To the Graduate Council:

I am submitting herewith a thesis written by Craig W. Peterson entitled "The Concentric Ripple Edge Evaluation and Progression Technique: An Autonomous Robot Navigation Algorithm." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Dr. Seung-Chul Lee, Major Professor

We have read this thesis
and recommend its acceptance:

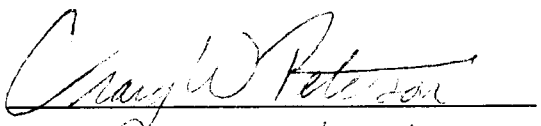
Accepted for the Council:

Associate Vice Chancellor
and Dean of The Graduate School

**STATEMENT OF PERMISSION TO USE**

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature 

Date 7-24-91

# The Concentric Ripple Edge Evaluation and Progression Technique: An Autonomous Robot Navigation Algorithm

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Craig W. Peterson

August, 1991

**DEDICATION**

This thesis is dedicated to me.

I hear and I forget
I see and I remember
I do and I learn

- Chinese Proverb -

Ripple in still water
When there is no pebble tossed,
Nor wind to blow.

R. Hunter, J. Garcia

**ACKNOWLEDGMENTS**

I would like to thank my major professor, Dr. S.C. Lee for his guidance in the development of this thesis. Thanks also to my other committee members, Dr. Moonis Ali and Dr. Bruce Whitehead, for their contributions. Their suggestions enhanced the depth and clarity of this thesis.

Special thanks to my proofreader, Brenda Lewis. More importantly, she deserves thanks for giving me confidence in my ability to write. I would also like to thank Mary Lo and the rest of the University of Tennessee Space Institute library staff. The speed and quality of their service is unsurpassed.

This section is incomplete without thanks for and recognition of all my parents have done for me. Their support, encouragement, and advice made achieving this degree possible. Thanks Mom. Thanks Dad.

