to the shape can be computed using an algorithm which requires only two passes over the network, which will now be described. Divide the directions of the links between cells into two classes, UPPER and LOWER, as shown in figure V.6. The first pass is in forward raster order and computes the minimum distance to the boundary when only links with UPPER directions are considered. The second pass is in

FIGURE V.5 Shortest paths in an 8-connected network.



(a)

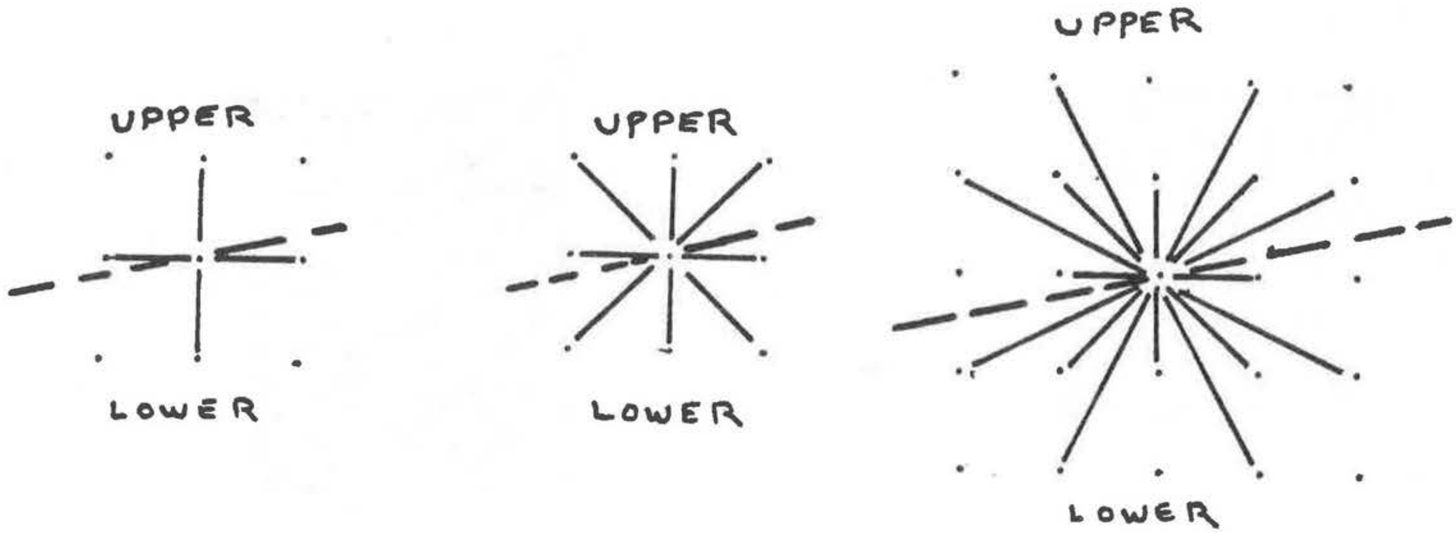
The unique shortest path from P to Q.



(b)

Multiple shortest paths from P to Q.

FIGURE V.6



UPPER and LOWER directions of links are considered in the first and second passes respectively of the iterative algorithm.

backward raster order and continues the minimum distance computation, adding in those links with LOWER directions. This is spelt out in steps 1 and 2 of SKEL-3 below. Surprisingly, after these two passes the minimum distance at every cell is correctly computed, even for cells whose fall-line(s) include links with both UPPER and LOWER directions. A proof of this can be found in [Montanari,1968].

Some notation is in order. Let the network cells in forward raster order be P1,P2, ... ,Pn. Cells Pi, Pj are <u>neighbours</u> if there is exactly one link between them, and are <u>HV-adjacent</u> if they are neighbours and their link is horizontal or vertical. Let Tij = T(Pi,Pj) be the distance between Pi and Pj. In the case of an 8-connected network, the possible values for Tij are 1 and the square root of 2, and in a 16-connected network, 1, the square root of 2 and the square root of 5. <u>UPPER(Pi)</u> {<u>LOWER(Pi)</u>} denotes the cells reached from Pi by traversing one link in an UPPER {LOWER} direction. Let NBRS(Pi)=UPPER(Pi) u LOWER(Pi) and let I be the cells outside the shape. Di, the minimum distance from Pi to the boundary of the shape, is to be computed for i=1,2, ... ,n.

Algorithm SKEL-1.

```
1. [Forward raster scan].
   For i=1,2, ... ,n do
       If Pi ∈ I, Di=0
       If Pi ∉ I,
          Di= oo if UPPER(Pi) is empty
          Di=min{ Tij+Dj | Pj ∈ UPPER(Pi) } otherwise.
2. [Backward raster scan].
   For i=n,n-1, ... ,1 do
       Di=min{ Di,{ Tij+Dj | Pj ∈ LOWER(Pi) } }.
3. [Define skeleton points].
   SKEL={ Pi | Di ≠ Dk-Tik for all Pk ∈ NBRS(Pi) }.
```

Two skeletons computed with this algorithm are shown in figure V.7. Note how they compare with their euclidean skeletons. As can be seen from this example, this algorithm suffers two deficiencies. First, the output is an unstructured set of points - no method is provided to link up the points into a graph structure. Second, the set of skeleton points is in general disconnected, so that it would be impossible to form a graph structure anyway.

V.2.4 The new algorithm

This last deficiency can be remedied by using definition 3, which defines the skeleton to be the locus of points with at

Figure V.7

```
I I I I I I I I I I I I I I I I I I I I
I *                                 * I
I   *                             *   I
I     * * * * * * * * * * * * * *     I
I     * * * * * * * * * * * * * *     I
I   *                             *   I
I *             * *                 * I
I I I I I I I   * *   I I I I I I I I I
I I I I I I I   * *   I I I I I I I I I
I I I I I I I   * *   I I I I I I I I I
I *             * *                 * I
I   *                             *   I
I     *       * * *           *       I
I       * * * * * * * * * * * *       I
I     *             * * * * * *       I
I   *                         * *     I
I *                 *           * *   I
I I I I I I I I I *             * *   I
I I I I I I I I I I I I I I I *     * I
I I I I I I I I I I I I I I I I I I I I
```

```
I I I I I I I I I I I I I I
I *                     * I
I   * * * * * * * * * *   I
I   * * * * * * * * * *   I
I *                     * I
I I I I I I I I I I I I I I
```

least two contact points. This suggestion can be implemented as follows. First, while computing the minimum distance to the figure, compute also the contact points for each cell. Second, classify all cells with at least two contact points as belonging to the skeleton.

Since the fall-lines and contact points of a cell cannot be defined until the final value of the minimum distance from the cell to the shape has been computed, they cannot begin to be computed until the backward raster scan of SKEL-1. Step 2 requires an extra operation, another raster scan is needed and step 3 must be replaced. This results in the following algorithm.

<u>Algorithm</u> <u>SKEL</u>-<u>2</u>.

1. [Forward raster scan].

    For i=1,2, ... ,n do

        If Pi ∈ I, Di=0

        If Pi ∉ I,

            Di= oo if UPPER(Pi) is empty

            Di=min{ Tij+Dj | Pj ∈ UPPER(Pi) } otherwise.

2. [Backward raster scan].

    For i=n,n-1, ... ,1 do

        Di=min{ Di,{ Tij+Dj | Pj ∈ LOWER(Pi) } }.

        Contacts(Pi)=U { contacts(Q) | Q ∈ LOWER(Pi) &

                                        d(Pi)=T(Pi,Q)+d(Q) }

3. [Second forward raster scan].

For i=1,2, ... ,n do

contacts(Pi)=contacts(Pi) u

U { contacts(Q) | Q ∈ UPPER(Pi) &

d(Pi)=T(Pi,Q)+d(Q) }

4. [Define skeleton points].

SKEL={ Pi | number-of-contacts-of(Pi) > 1 }

This improves matters a little, for it correctly classifies as skeleton points many points omitted by SKEL-1. At the same time, unfortunately, at places in the skeleton such as a straight corridor of even width, where SKEL-1 would compute a double row of skeleton points, SKEL-2 computes none.

What is needed is the introduction of points in between cells of the array as skeleton points, as suggested by the examples of figure V.7. The obvious way to do this is to introduce a skeleton point between any two HV-adjacent cells with disjoint sets of contact points. This doesn't work, however, because many pairs of HV-adjacent cells, far from any ridge-line, have HV-adjacent contact points, and consequently many spurious ridge points would get introduced. Consequently a more conservative condition must be used. Define two cells of the network to be neighbourly if they are identical or are HV-adjacent, and two non-empty sets S1, S2 of cells to be neighbourly if there is at least one neighbourly pair of cells, one member of the pair from S1 and one from S2. The modified algorithm is as follows.

Algorithm SKEL-3.

Steps 1,2,3 as for SKEL-2.

4. [Create ridge points].

For all i,j=1,2, ... n, (i < j) do

if not neighbourly(contacts(Pi), contacts(Pj))

then create-ridge-pt Rij between Pi and Pj.

5. [Define skeleton points].

SKEL= { all created ridge-points }

u { Pi | number-of-contacts-of(Pi) > 1 }

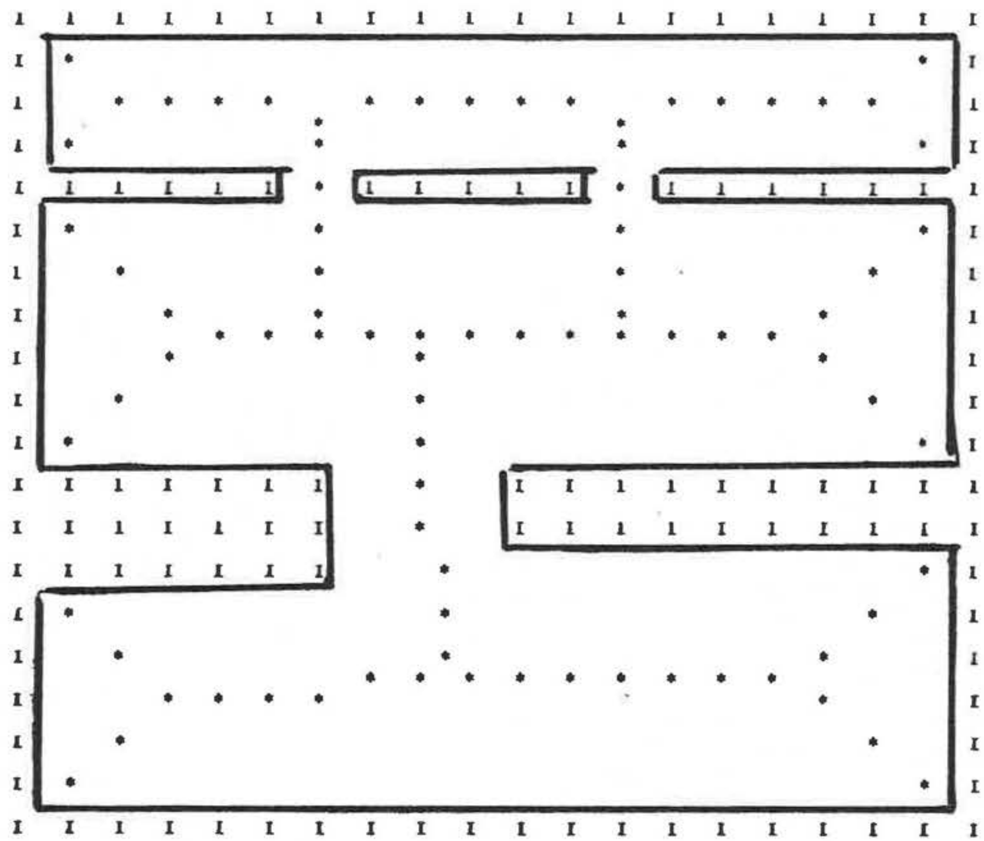
Figure V.8 shows the result of using SKEL-3 on two examples.

