SPATIAL PROBLEMS IN A SIMULATED ROBOT

R.S.Rosenberg and P.F.Hawley

Department of Computer Science
University of British Columbia
Vancouver, B.C., V6T 1W5.

ABSTRACT

This paper 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 environment 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 oresented 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.

1 Introduction

1.1. Aims and motivation

This work was 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.

In order to explore the abilities required to exhibit such skills we proceeded as follows:

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

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.

1-2 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. Our main interest is in a sufficient design for the intermediary, which will be referred to as the robot-controller.

Its major task, 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. The purpose of a world model is to allow the construction of plans and thus to better achieve the organism's goals. A world model must be built to explain the sensory input received so far, 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 interface between an organism and the outside world is defined by the organism's sensory receptors and action effectors, and is necessarily always sloppy.

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.

2. The simulated organism-environment system

2.1 The physical system

This system is the basic experimental tool; it provides sensory input for, and accepts motor output from, the simulated organism called Utak. Only its functional input-output characteristics are directly relevant to the discussion. The aim of this section 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 (see section 3), 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 it and executing a pushto or turn command. *The* physics involved is essentially trivial: TABLETOP simulates the permanence and impermeability of the shape of physical objects.

In building the TABLE TOP system our aim was to produce an experimental tool that was inexpensive to use. We were not concerned to find exact solutions to collision problems. Thus, the approximate solutions to collision problems that the TABLETOP system computes are guite sufficient. Previous simulations of the physics of planar polygonal shapes [1,7,91 have either been incomplete, incorrect, or computationally expensive to use, whereas the TABLETOP system is complete, correct to within certain limitations, and efficient. Completeness means that both motion and collisions are handled.

The design of TABLETOP is based on the use of two representations for objects, the Cartesian and the digital. The Cartesian representation specifies the shape of the object by a list of points where each point is given by two positive real numbers. The points are the points of inflection on the boundary of the shape. An edge in the Cartesian representation of an object is a pair of consecutive points in the list. Utak himself has a Cartesian representation, or position, consisting of a single pair of positive reals. In addition, Utak has an absolute orientation. Here we refer 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 colour, 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 digital representation of the object is defined to be the set of squares of TABLE that lie within, or are intersected by the edqes 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, 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 below.

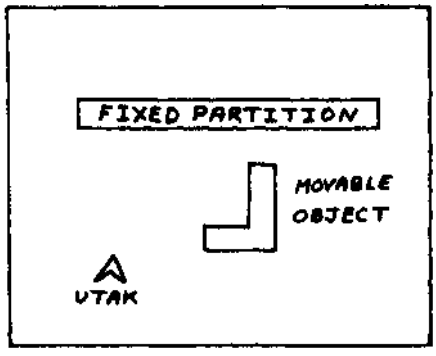

Figure 1. A Task Environment

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 release a held object. He has five motor outputs, only one of which is active at any one time. 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. 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. Figure 2 shows a retinal impression received by Utak corresponding to the situation shown in Figure 1.

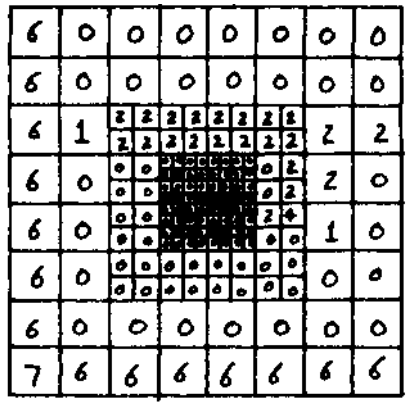

Figure 2. The integers in the squares from the retinal impression corresponding to Figure 1.

2.3 Examples of tasks for Utak

Here is a list of tasks, in English, which a competent organism-controller for Utak should be able to handle.

(i) "Go to the northeast corner"
(ii) "Go to the next room"
(iii) "Go to the square"
(iv) "Go round the square and return"
(v) "Push the square into the northeast corner"
(vi) "Push the square into the next room"
(vii) "Push the brick through the door"
(viii) "Push the brick around the corner"
(ix) "Push the L-shaped object into the next room"

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 is facing at that time becomes north in his orientation system. Among these tasks, the current system can handle (i) to (v).

3 Towards the design of a robot-controller

The purpose of this section is to present an approach to the design and implementation of a robot-controller for Utak.

3.1 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.