**TABLE OF CONTENTS**


**1** INTRODUCTION . . . . . 1

1.1 Problem Statement . . . . . 2

1.2 Overview of Thesis . . . . . 4

1.3 Related Work in Autonomous Robot Navigation . 5

1.4 The Concentric Ripple Edge Evaluation and Progression Approach . . . . . 11

**2** ASSUMPTIONS, REPRESENTATION AND TERMINOLOGY . . . . . 14

2.1 Assumptions . . . . . 14

2.2 World Representation . . . . . 15

2.2.1 Grid Representation . . . . . 15

2.2.2 Grid Cell Size Selection . . . . . 16

2.2.3 Goal Cell Value Assignment . . . . . 19

2.3 Definitions and Terminology . . . . . 20

**3** PROPAGATION METHODOLOGY . . . . . 25

3.1 Formal Definition of the Value Propagation Technique . . . . . 25

3.2 Example of Value Propagation in an Unobstructed Environment . . . . . 29

3.3 Example of Value Propagation with Obstacles Present . . . . . 32

**4** ALGORITHM DEFINITION . . . . . 43

4.1 Formal Definition of the CREEP Algorithm . . . 43

4.2 Scanning . . . . . 46

4.3 Planning . . . . . 48

4.4 Moving . . . . . 50

4.5 The Curiosity Factor . . . . . 52

4.6 Reducing Cell Value Calculations . . . . . 53

**5** ALGORITHM ANALYSIS . . . . . 55

5.1 Formal Analysis . . . . . 55

5.2 Wavelike Qualities of the Value Propagation in the CREEP Algorithm . . . . . 56

**6** CREEP SIMULATION AND ANALYSIS . . . . . 58

6.1 Example One . . . . . 60

6.2 Example Two . . . . . 63

6.3 Example Three . . . . . 69

6.4 Example Four . . . . . 69

6.5 Example Five . . . . . 72

6.6 Analysis of Example Cases . . . . . 84

**7** IMPLEMENTATION ON THE HERO 2000 . . . . . 87

7.1 The HERO 2000 . . . . . 88

7.1.1 The HERO BASIC Instruction Set . . . . 88

7.1.2 Remote Control Mode . . . . . 88

7.1.3 Sonar Capabilities . . . . . 89

7.1.4 Mobility . . . . . 89
7.2 The Control Program . . . . . 91
7.3 Implementation Analysis . . . . . 92

8 CONCLUSIONS AND SUMMARY . . . . . 94
8.1 Conclusions . . . . . 94
8.2 Summary . . . . . 95

BIBLIOGRAPHY . . . . . 99
VITA . . . . . 104

## LIST OF FIGURES

Figure 2-1: Four Adjacent Cells . . . . . 22

Figure 2-2: Propagation Waves . . . . . 24

Figure 3-1: The Propagation Routine . . . . . 26

Figure 3-2: Initial Grid . . . . . 30

Figure 3-3: First Cell Value Calculated . . . . . 31

Figure 3-4: Second Cell Value Calculated . . . . . 33

Figure 3-5: First Propagation Wave Complete . . . . . 34

Figure 3-6: Propagation Complete . . . . . 35

Figure 3-7: Initial Grid with Two Obstacles . . . . . 37

Figure 3-8: First Wave of Value Propagation with Obstacles . . . . . 38

Figure 3-9: Second Wave of Value Propagation with Obstacles . . . . . 39

Figure 3-10: Third Wave of Value Propagation with Obstacles . . . . . 40

Figure 3-11: Fourth Wave of Value Propagation with Obstacles . . . . . 41

Figure 3-12: Propagation with Obstacles Complete . . . 42

Figure 4-1: The CREEP Algorithm . . . . . 44

Figure 4-2: Placing a Detected Obstacle . . . . . 47

Figure 4-3: Robot Heading is Forty-Five Degrees . . . . 49

Figure 6-1: Example 1, Case 1,2,3,4 Obstacle Configuration . . . . . 61

Figure 6-2: Example 1, Case 1,2,3,4 Cause of a Zig-Zag 61

Figure 6-3: Example 1, Case 1,3,4 The Complete Path . . 62
Figure 6-4: Example 1, Case 2 A Path Deviation . . . . 64
Figure 6-5: Example 1, Case 2 The Complete Path . . . . 64
Figure 6-6: Example 2, Case 1,2,3,4 At The Initial Goal . . . . . 65
Figure 6-7: Example 2, Case 1,3 The Return Trip Values 65
Figure 6-8: Example 2, Case 1,3,4 The Complete Path . . 67
Figure 6-9: Example 2, Case 2 Cell Values for Return Trip . . . . . 67
Figure 6-10: Example 2, Case 2 The Complete Path . . . 68
Figure 6-11: Example 3, Case 1,2,3,4 Obstacle Configuration . . . . . 70
Figure 6-12: Example 3, Case 1,2,3,4 A Misleading Hole 70
Figure 6-13: Example 3, Case 1,2,3,4 The Complete Path 71
Figure 6-14: Example 4, Case 1,2,3,4 Obstacle Configuration . . . . . 73
Figure 6-15: Example 4, Case 1,2,3,4 At the Initial Goal . . . . . 73
Figure 6-16: Example 4, Case 1,2,3,4 The Complete Path 74
Figure 6-17: Example 5, Case 1,2,3,4 Obstacle Configuration . . . . . 74
Figure 6-18: Example 5, Case 1,2,3,4 Dead End Recovery 75
Figure 6-19: Example 5, Case 1 The Complete Path . . . 75
Figure 6-20: Example 5, Case 2 The First Dead End Avoided . . . . . 77
Figure 6-21: Example 5, Case 2 A Second Dead End Found 77

Figure 6-22: Example 5, Case 2 At the Initial Goal . . 78
Figure 6-23: Example 5, Case 2 The Complete Path . . . 78
Figure 6-24: Example 5, Case 3 The Complete Path . . . 79
Figure 6-25: Example 5, Case 3 Cell Values Before Returning . . . . . 79
Figure 6-26: Example 5, Case 3 Cell Values, Robot at (16,5) . . . . . 80
Figure 6-27: Example 5, Case 3 Values Leading Robot Astray . . . . . 82
Figure 6-28: Example 5, Case 3 Values After Robot is Stuck . . . . . 82
Figure 6-29: Example 5, Case 4 The Complete Path . . . 83
Figure 7-1: The HERO 2000 in Link Mode . . . . . 90