V.2.5 Ridge-following

The points of the skeleton will be referred to as *ridge cells*. A ridge cell may be either a cell of the original network or a point created between two cells of the original network. The output of SKEL-3 is an unordered collection of ridge cells which, to be useful, must be organized into chains of linked ridge cells. I use the term *ridge-following* to refer to the operation, on a network to which SKEL-3 has been applied, of inserting links between ridge cells and assembling three or more neighbouring ridge cells into "junctions" so that the resulting collection of cells, links, and junctions is a connected graph structure closely approximating the skeletal graph of the corresponding Euclidean shape.

Return again to the mountain metaphor. Imagine oneself straddling a ridge with one's left and right feet just to the

FIGURE V. 8a.



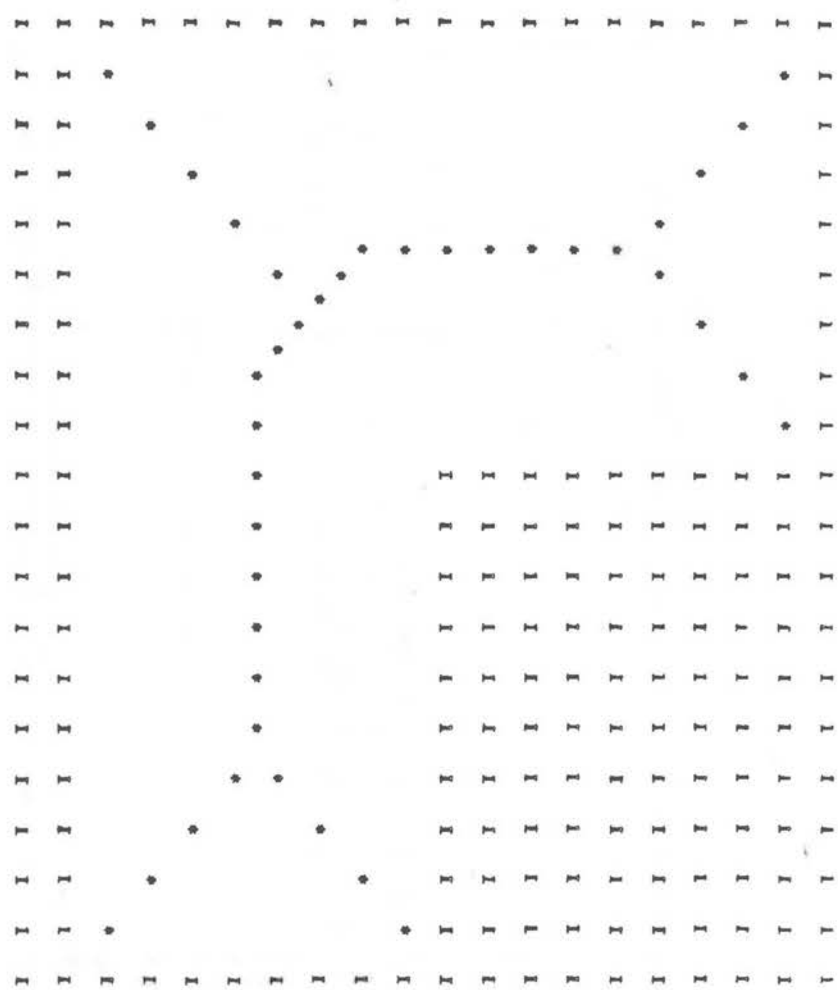
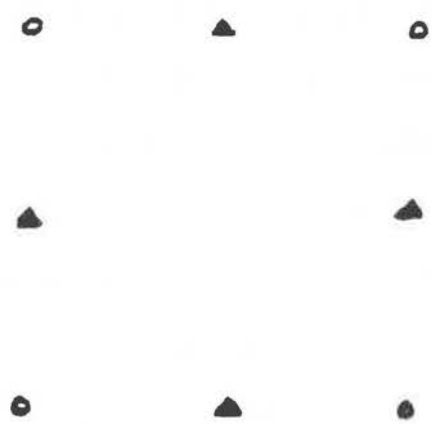


FIGURE V. 8b.

left and right respectively of the crest, and observe the contact points of the fall-lines which originate under one's left and right feet and descend on opposite sides of the ridge. The positions of these two contact points vary continuously with one's position on the ridge crest, except when a crossover point, where three or more ridges meet, is passed. The only exceptions to this statement could occur when the set of contact points includes an arc of a circle - but this cannot happen with polygonal figures or a digital image. Conversely the point on a ridge-crest from which a fall-line descends to a contact point on the shoreline varies continuously with the position of the contact point. In a network, this says that two ridge cells are to be linked only if their contact points are the same or are neighbourly in pairs.

To make this precise some more terminology must be introduced. A csquare is a unit square in the network with a cell at each corner. Any of the four corner cells may be a ridge cell, and in addition the centre of each side may be occupied by a constructed ridge cell. Thus a csquare may contain up to eight ridge cells, and there are $2^8=256$ possible different configurations of ridge cells on a csquare. This is illustrated in figure V.9. Two ridge cells are contiguous if they lie on a common csquare. Only contiguous ridge cells ever get linked together and when that happens let us say that they are crested and the link between them is a crest. Now the condition for cresting two ridge cells can be precisely stated:

FIGURE V.9



A CSQUARE in a network.
Each corner retinal cell may be classified as a ridge-cell and the triangles mark the positions of potential constructed ridge cells.

Two contiguous ridge cells R1, R2 can be <u>crested</u> iff

$\exists$ points C11, C12 $\in$ contacts(R1) and

$\exists$ points C21, C22 $\in$ contacts(R2)

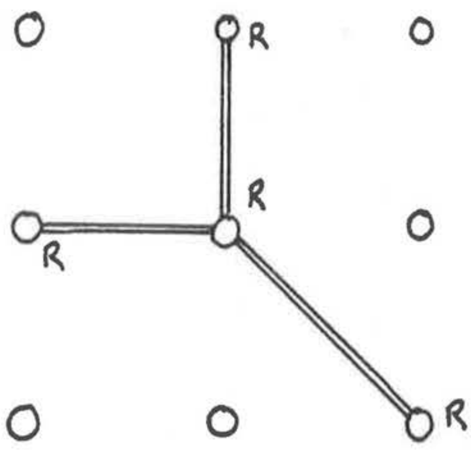
such that neighbourly(C11,C12) and

neighbourly(C21,C22).

The basic scheme for inserting crests and forming junctions is as follows. All csquares in the network are examined and the number of ridge cells in each determined. Those with less than two are ignored. If a csquare has two ridge cells then usually they will be crested although there are exceptions. In this way it is possible that one ridge cell may have crests to more than two other contiguous ridge cells; such a ridge cell is called a <u>junction cell</u>. If a csquare has three or more ridge cells some pairs of ridge cells will be crested and/or a set of three or more ridge cells may be grouped into a <u>junction set</u>. Examples of junction cells and junction sets appear in figure V.10. Now define a graph as follows. Every ridge-pt, junction-cell, or junction-set, is a vertex, and every crest is an edge of this graph. This is always a connected graph and is the <u>skeletal graph</u> of a connected region of a digital image.

Although there are 256 different csquare configurations this reduces to exactly 51 distinct cases after reflections and rotations are accounted for. A proof of this fact appears in appendix A.2, together with a listing of the 51 cases.

A note on implementation of the arithmetic operations should be made here. All quantities involved in the computation



Two junction cells.

Two junction sets.

FIGURE V.10

of the skeleton on an 8-connected network are in the form a + b*root2 where a, b are integers, that is to say they constitute an integral domain, so rather than use real arithmetic all arithmetic and comparison operations are done using integer arithmetic on pairs of integers (a,b). In other words Gaussian arithmetic is used.

V.2.6 Using parallelism to compute the skeleton

The above algorithm computes the skeleton in four raster scans, including one for forming junction sets and inserting crests. [Montanari,1968] gives an algorithm which proceeds by wave-front expansion in parallel and is equivalent to SKEL-1. The algorithm SKEL-3 can be modified to compute the skeleton in a similar parallel fashion.

V.2.7 Paths between objects and superfluous branches

With this new algorithm there is an edge of the skeletal graph emanating from every corner of a digital image. This is correct, in accordance with the definition of the skeleton. When the shapes involved have long straight lines aligned with the axes as borders, this is of no concern. But when the shapes are rotated slightly, many corners appear in a digitization and many superfluous edges appear in the skeletal graph. There are two ways to handle this problem, depending on the situation. If there are several isolated objects (an archipelago), and one is only concerned to find a route through the archipelago, then the only edges that need to be retained are those along which the skeleton points have contact points on distinct objects. I refer to these as _inter-object edges_, and the remaining edges as _internal edges_. The internal edges can be discarded and the search restricted to the inter-object edges. On the other hand, if one has to find paths from place to place within a single complex connected shape, all the edges are internal since the contact points of every skeleton point all lie on the same surrounding object. Various pruning strategies are available. If the contact points of an edge remain the same and are very close together, and the value of the quench function goes to zero (not a minimum greater than zero, as would occur in a passageway), then this edge may be removed. This handles the case of small steps introduced by the digitization of a line at approximately 45°. The threshold number used to decide when two contact points are "very close together" controls the size of bay that is represented in the skeletal graph. If one contact point of an edge remains constant while the distance from this contact point to the other contact point decreases, and the value of the quench function on the edge goes to zero, then this edge may be removed. This handles the case of an edge introduced into the skeleton by a small step in the digitization of a slightly inclined straight line.