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. The problem in step 6 is to reconcile the currently assumed default world model with the world model implied by the task statement. We make this assumption to circumvent the handling of natural language input.

2-2. A first approach to implementation

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 3.1. Step 6, reconciling a world model with a task statement, was initially ignored on the basis that simple path-finding problems that did not require reconciliation of two world models would suffice. Steps 1-5 were accomplished and then step 7, the implementation of a spatial planner, was tackled. It was found that current techniques for path-finding were not really satisfactory 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 section.

3.3 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 sfloor 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 rectanqle (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.

By specifying the world model in this manner it is easy to modify it 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.

3.4 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 3.1).

3.4.1 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 thev 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.

3.4.2 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 table top which is not entirely floorspace, and similarly one with a graylevel of 7 may correspond to an area of the table top 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.

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.

3.4.3 Accommodating the default world model to the first~"retinal

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 actual dimensions of the containing rectangle of the $floor of the world model. After 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.

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.

3.5 Perception; accommodation to subseque"nt retinal impressions

Once a world model has been constructed that correctly interprets 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 topologically identical with a retinal impression. From the predicted world model a predicted retinal impression is computed.

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.

3.6 "The spatial planner

This is the heart of the system, 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. section 4) to find a path from start to destination that only traverses cells representing space.

4 Path-finding and the skeleton of a planar shape

-"A 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 [12]. It 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 idea is to project all the shapes onto a rectangular network of cells, the screen of section 3 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 lenqth 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 [8] using their function f=g+h to evaluate incomplete paths.

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 positions. Two paths are topologically distinct 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, our initial concern was to compute any reasonable path, not necessarily an optimal one.

4.,2 The skeleton

Blum, who called it the Medial Axis Function [2], was the first to introduce the skeleton of a planar shape. Since then it has been the topic of several investigations [5,10,11,14,15] and has

been called the distance transformation, the grassfire transformation, or the symmetric axis transformation. [3,4] has comprehensively analysed it and written about its potential applications to the description of shape in biology. Various definitions and algorithms for computing the skeleton in a digital space are given in [16].

4.3 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 3. 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.

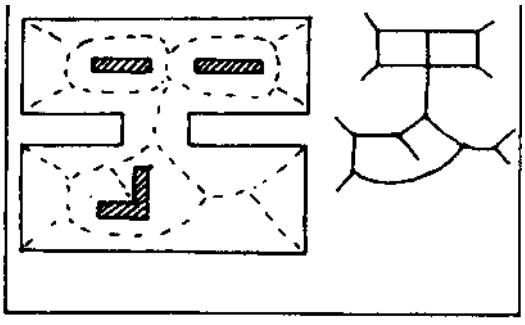

Figure 3. A skeletal graph and its corresponding path graph

5 Summary and conclusions

(a) We have explicitly described the features of the action cycle for a robot-controller; this has not been done before.

(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.

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

(c) There have been several robot simulation programs written before; however this is the first to handle the movement and collision of two dimensional shapes. Our contribution is to use a combination of the Cartesian and the digital representations to simulate the motion and collision of objects on a tabletop.

We began this project by asking what computational processes are required for spatial reasoninq. Our answer, and, briefly, the conclusion of the paper, 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.

Acknowledgements

This research was supported in part by the Natural Sciences and Engineering Council of Canada under grant no. A-5552.

References

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

[2] 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 , 362-380.

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

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

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

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

[7] Funt,B.V. WHISPER: A problem-solving system utilizing diagrams and a parallel processing retina. Proceedings Fifth International Conference on Artificial Intelligence, 1977(1977), 459-464.

[8] Hart,P., Nilsson,N., and Raphael,B. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Sys. Sci. Cybernetics7sSC-4, 2 (1968), 100-107.

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

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

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

[12] Nilsson,N.J. A mobile automaton :an application of artificial intelligence techniques. IJCAI-69.

[131 Nilsson,N.J., and Raphael,B. Preliminary design of an intelligent robot. Computer and information sciences, 7(13) (1967), 235-259.

[141 Pfaltz, J.L., and Azriel Rosenfeld. Computer representations of planar regions by their skeletons. Communications of the ACM, 10 (1967), 119-122.

[151 Itosenfeld, A., and J.L.Pfaltz. Distance functions on digital pictures. J. Pattern Recognition, 1 (1966), 33-61.

[161 Rowat,P.F. Representing spatial experienc and solving spatial problems in a simulated robot environment. Ph.D. Thesis, University of British Columbia, 1979.