## List of Symbols

| | |
|---|---|
| $\forall$ | For every |
| $\leftarrow$ | Is assigned |
| $\in$ | Is an element of |
| $\equiv$ | Is defined as |
| $\mid$ | Such that |
| $\neg$ | Logical not operator |
| **GRID** | A $m \times n$ matrix of cells |
| $\check{R} \equiv (r1, r2)$ | The location of the robot |
| $\check{G} \equiv (g1, g2)$ | The location of the goal |
| $O$ | The linked list of cells awaiting propagation |
| $\nu(x,y)$ | A real value assigned to cell (x,y) |
| $\psi(x,y)$ | A flag indicating if cell (x,y) has been scanned by the robot |
| $o(x,y)$ | A flag indicating if cell (x,y) is in $O$ |
| $\kappa(x,y)$ | A flag indicating if cell (x,y) is closed |
| $\delta(x,y,s,t)$ | A function that calculates the distance between points (x,y) and (s,t) |
| $\varsigma$ | The curiosity factor |
| $\beta(a,b)$ | A boolean function that returns 1 if a=b, a 0 is returned otherwise |
| $N(x,y)$ | The neighborhood of cell (x,y) |
| $MN(x,y)$ | The maximal neighbor of cell (x,y) |
| **NEW_OBSTACLES** | A global flag indicating if new obstacles were detected in the scanning phase |
| **GOAL_VALUE** | A constant assigned to $\nu(g1,g2)$ |
| $\rho$ | The heading of the robot, measured in degrees |
| $\gamma$ | A grid degree measurement, 0° corresponds to the positive x-axis. |

# CHAPTER 1

# INTRODUCTION

Applications exist in many fields for autonomous mobile robots. A self-guided robot capable of navigating through an unfamiliar environment can achieve missions dangerous or infeasible for people. For example, a robot rover on the surface of Mars could perform most, if not all, the experiments and data collection that a manned team could. The expense and danger of a manned mission to Mars is many times greater than one using autonomous robots.

Radio remote control of military vehicles is vulnerable to electronic jamming and other forms of interference. The radio communication can also jeopardize the secrecy of the mission or disclose the location of the rover. However, a self-guided robot is unaffected by these problems.

Additionally, an autonomous vehicle used in geological surveys can position remote sensing devices, gather samples from many locations, and perform on-site experiments over a wide area, freeing the geologist for more complex tasks. These are only a few examples of potential applications of autonomous robots.

### *1.1 Problem Statement*

Robot navigation techniques are divided into two categories based on the environment information available. When all data are available, the robot has complete information of its surroundings. Using this model and the A* algorithm it is possible to find the optimal path, avoiding all obstacles between the start and goal locations.

The path can be generated using a one-time, off-line calculation. Once planned, the robot executes the path from beginning to end.

When little or no data are initially available to the robot, it must operate with incomplete information. The path planning task is obviously more difficult in this situation.

A three step plan is used in navigating through an unknown environment. First, the robot must be able to detect obstacles and maintain obstacle location information. Second, a path must be generated that will avoid obstacles and lead the robot to the goal location. Finally, the robot must move along the path developed in the second step. These functions are handled by three separate modules: the scanner, the planner, and the mover.

The scanner acquires information about the surroundings of the robot. The planner uses that information to generate a path based on limited information. The mover follows the path generated by the planner. As new data are acquired

it may be necessary to revise the plan generated in an earlier phase. Feedback between the robot's mover, scanner, and planner modules is critical. Information about newly discovered obstacles from the scanner must be available to the planner; path revisions need to be communicated to the mover.

Obstacle detection can be accomplished in many ways. For example, tactile sensing, radar, sonar, and image processing techniques can be used individually or jointly. The advantages and complexities of several methods are explained in [Daily et al, 1988; Matthies et al, 1988;]. The important issue is that the algorithm defined herein works regardless of the sensor type(s) used.