V.3 Using the skeleton for path-finding

Use of the skeleton of the shape delineated on a network promised to overcome most of the objections to the use of heuristic search for pathfinding.

First assume that the start cell S and the destination cell D lie on the skeletal graph. Then a path from S to D along the skeletal graph is certainly a spatial path. In searching for a path through the skeletal graph, the junction cells and junction sets are the only places where a choice is needed. To simplify this search the pathgraph, homomorphic to the skeletal graph, is defined as follows. The vertices of the pathgraph correspond to the junctions (cells or sets) of the skeletal graph, and an edge between two vertices of the pathgraph corresponds to the chain of cells in the skeletal graph between the corresponding junctions. A skeletal graph and its corresponding pathgraph is shown in figure V.11. Now all the topologically distinct spatial paths from S to D are found by using a standard graph traversing algorithm on the pathgraph, with considerably less search than when the network of cells is searched directly.

If the start S or destination D are not on the skeletal graph then the nearest points to S and D on the skeletal graph, S' and D', must first be found. This can be done by following the fall-line at that point upwards until the skeletal graph is encountered. Then proceed as before. If it happens that several points on the skeletal graph are equally close to S or to D, then the pathfinding algorithm requires a trivial

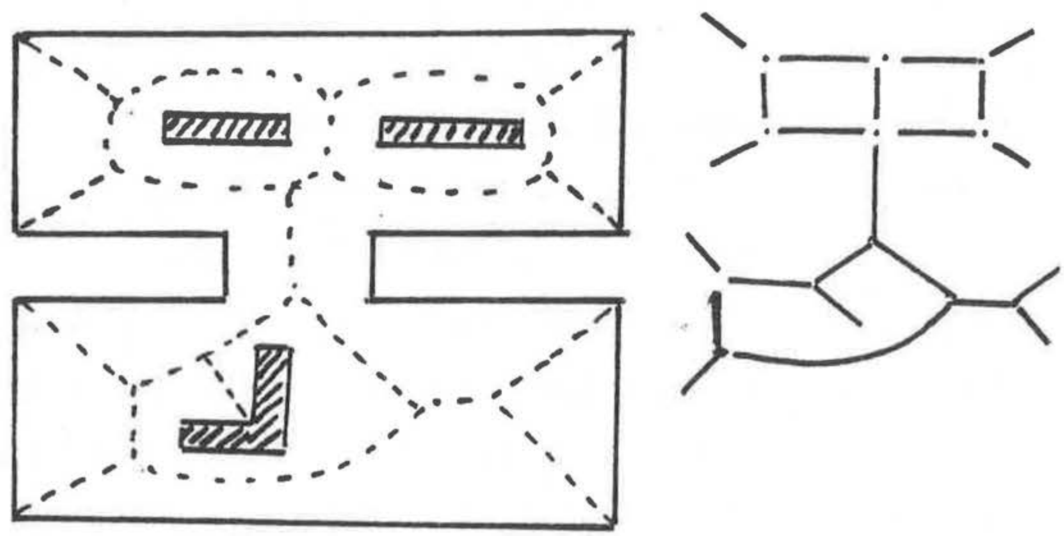


FIGURE V.11 A skeletal graph and its pathgraph.

modification.

V.3.1 Describing a skeletal path

A path specified by a path through the skeletal graph can be naturally described in terms of the objects of the world model. For instance in figure V.12 the path from S to D can be described as

(move ((keep-right-of ob1) & (keep-left-of cb2)) d1)

(turn right 45°)

(move ((keep-right-of cb3) & (keep-left-of cb2)) d2)

(move ((keep-right-of cb3) & (keep-left-of cb5)) d3)

(move ((keep-right-of ob4) & (keep-left-of cb5)) d4)

(turn right 60)

(move ((keep-right-of ob6) & (keep-left-of ob5)) d5).

Such a path description can be generated from the skeletal graph by examining the two contact points of each ridge-cell on the path. One belongs to the nearest object on the left hand side and the other belongs to the nearest object on the right hand side. So long as no two distinct objects on the screen run into each other, that is to say there is always a channel of spatial cells between distinct objects, then the membership of the contact cells of ridge cells can only change at junctions and therefore the description of ridge cells along any one edge of the pathgraph remains constant.

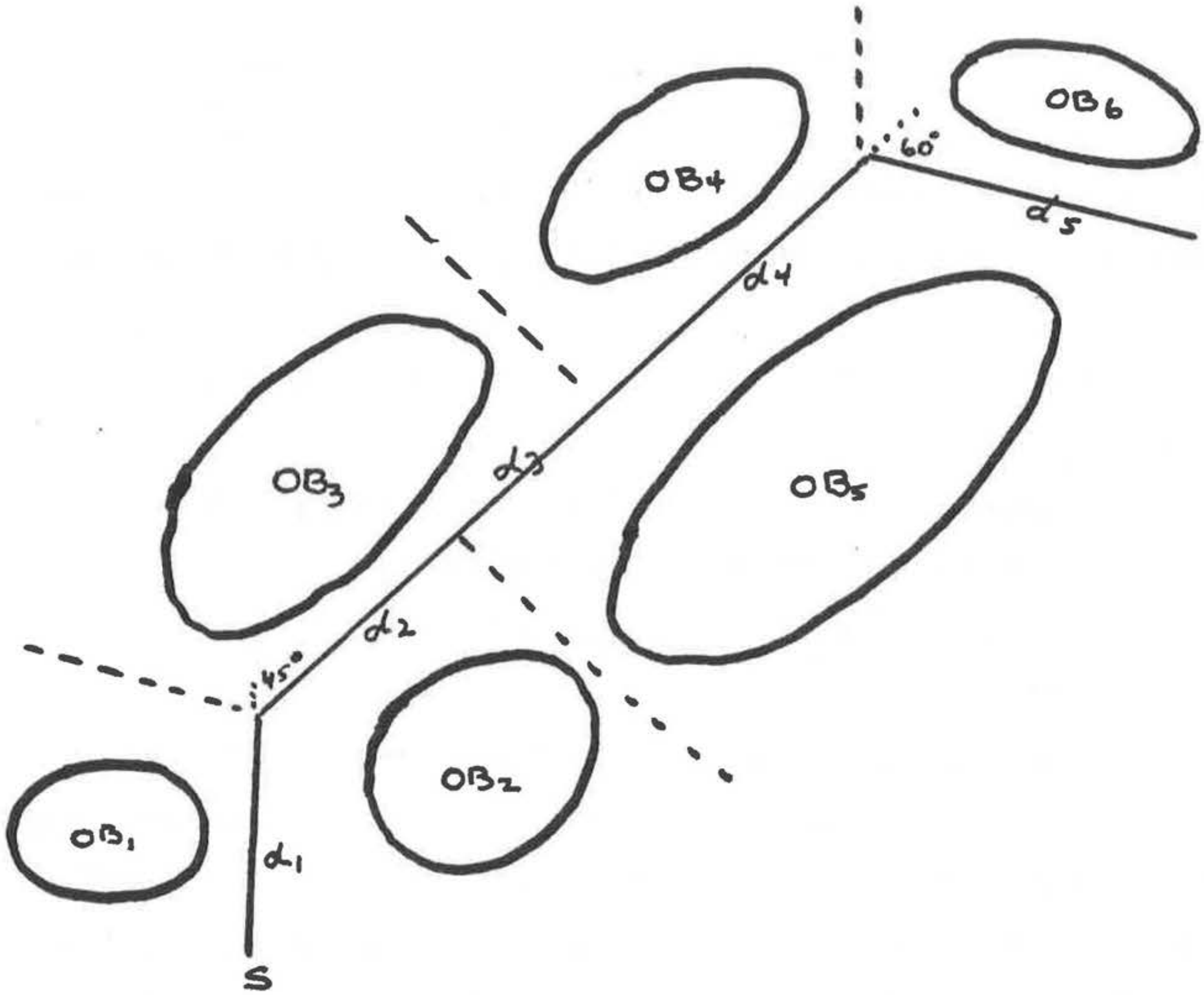


FIGURE V.12 Skeletal path between objects. The skeleton allows a "natural" description to be easily derived.

V.3.2 Optimizing a skeletal path

The example in figure V.13 shows that it is necessary to consider optimizing a skeletal path. If θ is the angle between consecutive corridors in the figure, then the length of the optimized path is sin(θ/2) times the length of the unoptimized skeletal path, a considerable improvement if θ is, say, 45° since in that case the optimized path is approximately 0.1 as long as the skeletal path.

In certain special cases, the optimization can be carried out roughly as follows. Let S,D be the start and destination of the skeletal path to be optimized, and define a function e on the skeletal path by e(p)=perpendicular distance from p on the skeletal path to the straight line SD. As p varies from S to D find the maximum of the function e(p)-q(p); if this is not positive nothing needs be done, for in this case the straight line SD is the shortest path from S to D. Otherwise let the maximum be at P* and apply this algorithm recursively to the paths SP* and P*D. Unfortunately this method cannot be carried out in general.

The problem here might be christened the rope-tightening problem, described as follows. Given an environment of arbitrary two dimensional shapes on a tabletop, two points S, D at vacant spots, and a rope laid out from S to D along a skeletal --or arbitrary-- path: compute a description of the curve assumed by the rope when tension is applied at S or D, any slack being taken up as required. Preferably the solution