Briefly summarized, a grid-based representation is used to store environment information. As obstacles are discovered, the corresponding grid cell is tagged as being blocked. The location of the goal and the current position of the robot are also maintained. A detailed description of the grid representation scheme is given in Chapter 2.

A robot requires a path planning strategy to (1) determine which paths might lead to the goal location, (2) select which of the possible paths should be explored, (3) allow the robot to avoid obstacles, and (4) recover from dead ends. Several valid approaches are described in Section 1.3.

Research initiated by S.C. Lee, V. Minsky, and J. Perry

in 1987 at the University of Tennessee Space Institute is used as a basis for the algorithm defined in this thesis. The original algorithm, Block Emanating Signal, has been modified and renamed as the Concentric Ripple Edge Evaluation and Progression (CREEP) algorithm. The CREEP approach meets the four needs listed above and has several advantages. This thesis formally defines the CREEP algorithm, demonstrates the flexibility and power of the approach, provides an analysis of the algorithm, and examines the feasibility of implementing the CREEP on a HERO 2000 robot.

### *1.2 Overview of Thesis*

Chapter 1 stresses the importance of path planning in autonomous robot navigation. Several path planning approaches currently being investigated are described. A brief preview of the CREEP algorithm is also given.

The second chapter provides a detailed description of the grid-based representation used by CREEP. Criteria for grid cell size selection are presented. Terminology used throughout this thesis is also defined in this chapter.

A critical portion of the CREEP algorithm is the wavelike manner in which values propagate through the grid. Chapter 3 gives an in-depth description of the propagation routine. Two examples are used to clarify the concept. The first demonstrates the technique without obstructions; the second shows how obstacles affect the process.

The CREEP algorithm is formally defined in Chapter 4. The three major steps of the algorithm are described: scanning, planning, and moving. An alternative for calculating cell values is also presented.

In Chapter 5 the performance and time complexity analysis of the CREEP algorithm is presented. Several desirable properties of the wavelike manner in which values propagate through the grid are stated.

Results from computer simulations of CREEP are examined in Chapter 6. Several examples illustrate the strength of this approach. An analysis of the simulation results is presented at the end of the chapter.

The CREEP algorithm has been implemented using a HERO 2000 robot. Descriptions of the robot, its language interpreter (HERO 2000 BASIC), its sensor system, and mobile capability, are provided in Chapter 7. The control program is also discussed.

Conclusions are presented in Chapter 8. The final section gives a summary of all material covered in this thesis.

### *1.3 Related Work in Autonomous Robot Navigation*

Many approaches to autonomous robot navigation have been studied. Several papers presenting techniques and results are discussed in this section. Similarities with and contrasts to the CREEP approach are mentioned.

Rule-based expert systems have been successfully applied in many domains. An industrial process controller and fault diagnostic expert system that has been modified for use in robot navigation is described in [DeSaussure et al, 1987] . Two main algorithms are used in this system.

The first, the navigation algorithm, consists of a scanning phase and a motion phase. If no obstacles are detected in the sonar sweep, the vehicle moves toward the goal. To circumnavigate a detected obstacle, the robot moves to within three feet of it. At this point another scan is performed and an intermediate goal location is established. The robot moves to this new, temporary goal and repeats the process.

The second algorithm is invoked to handle unexpected obstacles (i.e., an obstacle detected in a location where none was previously found). The action taken is based on the classification of the obstacle. Some example rules using such a classification are:

If the object is stationary and three feet or taller, then invoke the navigation algorithm.

If the object is stationary and less than three feet tall, then move the object out of the way and invoke the navigation algorithm. *NOTE: An assumption is made that any object smaller than three feet tall can be moved.*

If the object is moving away from the robot, then wait

for the path to clear and proceed.

If the object is moving toward the robot, then take evasive action.

One obvious difficulty of any rule-based approach is the maintenance of the rules. Contradictions must be avoided. Additionally, the time required to search for applicable rules increases as the size of the rule base grows.