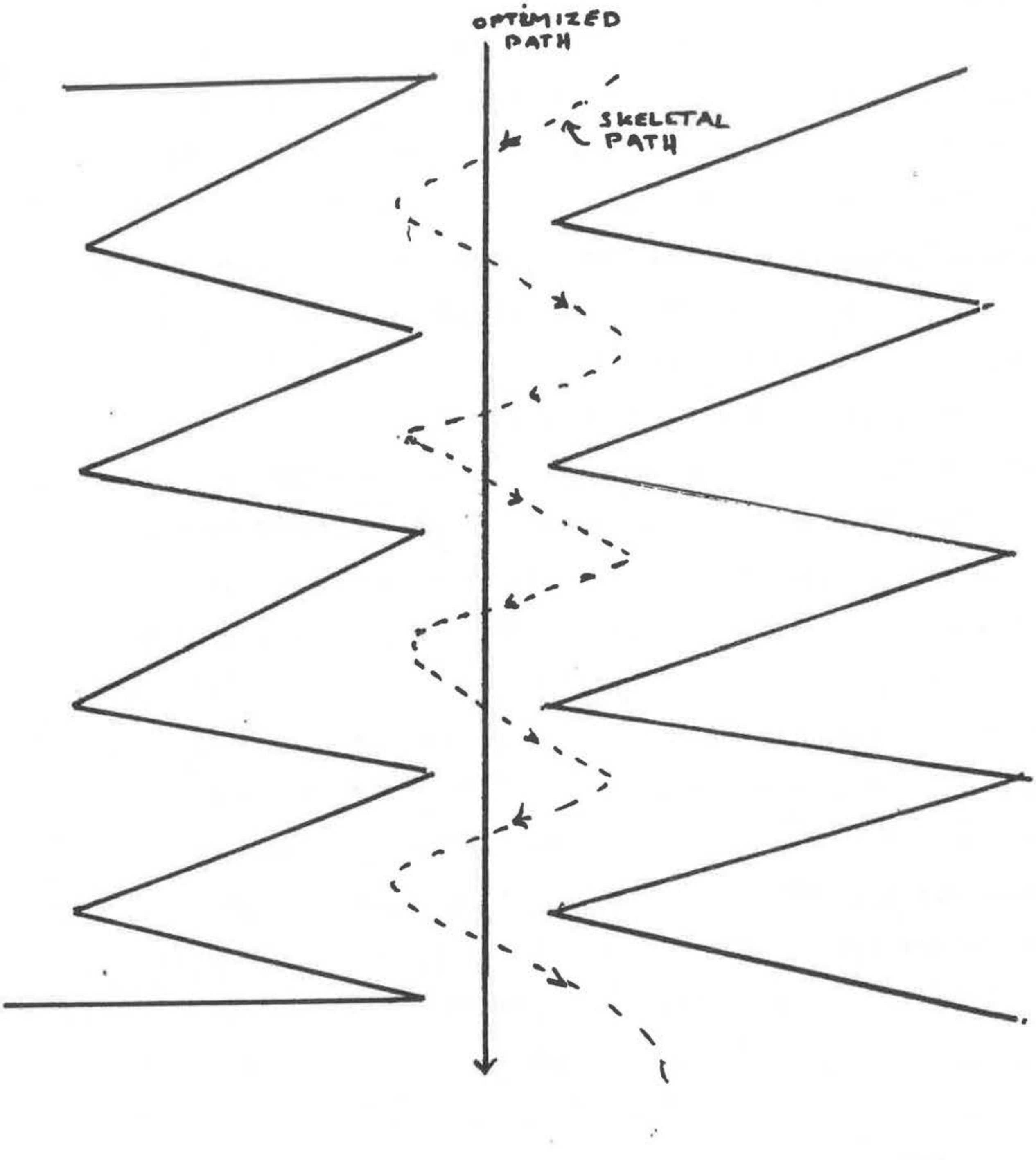


FIGURE V.13 Example illustrating the need to optimize a skeletal path.

should be stated in terms of an array of cellular automata.

One obvious strategy is to move the path in the direction of the centre of curvature at positions where the absolute value of the radius of curvature has a local minimum. The local minima of the radius of curvature correspond to the bends in the rope and the effect of moving such points inwards towards the local centre of curvature is to smooth out these bends. The radius of curvature at a point p of the path can be found by a local operator that looks at a small set of adjacent points of the path centred on p (say five neighbouring points). Then one pass along the whole path determines the local minima of the radius of curvature. If such a point is already adjacent to a point of an obstacle on the same side as the radius of curvature it cannot be moved any further. Otherwise each point B of local minimum is moved to the closest grid point B' in the direction of the centre of curvature. If the adjacent points to B in the path were A, C, then the adjacent points to B' in the new path remain A, C. The process is now repeated until either the path is straight, with infinite radius of curvature at all points, or else the only points of inflection occur where the path goes round an extreme point of an obstacle, the path being straight otherwise.

What I have just sketched is a relaxation algorithm for obtaining a curve whose second derivative is zero everywhere except for points where the curve contacts an obstacle. In effect it simulates the rope-tightening.

V. Path-finding and the skeleton of a planar shape

V.3.3 Comparison of skeletal and A* path finding

The complexity of both is linear in the number of cells in the network, except that the constant of linearity is much greater in the skeletal case. This is clear for A* since the number of nodes expanded by A* is bounded by the number of cells in the network. In the case of skeletal path-finding there are two parts to consider, finding the skeletal graph and searching the graph. The first operation is linear in the number of cells and in the second, the number of nodes expanded by a graph traversal algorithm is bounded by the number of cells. On the other hand a skeletal path has a natural description in environmental terms. If the use of parallelism is considered, then it is likely that the skeletal path-finding method comes out well. If the raster scan algorithm for the skeleton is replaced by a parallel wavefront-expanding method, it takes w wave-front expansions when the maximum value of the quench function is w. All the ridge-crest links can be computed in one parallel operation, except for some junction sets where two operations would be required.

V.4 Other applications of the skeleton

The skeleton can also be applied to finding a path for moving an object, finding empty space, and other problems. I describe each application in turn.

V.4.1 Object moving

Suppose there is an environment of obstacles and an object to be moved from one position to another. The simplest shape for which a path can be found by the skeleton is a circle.

V.4.1.1 Circular shaped object of radius r

First find the pathgraph of the empty space and remove from it any edge whose chain of cells in the skeletal graph contains a cell at which the value of the quench function is less than r. Then, if the initial and terminal positions of the circular shape are S and D, apply a heuristic graph traversal algorithm to the pathgraph to find the shortest path from S to D that follows the arms of the skeletal graph. Since the quench function along this path is everywhere at least as great as r, the circular shape can certainly be moved along this path provided the centre of the shape is kept on the path.

V.4.1.2 Other object shapes

The basic idea of this approach is to find the skeleton of the empty space, the skeleton of the shape to be moved, and work with the skeletons instead of the original shapes. The condition for a shape to be contained within a space can be stated in terms of the skeletons as follows:

(*) Let the quench function of the skeleton of the shape be q and the quench function of the skeleton of the empty space be r. Then for all points x of the skeleton of the shape, there must exist at least one point y on the skeleton of the space such that the circle centre x radius q(x) is contained in the circle centre y radius r(y).

Consider a long thin shape like a stick. By adding semi-circular ends if necessary, its skeletal graph is a straight line segment. Clearly if this line segment can be kept aligned with the skeletal graph of the space while the shape is being moved then condition (*) is easily checked, since then points of the shape's skeletal graph lie on the space's skeletal graph.

V.4.1.3 An L-shaped object

This is an awkward problem for humans at the best of times and in retrospect it was perhaps overly optimistic to think that a clean approach to its solution could be obtained through the use of skeletons. Although the skeleton does provide a clean approach to the problem of a circular shaped object, I have not yet found a useful application of it to the problem of moving more complex shaped objects.

The L problem can be viewed as requiring the simultaneous solution of two interacting subproblems. Namely, since each arm is a rectangular stick, first solve the problem of moving a stick though the doorway. Then, tackle the L problem by simultaneously solving two problems of moving a stick through the doorway, one from each arm of the L, with the complication that any movement of one stick causes a movement in the other.

V.4.2 Finding empty space

An interesting application of the skeleton is to the findspace problem: find space on a cluttered tabletop to put down another object. This is the FINDSPACE problem of Sussman [1973]. The position of the maximum sized circular spaces on the tabletop can be found directly from the quench function of the skeleton of the empty space. This is done by traversing the skeletal paths and finding the local maxima of the quench function. If the circle with radius equal to the maximum of the quench function is sufficient to contain the extra object then the problem is solved; if not, the positions of the maximal circles within the space are good candidates for the position of the extra object.

V.4.3 Finding the shortest distance between two shapes

Given two isolated planar shapes, a shortest straight line between them can be found by means of the skeleton in the following way. First find the skeleton of the empty space surrounding the shapes, and retain only those edges of the skeletal graph having one contact point on each shape. These are the <u>inter-shape edges</u>. Then, find a point P on the inter-shape edges at which the quench function takes its minimum value. The straight line joining the two contact points of P is then a shortest straight line between the two shapes.

Rather than developing the extensive mathematical machinery necessary to make the above description rigourous, I offer the following intuitive justification. At every point on an inter-shape edge one can draw a circular disc centred at that point, radius the quench function there, and with one contact point on each shape. As the inter-shape edges are traversed the disc expands and contracts in radius. The minimum value of the radius of this disc is assumed at one or more points of the inter-shape edges. At such a point the line connecting the contact points of the disc must be a diameter of the disc, and moreover is a line of shortest distance between the edges.

V.4.4 Finding nearest neighbourhood regions

Let P be a collection of points in the plane. The nearest neighbourhood of a point p ∈ P consists of all points x on the plane such that x is closer to p than to any other point in the collection P. The nearest neighbourhoods of a collection of points is also known as the Voronoi diagram or Dirichlet tessellation [Green & Sibson, 1978]. Consider the skeleton of the plane with point objects p, q, r,... corresponding to the points of P.

Lemma. The edges of the skeletal graph of the space surrounding the points P form the boundaries of the nearest neighbourhoods of the points of P.

Proof sketch. Every point of the plane is either equidistant from two or more points of P, or else is closer to one point than to any other point of P. In the first case the point lies on the skeletal graph of the space surrounding the points of P, in the second case the point is in the nearest neighbourhood of some point of P. Let x be a point in the skeletal graph of the space surrounding the points of P, with contact points p,q ∈ P. Thus x is not a vertex of the skeletal graph. In any arbitrarily small neighbourhood of x, there are points closer to p than to any other point of P, points closer to q than to any other point of P, and points of the skeletal graph. Hence x is a boundary point of the nearest neighbourhood of p and of the nearest neighbourhood of q. Similarly, if x is a vertext of the skeletal graph with contact points p,q,r,..., x is a boundary point of the nearest neighbourhoods of p,q,r,....

Thus the skeleton of a collection of shapes is a generalization of the boundaries of the nearest neighbourhood regions of a collection of points.

V.5 Summary

In this chapter I sketched various approaches to path-finding and proposed one based on the use of the skeleton of the shape of the empty space. In order to implement this latter approach I developed a new iterative algorithm for the skeleton that is guaranteed to compute a connected skeleton from a connected shape. I sketched how the skeleton could be used as a heuristic aid in the solving of object-moving problems, and showed how the skeleton could be applied to finding the shortest distance between objects and to finding nearest neighbourhood regions.

With this chapter on pathfinding I have specified a class of algorithms which can be used as a basis for the solution of pathfinding and object moving problems in the spatial planner. Thus the outline design of the robot-controller is complete.

CHAPTER VI

SUMMARY, CONCLUSION, AND FUTURE WORK

VI.1 Summary

After an introductory chapter, I reviewed at some length the nature of Artificial Intelligence, introduced the approach of simulating a made up organism, and reviewed several closely related pieces of work. Then I described in some detail a system for simulating an environment and the sensori-motor parts of an organism to inhabit it. In chapter IV I sketched the overall features of the control program of such an organism, basing it on the notion of the action cycle. One partially implemented design was described and an alternative approach proposed. An important requirement for any organism, in particular for its controller, is a path-finding ability. To this end I described in chapter V an approach to path-finding based on the skeleton of a shape. The skeleton, which was originally motivated by physiological considerations and first applied to shape description, is also a useful tool for path-finding and as a heuristic for other spatial problems. I developed a new iterative algorithm to compute it. Although the actual amount of computation to find the skeleton on a serial machine may be greater than that for other techniques for path-finding, it reduces the heuristic search required for finding a route between obstacles.