A conceptual method for robot rover navigation in an essentially unknown environment using fuzzy sets and fuzzy rules is presented in [Gaffney 1972]. An example of a fuzzy set "the area several feet ahead." That is, the set of all land 'several' feet in front of the rover is a fuzzy, or poorly defined, concept.

To quantify 'several' feet, a probability is assigned to values in an acceptable range. The data below show one possible representation. A random number is used to select one of the weighted values; the value of the fuzzy variable

| **Distance (ft)** | **Probability of Selection** |
|---|---|
| 0 | 0.05 |
| 1 | 0.2 |
| 2 | 0.25 |
| 3 | 0.35 |
| 4 | 0.15 |

becomes the selected value. This process is used each time a value is needed. The random selection of values makes verification and reproduction of test results difficult, if not impossible.

Several intuitive rules (heuristics) for robot navigation have been studied in [Chatergy, 1985; Ilari, 1990; Mehrotra, 1989]. The major drawback of heuristic rules is their limited applicability. That is, each rule works only for a limited number of scenarios and poorly or not at all in others.

Just as there are many path planning techniques, the representation schemes are also varied. Many approaches use a grid-based representation similar to that used in the CREEP algorithm [Elfes, 1989; Lee, 1961; Matthies et al, 1988; Shiller et al, 1991]. Lee's method, used in designing circuit paths in chip design, assumes complete information. His approach attempts to minimize both the path length and the number of existing paths that are crossed by the new path being generated. The CREEP approach lends additional flexibility by allowing path planning when complete information is not available and by permitting moves in diagonal directions (a practice not used in chip design). Elfes' method works in an environment with incomplete information, allows diagonal movement, and relies on the A* algorithm for path planning.

The grid representation used in [Matthies et al, 1988]

is used to incorporate sonar data from a mobile robot. The central issue of this paper is the proper interpretation of sonar data, not path planning.

A modified algorithm, originally intended for manipulator movement planning, is discussed in [Shiller, 1991]. A set of possible solutions is generated. A filtering process eliminates unlikely candidates; a smoothing algorithm is applied to the remaining paths. The best of these is selected and followed.

Another common technique uses a quad-tree to represent environment information [Ali et al; 1987, Mehrotra, 1989; Wong et al, 1986]. This scheme reduces the amount of information stored for an image by clustering homogeneous regions. The leaves of the tree store individual cell information and are labeled as either open or blocked. Intermediate nodes are labeled based on the labels of their children; a node with all of its children open (closed) is labeled open (closed). Intermediate nodes can also be tagged as mixed. Thus, once an open (closed) node is found, it is not necessary to explore deeper into the quad-tree, as all of its children are known to be open (closed). The major drawback to this scheme is the time needed to search the tree; conventional search techniques applied to quad-tree traversal are slow.

The "wavefront propagation" technique has also been studied by other researchers [Chavez et al, 1984; Mitchell

et al, 1987; Parodi, 1985; Witkowski, 1983]. The wavefront propagation method is one of the central components of the path planning algorithm presented in this thesis.

The terrain representation used in [Chavez et al, 1984] differs from the grid-representation used in the CREEP approach. The environment is divided into 'patches' of homogeneous regions; a patch is characterized by equal traversability within its borders. Each region has a traversal cost associated with it and moves are made from the center of one region to the next. A movement cost within a patch is calculated by multiplying the distance travelled by the traversability rating for that region. Movement cost between regions is the sum of the intra-regional movement costs.

The different representation schemes used by three different path planning algorithms are presented in [Mitchell et al 1988]. An overseeing mission planner determines which of these techniques to apply. The grid-based path planner represents data in a grid; the road network planner uses a graph network representation; the weighted region planner divides the terrain into polygonal regions.

Of these three methods, the CREEP algorithm most closely resembles the grid-based approach, both for its representation and cell assignment technique. However, Mitchell incorporates *a priori* information in the grid. Path evaluation begins at the robot's current location and

all paths of equal length leading to the goal are considered as candidates. Whereas, the CREEP cell assignment begins at the goal cell and stops when the wave reaches the robot. The maximal neighbor of the robot represents the single path the robot will follow.