So let me stand back and take stock. What has been solved and what problems uncovered? How does this work relate to other work? What contribution does it make to the Artificial Intelligence enterprise? The advances contained in this work are threefold.

(a) In the first place I explicitly laid out the features of the action cycle for a robot-controller; this has not been done before. The implementation done thus far was based on the analogy between the process of perception and the scientific method whereby one is always acting on the basis of a collection of hypotheses and gathering evidence for these hypotheses in the form of sensory input data.

(b) A spatial reasoning module is an important and essential part of any robot-controller. It makes plans for action on the basis of the current collection of hypotheses about the form of the environment. The second advance is the development of a new approach to problems of spatial reasoning based on the use of the skeleton of a two-dimensional shape. When the environment has been drawn on an array of points like a screen, an iterative algorithm requiring a constant amount of computation reduces any pathfinding problem to a simpler graph-traversal problem. Each edge of the graph corresponds to a path between two objects, each node corresponds to a junction of three or more paths, while the number of nodes is reduced to a minimum. Thus the amount of heuristic search is reduced. There are other ways to do this that are based on a Cartesian representation of the shapes of objects, which will require much more search if the shapes in the environment have much extraneous detail. The method presented here could clearly be computed by a neuronal network. Thus it is interesting to ponder if this is one of the algorithms actually used by the functioning mammalian brain.

Some interesting technical problems were uncovered in this approach. One is called the rope-tightening problem. When you have found one reasonable path between two points, how do you tighten the path to the shortest possible way? The other technical problem relates to elucidating the full details for moving an L-shaped object through a doorway. I presented one suggestion for approaching the solution to this problem.

(c) As discussed in chapters II and III there have been several robot simulation programs written before and the full details of my robot simulation were described there. Mine is the first to handle the movement and collision of two dimensional shapes. Of the previous two major robot simulations, Becker & Merriam used a Cartesian representation and Nilsson & Raphael used a digital representation of shapes. The Cartesian represents shapes as a series of points given by Cartesian coordinates while the digital represents a shape directly as an array of points like a screen. Of the previous rigid object motion simulations, Eastman and Pfefferkorn used Cartesian representations while Funt and Baker used digital representations. My advance was to use a combination of the Cartesian and the digital representations to simulate the motion and collision of objects on a tabletop.

VI.2 Conclusion

I began the thesis by asking what computational processes are required for spatial reasoning. My answer, and, briefly, the conclusion of the thesis, is this. Computational processes incorporating algorithms for computing the digital skeleton of a planar shape may prove to be sufficient for the spatial reasoning of a robot-controller.

VI.3 Research problems

This consists of a list of problems encountered in the course of our project, that need further investigation.

1. Problems related to the simulated environment.

(a) Extend TABLETOP to compute exact collision points between robot or object and an obstacle. Two methods for doing this were described in III.3, which should be implemented and tested.

(b) Simulate parallel operating hardware to carry out the TABLETOP simulation.

(c) Generalize the TABLETOP simulation to three dimensions.

(d) Design a more interesting environment. Trivially, this can be done, for example, by allowing randomized action effects; less trivially, by allowing independently moving objects. Since any extension of the environment requires a corresponding increase in the capabilities of the organism-controller, no such extension should be contemplated until the current environment is competently handled by the current organism-controller.

2. Problems related to the organism-controller.

(a) Allow a restricted form of natural language input for the task statement.

(b) Design a more 'realistic' form of vision.

(c) Implement the spatial planner. In particular, extensive experimentation with the L-shaped object problem is required.

3. Problems related to the skeleton.

(a) Implement skeleton algorithms that use 16-connected cells rather than 8-connected cells, and compare their performance with the algorithm for 8-connected networks and with true Euclidean skeletons.

(b) Extend the skeleton algorithms to apply to three dimensions. The short definition of the 3D skeleton is "the locus of the centre of maximal spheres", and in general the 3D skeleton is a surface not a line. It may, however, be of some use in planning the movement of objects in three dimensions.

APPENDICES

A.1 TABLETOP user's manual

Run LISP, then type the following to bring up the TABLETOP system:

```
(DISKIN BSR1:BASIC RSR1:SBW#LISP RSR1:ENVIRONMENT#5 COH4:SBW#OH)
(OHSENSE)
```

A snapshot of the environment is then displayed on the screen, and the bug can now be controlled by a small number of commands. After each command is given the retinal impression and tactile impression of Utak are displayed.

In the following commands, "distance" is a positive or negative number with a decimal point. "orientation" is a compass direction (one of N,NE,E,SE,S,SW,W,NW) or a positive or negative number with a decimal point. A zero number means north, a positive number means an angle measured clockwise from north, and a negative number means an angle measured anti-clockwise from north. "radians" is a positive or negative number, or one of the angles PI or PI/2. "degrees" is a positive or negative number. For both "radians" and "degrees", a positive number means a clockwise turn and a negative number means an anti-clockwise turn.

(SLIDE distance orientation) The bug moves approximately "distance" units in the direction "orientation".

(HOLD) if the bug is immediately adjacent to a movable object then after this command is executed the bug and the adjacent object are cemented together and move as one rigid object. The PUSHTO and TURN commands must now be used.

(LETGO) Undoes the effect of HOLD. After this command the bug can move freely again, by means of the SLIDE command.

(PUSHTO distance orientation) The bug and the object held move as one rigid unit approximately "distance" units in the approximate direction "orientation".

(TURN radians) The bug and the object held turn as one rigid unit through an angle of approximately "radians" radians.

(TURND degrees) The bug and the object held turn as one rigid unit through an angle of approximately "degrees" degrees.

(WORLD integer) This sets up a new environment for the bug. "integer" must lie between 0 and 8. These are predefined environments; there are also facilities for setting up an environment directly using the functions START-SRW, CREATE-OBJECT, and PUTPUSHER. Examples of their calling sequences can be found in

BSR1:ENVIRON#5.

A.2 A combinatorial lemma

A csquare is defined as a square with eight locations on it, one at each corner and one at the midpoint of each side. Each location may be occupied or vacant. Thus the set S of all csquares contains $2^8=256$ members. Now consider the group G of rotations and reflections of a square into itself. G has eight elements, consisting of the identity, three rotations, and four reflections. Each element of G acts as a permutation of the set S. Two elements of s1, s2 of S are equivalent if there is a group element g such that gs1=s2. This is an equivalence relation that divides S into a number of equivalence classes.

Lemma. The number of equivalence classes of csquares under the group G of rotations and reflections of a csquare is 51.

Proof. By Burnside's lemma (see for example, [de Bruijn,1964,p. 150]) the number of equivalence classes is given by

$$1/|G| \quad psi(g)$$

where |G| denotes the number of elements of G, and, for each g, psi(g) denotes the number of elements of S that are invariant under g, that is, the number of $s \in S$ for which gs=s.

For the group G of reflections and rotations of a square, |G|=8. G={I,R1,R2,R3,RR1,RR2,RR3,RR4} where I is the identity, Ri,i=1,2,3 are the rotations, and RRj,j=1,2,3,4 are the reflections. Then one has psi(I)=256; psi(R1)=psi(R3)=4;

psi (R2) = 16; psi (RR1) = psi (RR2) = psi (RR3) = psi (RR4) = 32. Thus the number of equivalence classes is

1/8{256+4+16+4+4*32}=408/8=51. QED.

One representative from each equivalence class is shown in figure A2.1.

Figure A2.1

```
0 . .   . * .   0 * .   0 . 0   0 . .   . * .   0 . .
.   .   .   .   .   .   .   .   .   *   .   *   .   .
. . .   . . .   . . .   . . .   . . .   . . .   . . 0

. * .   0 * 0   0 * .   0 . 0   . * 0   0 * .   0 . 0
.   .   .   .   .   *   .   *   .   *   .   .   .   .
. * .   . . .   . . .   . . .   . . .   . . 0   . . 0

0 * .   0 . 0   0 . .   . * .   0 * 0   0 * 0   0 * .
.   .   .   .   .   *   .   *   .   *   .   .   .   *
. * .   . * .   . * .   . * .   . . .   . . 0   . . 0

0 * 0   0 * .   0 . 0   . * 0   0 * .   0 . 0   . * 0
.   .   .   *   .   *   .   *   .   .   .   .   .   .
. * .   . * .   . * .   . * .   . * 0   . * 0   . * 0

0 . .   0 . 0   . * .   0 * 0   0 * 0   0 * 0   0 * .
.   *   .   .   *   *   .   *   .   *   .   .   .   *
. * 0   0 . 0   . * .   . . 0   . * .   . * 0   . * 0

0 . 0   . * 0   0 * 0   0 * .   0 * .   0 * 0   0 * 0
.   *   .   *   .   .   .   *   *   *   .   *   .   *
. * 0   . * 0   0 . 0   0 . 0   . * .   . * 0   0 . 0

0 * 0   0 * 0   0 * 0   0 * .   0 * 0   0 * 0   0 * .
.   *   .   .   *   *   *   *   .   *   *   *   .   *
0 * .   0 * 0   . * .   . * 0   0 * 0   . * 0   0 * .

0 * 0   . . .
*   *   .   .
0 * 0   . . .
```

A.3 On Funt's rigid shape rotation algorithm

Let a situation be an arrangement of squares on the WHISPER array and let a rotation be an ordered pair {centre of rotation, angle of rotation}. A rotation transforms a situation into a new situation. This is defined precisely as follows. Suppose a rotation rho={C,alpha} is used to rotate situation SIT1 into a new situation SIT2. The transformation is carried out by rotating each square Q in SIT1 independently. The image square R of Q after rotation rho is computed by the following method. Let P be the centrepoint of square Q. Now with centre C and radius CP, rotate P by amount alpha to a new position P' and determine which square of the array contains P'. This is R, the image square of Q under rho. Note that the distance between R and P' may have any value up to root2/2. The transformed situation SIT2 consists of the set of all image squares under rho.

The simplest examples illustrating why the depiction of an object on the array disintegrates are shown in figure A3.1 . Note that the distance between the centres of two edge-adjacent squares is one whereas the distance between the centres of two diagonally corner-adjacent squares is root2. Thus the centres of two edge-adjacent squares, when rotated by 45°, can map into the same square. This is a merge of two squares into one. Similarly, the centres of two corner-adjacent squares can map

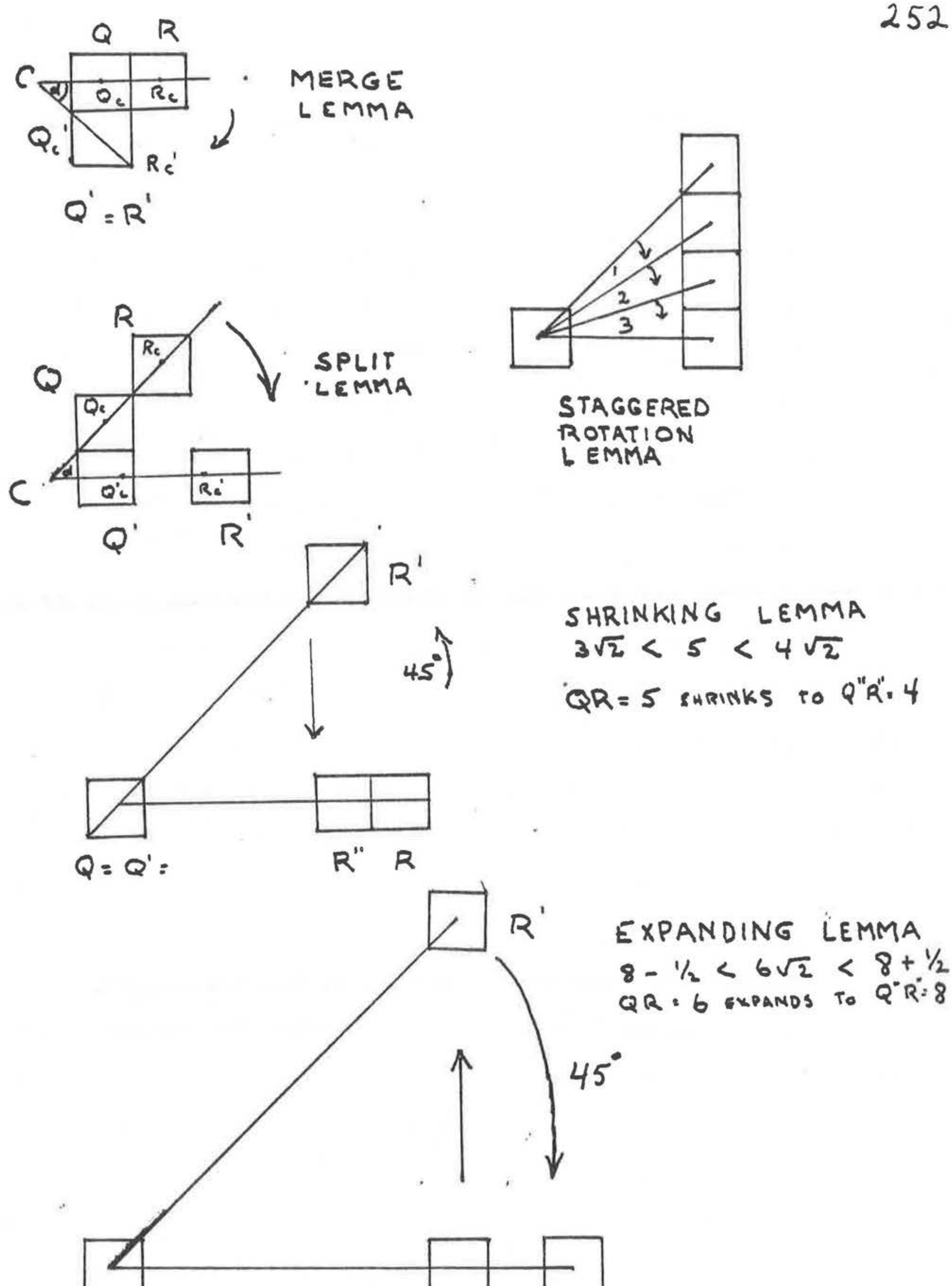
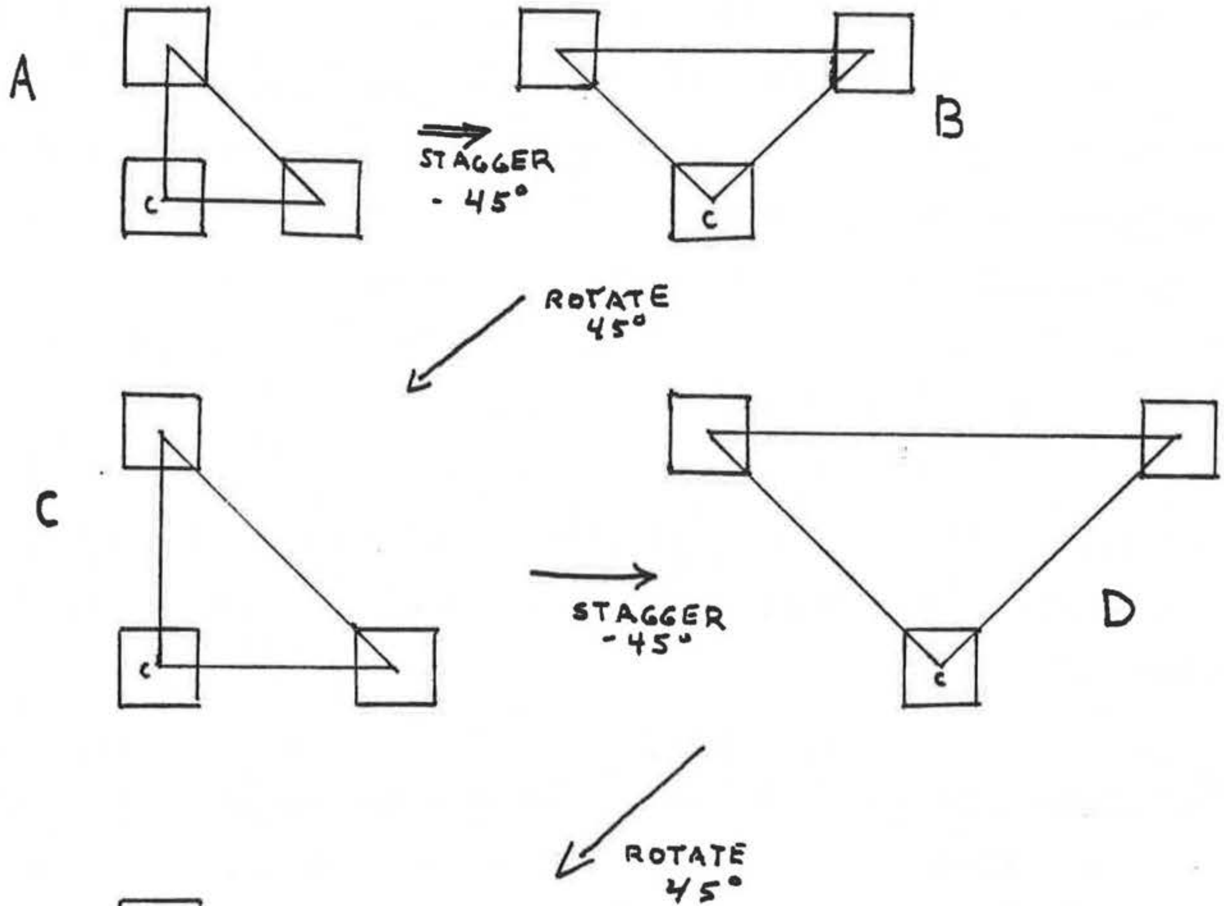
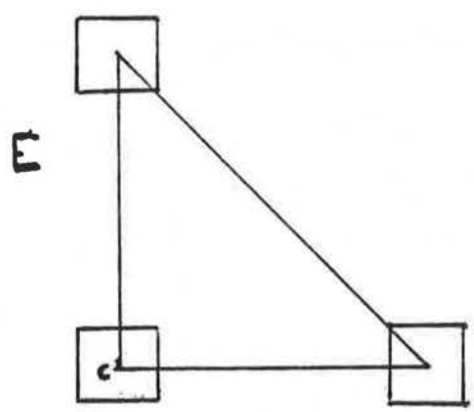

MERGE LEMMA
Q' = R'
SPLIT LEMMA
STAGGERED ROTATION LEMMA
SHRINKING LEMMA
3√2 < 5 < 4√2
QR = 5 SHRINKS TO Q"R" = 4
45°
Q = Q' =
R" R
EXPANDING LEMMA
8 - ½ < 6√2 < 8 + ½
QR = 6 EXPANDS TO Q"R" = 8
45°
Q = Q' = Q"
R R"

FIGURE A3.1





SPREADING THEOREM.

The axis distance between the squares in situation A is 2, the axis distance between the squares in situation E is 4.

FIGURE A3.1 (continued)

into two non-edge-adjacent squares in a row. This is a split between two previously adjacent squares. Given an initial situation and a sequence of rotations, the final situation is the situation arising after all the rotations of the sequence have been applied in succession to the initial situation. Let the centre-line of two squares be the line joining their centres. Let a n-square situation be a situation with exactly n squares. Then the following lemmas hold.

Merge lemma. For any two-square situation there is a sequence of rotations such that the final situation is a one-square situation.

Split lemma. For any two-square situation and arbitrarily large number X, there is a sequence of rotations such that the final situation is a two-square situation and the distance between the centres of the squares is at least X.

Three further lemmas are needed to prove the split/merge lemmas.

Staggered rotation lemma. In a two-square situation let the centre-line of the two squares lie at some angle between the horizontal and 45°. Let the horizontal distance between their centres be n. Then two sequences of rotations can be found, each of which keeps the horizontal distance between the centres of the squares equal to n. One has property (a) and one has property (b) in the corresponding final situations.

(a) The centre-line of the two squares is horizontal.

(b) The centre-line of the two squares makes an angle of 45° with the horizontal.

The staggered rotation lemma essentially says that if two squares have a 45° centre line and are separated by n intermediate squares ($n \geq 0$), then they can be rotated in a staggered fashion so that they have a horizontal centre line but are still separated by exactly n intermediate squares; and conversely. To be more precise, suppose that in a two-square situation the coordinates of one square relative to the other are (ix,iy). Then the <u>axis distance</u> between the squares is m = Max{|ix|,|iy|}. The staggered rotation lemma then says that any two-square situation can be transformed into any other in which the axis distance between the squares is the same.

<u>Shrinking lemma</u>. Given a two-square situation with axis distance n between the squares, and suppose

(m - 1)*root2 < n < m*root2

holds for some integer m > n. Then there exists a sequence of rotations such that in the final situation the axis distance between the squares is m.

(Proof: one 45° rotation plus one staggered rotation.)

The proof of the merge lemma now follows by alternate applications of the shrinking lemma and the staggered rotation lemma.