Witkowski's [1983] algorithm was implemented on an ICL Distributed Array Processor. The assumption of complete information is made. The entire path is planned off-line before any attempt is made to initiate navigation. The CREEP implementation operates with incomplete information, and performs calculations as the robot navigates through its environment. No *a priori* knowledge is assumed.

### *1.4 The Concentric Ripple Edge Evaluation and Progression Approach*

This section provides a brief description of the CREEP algorithm. An in-depth explanation of the technique is provided in Chapters 3 and 4.

Incomplete information of the robot's surroundings is assumed. Initially, only the position of the robot and the relative position of the goal are known. The robot is assumed to have a finite scanning range; all obstacles are stationary and can be detected by the robot, provided they are within range and are not hidden by other obstacles. The number and location of obstacles are initially unknown. The surface on which the robot travels on is assumed to be level and without holes.

The environment is modeled using a two dimensional grid, divided into equal-sized cells. As the robot gains data about its surroundings, corresponding grid cells are tagged as having been *seen* by the robot.

In addition to flags, each cell has a real value associated with it. When the algorithm is initialized, all cells, except the goal cell, are given a value of zero. The goal cell is initialized to a constant (the determination of this value is explained in Section 2.2.3). Any detected obstacles are assigned a value of -1.

All other cell values are determined in a propagation routine (specifically described in Section 3.1). A signal, initially equivalent to the value of the goal cell, emanates in all directions outward from the goal. The signal wave weakens in proportion to its distance from the goal. As the wave moves outward, it crosses cells; those cells are assigned values equal to the strength of the wave. Hence, the smaller a cell's value, the greater its distance from the goal cell.

The wave must travel around obstacles. Thus, the signal wave is weakened by obstacles since it is forced to travel a greater distance. Obviously, the strongest wave to reach the robot travels the shortest distance and, therefore, represents the best path to explore.

Once the propagation is complete, all cells adjacent to the robot are examined. The cell with the maximum value

lies in the direction from which the strongest wave arrived. The robot moves to this cell and begins the scan, plan, move cycle again. The propagation routine is invoked only when new obstacles are detected. This scan, plan, move cycle repeats until either the robot reaches the goal or all paths are explored and determined to be blocked. Once at the goal, the initial starting location is established as the new goal. All obstacle information is maintained and a return path is planned and executed.

# CHAPTER 2

# ASSUMPTIONS, REPRESENTATION AND TERMINOLOGY

This chapter lists the assumptions used for developing the CREEP algorithm. The grid representation scheme is also described. The criteria for selection of grid cell size are presented. The importance of assigning a proper value to the goal cell is discussed. The calculation of this value is also presented. The final section of the chapter provides definitions of terms used in the remainder of the thesis.

## *2.1 Assumptions*

Several assumptions are made for the simulation and implementation of the CREEP algorithm. First, the robot has the ability to move. Second, the scanning range of the robot is finite. Third, all obstacles are stationary and can be detected by the robot, provided they are within the sensor range and are not hidden by other obstacles. The number and location of the obstacles are initially unknown to the robot. Fourth, the surface on which the robot travels is assumed to be level and without holes. Finally, the environment the robot operates in is assumed to be closed (e.g., a room with surrounding walls).

The final assumption is made for reasons of convenience, rather than of necessity. The algorithm still functions without this additional constraint, but the implementation is simplified by including it.

## *2.2 World Representation*

The three dimensional environment is modeled using a two dimensional plane. All data obtained about obstacles are stored on this plane. All movement is modeled in two dimensions. Future research may remove these constraints in order to develop a more elaborate model.

### *2.2.1 Grid Representation*

The navigable terrain is divided into equal size cells. The internal representation of the terrain is maintained as an $m \times n$ grid containing $mn$ square cells. This grid stores all known information about the terrain and is updated as new data are acquired. The robot and goal locations are also maintained.