Expanding lemma. Given a two-square situation with axis distance m between the squares, and suppose

$$n - 1/2 < m*\mathrm{root2} < n + 1/2$$

holds for some integer $n > m$. Then there exists a sequence of rotations such that in the final situation the axis distance between the squares is n.
(Proof: one staggered rotation plus one 45° rotation.)

The proof of the split lemma now follows by alternate applications of the expanding lemma and the staggered rotation lemma. Successive applications of the merge lemma prove the following

Collapsing theorem. Given any initial situation there is a sequence of rotations such that the final situation contains only one square.

Conversely, the question is whether the split lemma can be generalized to multisquare situations. The answer is yes.

Spreading theorem. Given any situation with S squares and given an arbitrarily large number X, there is a sequence of rotations such that in the final situation there are still S squares and the distance between any two squares is at least X.
Proof (outline). Pick a pair of squares with minimum axis

distance and take the centre of one of these as the centre of rotation for all following rotations. First jiggle each square in turn until all the squares lie on either of the two diagonal lines through the centre of rotation. No merges must be allowed to occur in this jiggling. An individual square can always be moved without moving any other squares, by picking a centre of rotation closer to that square than any other. In particular, 'compact' sets of squares, as would occur in the original depiction of an object, can be split up without any merges. Now rotate 45° so that the diagonal lines are in the horizontal/vertical position, then carefully stagger back to the diagonal position. The 45° rotation does the splitting, the staggering regains the standard position. Repeat this until sufficient spreading has occurred.

The collapsing and spreading theorems are enough to show that Funt's object rotation scheme cannot work.

REFERENCES

[Amarel,1968]
Amarel,S. On representations of problems of reasoning about actions. In: D.Michie (ed.), Machine Intelligence 3, Edinburgh University Press, 1968.

[Anderson,1978]
Anderson,J.R. Arguments concerning representations for mental imagery. Psychological review, 85(4) (July 1978), pp.249-277.

[Arbib & Lieblich,1977]
Arbib,M.A., and Lieblich,I. Motivational learning of spatial behaviour. In: J.Metzler (ed.), Systems Neuroscience, Academic Press, New York, 1977, pp.221-239.

[Baker,1973]
Baker, Richard. A spatially oriented information processor which simulates the motions of rigid objects. Artificial Intelligence, 4 (Spring 1973), pp.29-40.

[Barrow & Tenenbaum,1978]
Barrow,H.G., and Tenenbaum,J.M. Recovering intrinsic scene characteristics from images. In: Hanson,A.R, and Riseman,E.M. (eds.), Computer Vision Systems. Academic Press, New York, 1978, pp.3-26.

[Bartlett,1932]
Bartlett,F.C. Remembering. Cambridge University Press, 1932.

[Becker,1972]
Becker, J.D. "Robot" computer problem solving system. Report 2316, Bolt, Beranek & Newman Inc., 1972.

[Becker,1973]
Becker, J.D. "Robot" computer problem solving system. Report 2646, Bolt, Beranek & Newman Inc., 1973.

[Becker,1973]
Becker,J.D. A model for the encoding of experiential information. In: R.C.Schank and K.M.Colby, Computer models of thought and language, W.H.Freeman & Co., 1973.

[Becker & Merriam,1973]
Becker,J.D., and Merriam,W. "Robot" computer problem-solving system. Report 2646, Bolt, Beranek & Newman Inc., 1973.

[Bindra,1976]
Bindra,D. A theory of intelligent behaviour. John Wiley and sons, 1976.

[Bitterman,1975]
Bitterman,M.E. The comparative analysis of learning. Science, 188 (1975), pp.699-709.

[Black,1968]
Black,F. A deductive question-answering system. In: M.Minsky (ed.), Semantic Information Processing, MIT Press, 1968.

[Blum,1967]
Blum,H. A transformation for extracting new descriptors of shape. In: W.Walthen-Dunn (ed.), Models of the perception of speech and visual form. MIT Press, 1967, pp.362-380.

[Blum,1973]
Blum, Harry. Biological shape and visual science (Part I). J. Theoretical Biology, 38(2) (1973), pp.205-287.

[Blum,1974]
Blum,H. A geometry for biology, in: Mathematical Analysis of Fundamental Biological Phenomena, in: Annals New York Academy of Sciences 231, pp.19-30.

[Bower,1977]
Bower,T.G.R. A primer of infant development. W.H.Freeman, 1977.

[Brindley,1969]
Brindley,G.S. Nerve net models of plausible size that perform simple learning tasks. Proc. Roy. Soc. London series B, 174 (1969), pp.173-191.

[Burks,1978]
Burks,A.W. Computer science and philosophy. Logic of computers group tech. report 218, Computer science department, U.Michigan, 1978.

[Calabi & Hartnett,1968]
Calabi,L., and Hartnett,W.E. Shape recognition, prairie fires, convex deficiencies, and skeletons. American Math. Monthly, 75 (April 1968), pp.335-342.

[Chien & Weissman,1975]
Chien, R.T., and Weissman,S. Planning and execution in incompletely specified environments. Proceedings Fourth International Conference on Artificial Intelligence, 1975, pp.169-174.

[Coles et al.,1975]
Coles, L.Stephen, A.M.Robb, P.L.Sinclair, M.H.Smith, R.B.Sobeck. Decision analysis for an experimental robot with unreliable sensors. Proceedings Fourth International Conference on Artificial Intelligence, 1975, pp.749-757.

[Cook & Reckow,1974]
Cook,S., and Reckow,R. On the lengths of proofs in the propositional calculus. Proceedings 6th Annual Symposium on Theory of Computing, ACM, 1974, pp.135-148.

[Davis,1963]
Davis,M. Eliminating the irrelevant from mechanical proofs. Proc.symposia in applied mathematics, 15 (1963), pp.15-30.

[de Bruijn,1964]
de Bruijn,N.G. Polya's theory of counting. In: E.F.Beckenbach, Ed., Applied Combinatorial Mathematics, John Wiley, New York, 1964.

[Dennett,1978a]
Dennett,D.C. Commentary "Why not the whole iguana?". The Behavioural and brain sciences 1(1) (1978), pp.103-104.

[Dennett,1978b]
Dennett,D.C. Brainstorms. Bradford Books, 1978.

[Donaldson,1978]
Donaldson,M. Children's minds. Fontana, 1978.

[Doyle,1978]
Doyle,J. Truth maintenance systems for problem-solving. TR-419, MIT AI Lab., 1978.

[Eastman,1973]
Eastman, Charles M. Automated space planning. Artificial Intelligence, 4(1) (1973), pp.41-64.

[Eccles et al.,1977]
Eccles,M.J., McQueen,M.P.C. And Rosen,D. Analysis of the digitized boundaries of planar objects. Pattern Recognition, 9(1) (January 1977), pp.31-41.

[Ermentrout & Cowan,1978]
Ermentrout,G.B. And Cowan,J.D. Large scale activity in neural nets, SIAM Journal of Applied Math.,1978.

[Fahlman,1974]
Fahlman, Scott E. A planning system for robot construction tasks. Artificial Intelligence, 5(1) (1974), pp.1-49.

[Feldman & Sproull,1977]
Feldman,J.A., and Sproull,R.F. Decision theory and AI II: the hungry monkey. Cognitive Science, 1 (1977), pp.158-192.

[Fikes & Nilsson,1971]
Fikes, R.E., and Nilsson,N.J.. STRIPS: A new approach to the application of theorem-proving to problem solving. Artificial Intelligence, 2 (1971), pp.189-208.

[Fikes, Hart, & Nilsson,1972]
Fikes,R.E., Hart,P.E., and Nilsson,N.J. Learning and executing generalized robot plans. Artificial Intelligence, 3 (251-288 1972), pp..

[Freeman,1974]
Freeman,H. Computer processing of line-drawing images. Computing Surveys, 6(1) (March 1974), pp.57-97.

[Friedman,1967]
Friedman,L. Instinctive behaviour and its computer synthesis. Behavioural Science, 12(2) (March 1967), pp.85-108.

[Friedman,1977]
Friedman, Leonard. Robot perception learning. Proc. Pattern directed inference systems workshop, SIGART Newsletter June(1977), pp.44-49.

[Funt,1976]
Brian V. Funt. WHISPER: A computer implementation using analogues in reasoning. Technical report no. 76-09, Department of Computer Science, University of British Columbia, 1976.

[Gaschnig,1978]
Gaschnig,J. Experimental studies of backtrack vs. Waltz-type vs. new algorithms for satisficing assignment problems. Proc.second national conference CSCSI(1978), pp.268-277.

[Gass et al.,1976]
Gass,C.L., Angehr,G., and Centa,J. Regulation of food supply by feeding territoriality in the rufous hummingbird. Can. J. Zool. 54, pp.2046-2054.

[Gelernter,1963]
Gelernter,H. Realization of a geometry-theorem proving machine. In: E.A.Feigenbaum and J.Feldman (eds.), Computers and Thought, McGraw-hill,1963.

[Gilmore,1960]
Gilmore.P.C. A proof method for quantification theory. IBM Journal for research and development, 4 (1960), pp.28-35.

[Good,1965]
Good,I,J. Speculations concerning the first ultraintelligent machine. In: F.L.Alt and M.Rubinoff (eds.), Advances in computers, Vol.6, Academic Press, pp.31-88, 1965.

[Green,1969]
Green,C. Application of theorem-proving to problem solving. IJCAI-69 (1969), pp.219-237.

[Green & Sibson,1978]
Green, P.J., and Sibson,R. Computing Dirichlet tessellations in the plane. Computer Journal, 21(2) (June 1978), pp.168-173.

[Hart,Nilsson,Raphael,1968]
Hart,P., Nilsson,N., and Raphael,B. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Sys. Sci. Cybernetics,SSC-4, 2 (1968), pp.100-107.

[Hebb,1949]
Hebb, D.O. The Organization of Behaviour: a Neuropsychological Theory. John Wiley and Sons Inc., New York, 1949.

[Hebb,1968]
Hebb,D.O. Concerning imagery. Psychological Review, 75 (1968), pp.466-477.

[Hilbert,1927]
Hilbert,D. The foundations of mathematics. Reprinted in: J.Van Heijenoort (ed.), From Frege to Godel, Harvard University Press, 1967.

[Howden,1968]
Howden, W.E. The sofa problem. Computer Journal, 11 (May to November 1968), pp.299-301.

[Jacobs & Kiefer,1973]
Jacobs,W., and Kiefer,M. Robot decisions based on maximising utility. Proceedings Third International Conference on Artificial Intelligence, 1973, pp.402-410.

[Jerison,1973]
Jerison,H.J. The evolution of the brain and intelligence. Academic Press, 1973.

[Kandel,1976]
Kandel,E.R. Cellular basis of behaviour. W.H.Freeman, 1976.

[Kandel,1979]
Kandel,E.R. Behavioural biology of aplysia. W.H.Freeman, 1979.

[Kleene,1956]
Kleene,S.C. Representation of events in nerve nets and finite automata. In: E.C.Shannon and J.McCarthy (eds.), Automata studies, Princeton U.P.

[Kohler,1925]
Kohler,W. The mentality of apes. Translated by E.Winter and republished 1976 by Liveright, New York.

[Kosslyn & Pomerantz,1977]
Kosslyn,S.M., and Pomerantz,J.R. Imagery, propositions, and the form of internal representations. Cognitive Psychology, 9 (1977), pp.52-76.

[Kowalski,1969]
Kowalski,R. Search strategies for theorem-proving, B.Meltzer and D.Michie (eds.). Machine Intelligence 5, Edinburgh University Press, 1970, pp.181-201.

[Kuhn,1962]
Kuhn,T. The structure of scientific revolutions. University of Chicago Press, 1962.

[Kuipers,1977]
Kuipers, Benjamin J. Representing knowledge of large-scale space. Ph.D. Thesis, AI-TR-418, MIT AI lab., 1977.

[Levi & Montanari,1970]
Levi, G., and U.Montanari. A gray-weighted skeleton. Information and Control, 17 (1970), pp.62-91.

[Ludlow,1976]
Ludlow,A.R. The behaviour of a model animal. Behaviour, 58 (1976), pp.131-172.

[McCarthy,1968]
McCarthy,J. Programs with common sense. In: M.Minsky(ed.), Semantic Information Processing. MIT Press, 1968.

[McCarthy,1977]
McCarthy,J. Epistemological problems of AI, Proceedings Fifth International Conference on Artificial Intelligence, 1977, pp.1038-1044.

[McCarthy & Hayes,1969]
McCarthy,J., and Hayes,P.J. Some philosophical problems from the standpoint of artificial intelligence, B.Meltzer and D.Michie (eds.). Machine Intelligence 4, Edinburgh University Press, 1969, pp.463-502.

[McCarthy et al.,1978]
McCarthy,J., Hayashi,T., and Igarashi,S. On the model theory of knowledge. AIM-312, Stanford AI Lab., 1978.

[Mackworth,1977]
Mackworth,A.K. On reading sketch maps, Proceedings Fifth International Conference on Artificial Intelligence, 1977, pp.598-606.

[Marr,1976]
Marr,D. Early processing of visual information. Phil.Trans.Roy.Soc.London, 275(942) (October 1976), pp.483-524.

[Marr,1977]
Marr, D. Artificial Intelligence - a personal view. Artificial Intelligence, 9(1) (August 1977), pp.37-48.

[Marr & Poggio,1976]
Marr,D., and Poggio,T. From understanding computation to understanding neural circuitry. AIM-357, MIT AI lab., 1976.

[Marr & Poggio,1976]
Marr,D., and T.Poggio. Cooperative computation of stereo disparity. Science, 194 (15 October 1976), pp.283-287.

[Martelli,1977]
Martelli,A. On the complexity of admissible search algorithms. Artificial Intelligence, 8 (1) (February 1977), pp.1-13.

[Masterman,1970]
Masterman,M. The nature of a paradigm. In: I.Lakatos and A.Musgrave (eds.), Criticism and the growth of knowledge, Cambridge U.P., pp.59-89,1970.

[Merriam,1975]
Merriam, E. W. An experimental robot computer problem solving system. Report 3108, Bolt, Beranek & Newman Inc., 1975.

[Michie,1978]
Michie,D. Seminar at UBC, fall 1978.

[Miller,1978]
Miller,L. Has AI contributed to an understanding of human mind?. Cognitive Science, 2 (1978), pp.111-127.

[Miller,1974]
Miller,G.A. Needed: a better theory of cognitive organization. IEEE Trans. On Systems, Man, and Cybernetics, SMC-4 (1) (January 1974), pp.95-97.

[Miller,Galanter, & Pribram,1960]
Miller,G.A., Galanter,E., and Pribram,K.H. Plans and the structure of behaviour. Holt,Rhinehart and Winston, 1960.

[Minsky & Papert,1972]
Minsky,M., and Papert,S. AI Progress Report. AIM-252, MIT AI Lab., 1972.

[Minsky,1975]
Minsky,M. A framework for representing knowledge. In: P.H.Winston (ed.), The psychology of computer vision. McGraw-Hill,1975.

[Montanari,1968]
Montanari, U. A method for obtaining skeletons using a quasi-euclidean distance. Journal of the ACM, 15 (4) (October 1968), pp.600-624.

[Montanari,1969]
Montanari, U. Continuous skeletons from digitized images. Journal of the ACM, 16 (4) (October 1969), pp.534-549.

[Moore,1956]
Moore.E.F. Gedanken experiments on sequential machines. In: C.E.Shannon and J.McCarthy (eds.), _Automata studies_. Princeton University Press, pp.129-153.

[Moore & Golledge,1976]
Moore,G.T., and Golledge,R.G. (eds.). _Environmental knowing: theories, research, and methods_. Dowden,Hutchinson and Ross, 1976.

[Moore,1975]
Moore, Robert C. Reasoning from incomplete knowledge in a procedural deduction system. AI-TR-347, MIT AI Lab., 1975.

[Moran,1973]
Moran, Thomas P. The symbolic nature of visual imagery, Proceedings Third International Conference on Artificial Intelligence, 1973, pp.472-477.

[Newell & Simon,1963]
Newell,A., and Simon,H.A. GPS, a program that simulates human thought. In: E.A.Feigenbaum and J.Feldman (eds.), _Computers and thought_. McGraw-Hill.

[Newell & Simon,1976]
Newell,A., and Simon,H.A. Computer science as empirical enquiry: symbols and search. Communications of the ACM, _19(3)_ (1976), pp.pp.113-126.

[Nilsson,1974]
Nilsson,N.J. Artificial Intelligence. _IFIP Congress, Stockholm, Sweden_ (1974), pp..

[Nilsson & Raphael,1967]
Nilsson,N.J., and Raphael,B. Preliminary design of an intelligent robot. Computer and information sciences, _7(13)_ (1967), pp.235-259.

[Papert et al.,1970]
Papert et al., Logo Memos, 1970-1971.

[Parzen,1960]
Parzen,E. _Modern probability theory and its applications_. John Wiley and sons, 1960.

[Pfaltz,1967]
Pfaltz, J.L., and Azriel Rosenfeld. Computer representations of planar regions by their skeletons. Communications of the ACM, _10_ (1967), pp.119-122.

[Pfefferkorn,1975]
Pfefferkorn, C.E. A heuristic problem solving design system for equipment or furniture layouts. Communications of the ACM, 18(5) (May 1975), pp.286-297.

[Prawitz,1960]
Prawitz,D. An improved proof procedure. Theoria, 26 (1960), pp.102-139.

[Pylyshyn et al.,1978]
Pylyshyn, Z.W., E.W. Elcock, M. Marmor, and P. Sander. Exploration in visual-motor spaces. Proc. Second National Conference Canadian Society for Computational Studies of Intelligence, pp.236-243.

[Reiter,1972]
Reiter,R. The use of models in automatic theorem-proving. Technical Report 72-09, Computer Science department,University of B.C., 1972.

[Robinson,1965]
Robinson,J.A. A machine oriented logic based on the resolution principle. Journal of the ACM, 12(1) (January 1965), pp.23-41.

[Rosenfeld & Pfaltz,1966]
Rosenfeld, A., and J.L.Pfaltz. Distance functions on digital pictures. J. Pattern Recognition, 1 (1966), pp.33-?.

[Rosenfeld & Kak,1976]
Rosenfeld,A., and Kak,A.C. Digital Picture Processing. Academic Press, New York, 1976.

[Sacerdoti,1977]
Sacerdoti, Earl D. A structure for plans and behaviour. Elsevier North-Holland, Inc., 1977.

[Shepard,1978]
Shepard,R.N. The mental image. American Psychologist, (February 1978), pp.125-137.

[Simon,1956]
Simon,H.A. Rational choice and the structure of the environment. Psychological Review, 63 (1956), pp.129-138.

[Simon,1977]
Simon.H.A.. Artificial intelligence systems that understand. Proceedings Fifth International Conference on Artificial Intelligence, 1977, pp.1059-1073.

[Sloman,1971]
Sloman, Aaron. Interactions between philosophy and AI: the role of intuition and non-logical reasoning in intelligence. Artificial Intelligence, 2 (1971), pp.209-225.

[Sloman,1978]
Sloman,A. The computer revolution in philosophy: philosophy, science, and models of mind. Harvester Press, 1978.

[Stallman & Sussman,1977]
Stallman,R.M., and Sussman,G. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit synthesis. Artificial Intelligence, 9(2) (October 1977), pp.135.

[Sussman,1973]
Sussman,G.J., The FINDSPACE problem. AIM 286, MIT AI Lab., March 1973.

[Sussman,1975]
Sussman,G. A computer model of skill acquisition. American Elsevier, 1975.

[Sussman,Winograd, & Charniak,1971]
Sussman,G., Winograd,T., and Charniak,E. Microplanner reference manual. AIM-203, MIT AI Lab., 1971.

[Tate,1975]
Tate,A. Interacting goals and their use, Proceedings Fourth International Conference on Artificial Intelligence, 1975, pp.215-218.

[Thompson,1977]
Thompson,A.M. The navigation system of the JPL robot,Proceedings Fifth International Conference on Artificial Intelligence, 1977, pp.749-757.

[Toda,1962]
Toda,M. The design of a fungus eater: a model of human behaviour in an unsophisticated environment. Behaviour, 7 (1962), pp. 164-183.

[Tolman,1948]
Tolman,E.C. Cognitive maps in rats and men. Psychological Review, 55 (1948), pp. 189-208.

[Trowbridge,1913]
Trowbridge,C.C. On fundamental methods of orientation and imaginary maps. Science, 88 (1913), pp.888-897.

[Tseitin,1968]
Tseitin,G.S. On the complexity of derivation in propositional calculus. In: A.O.Slisenko (ed.), _Studies in constructive mathematics and mathematical logic, Part II_, 1968.

[Warren,1974]
Warren,D.H.D. A system for generating plans. Memo 76, Department of computational logic, University of Edinburgh, 1974.

[Wells,1978]
Wells,M. _Octopus: physiology and behaviour of an advanced invertebrate_. John Wiley and sons, 1978.

[Winston,1970]
Winston,P.H. Learning structural descriptions from examples. AI-TR-231, MIT AI Lab., 1970.

[Winston,1978]
Winston,P.H. Learning by creating and justifying transfer frames. Artifial Intelligence, _10(2)_ (April 1978), pp. 147-172.

[Zucker,1978]
Zucker,S.W. Local structure consistency and continuous relaxation. TR-78-11, Department of electrical engineering, McGill University, 1978.