The cells are represented as structures, allowing multiple pieces of information to be stored with each. Several boolean flag values are maintained. A list at the beginning of this thesis provides the symbols associated with each of these flags. For example, once a cell has been scanned it is tagged as *seen*, $\psi(x,y)$; the propagation routine (described in Chapter 3) uses *closed*, $\kappa(x,y)$, and

*open*, o(x,y), flags to differentiate between cells that have had values assigned during the current propagation process and those awaiting value calculation, respectively.

A real number is associated with every cell, v(x,y). Cells containing obstacles are given a value of -1. All other cells have non-negative values. The calculation of these values is described in Chapter 3. In short, the greater the distance the robot would have to travel from a cell to the goal, the smaller the cell value.

The goal cell's value, **GOAL_VALUE**, is constant and is the maximum of any in the grid. The calculation of this constant value is discussed in Section 2.2.3.

### *2.2.2 Grid Cell Size Selection*

Many criteria must be examined when selecting the size of the grid cells. Some considerations include the distance between the goal and the robot's starting point, the accuracy of obstacle representation desired in the modeling process, the storage capacity of the robot, the size of the robot, and the accuracy and scanning range of the sensors.

The only data the robot begins with are the locations of itself and the goal. Clearly, the grid must include each of these positions. These two points provide minimum boundaries for the grid. A rectangle is defined by two opposing corners. Using the goal as one corner and the

starting position as the opposing corner defines a rectangle that can be subdivided into cells for grid representation.

However, this minimal rectangle could prove too restrictive. If obstacles exist near either point, movement outside the rectangle described may be necessary in order for success to be achieved. Therefore, expanding the defining rectangle beyond the limits of the goal and start locations provides more potential paths for exploration. Once the dimensions of this bounding rectangle are determined, other criteria may be applied to determine the most suitable size for individual cells.

The dimensions of the grid cells can be varied according to the precision desired in representing the robot's surroundings. Clearly, the larger the cells, the less accurate the mapping from the real world to the grid will be. The larger the cell size, the fewer cells required to represent the environment. Savings are gained in computation time by sacrificing model precision and storing information about fewer grid cells.

Conversely, smaller cells give additional accuracy to the model. The cost of extra refinement is paid in greater calculation time, primarily in the propagation routine.

The storage capacity of the robot provides an upper bound for the number of cell structures that can be maintained. For example, if each cell requires 4 bytes of memory and 1K (1024 bytes) is available as storage, then no

more than 256 grid cells could be maintained. If memory is limited, it would be possible to buffer grid information on another storage medium (e.g., magnetic tape or disk) at a considerable loss of speed.

Additional criteria may be influenced by the hardware available on the robot. Maximizing storage utilization (use all available memory for modelling), minimizing storage access delays (i.e., do not use slow storage media) or minimizing computation time may be desirable, depending on the resources available.

The cells should not be larger than the range of the scanner. As stated earlier, the robot regards a cell that is partially blocked as being impassable. If cells sides were greater than the scanning range, an obstacle in the far region of an obstacle would not be detected, and the cell would incorrectly be considered open. In effect, a cell size that is too large causes nearsightedness or possibly blindness for the robot.

The accuracy of the sensor system must also be considered. For example, if a sensor is accurate within plus or minus 6 inches, then a cell size of 12 inches or less would cause discrepancies in the grid representation. The same cell may be seen as clear in one scan, yet blocked in the next. An approach for dealing with this type of inconsistency has already been outlined in [Elfes, 1989].

The simulation and implementation of the CREEP algorithm assumes a cell size roughly equal to the dimensions of the HERO 2000 robot (eighteen inch sides). Thus, the robot can move from one cell to another. Whereas, in a scenario using smaller cells, it would be necessary to find multiple adjacent cells for the robot to move into. Conversely, if a larger cell size is used, a small obstacle in any portion of that cell would cause it to be blocked. Thus, the robot would not attempt to move through this cell even though such a move is, in reality, possible.

### *2.2.3 Goal Cell Value Assignment*

The goal cell is the starting point for the propagation routine. Each cell value is calculated based on the length of the path, including obstacle avoidance, that connects a cell to the goal. A cell's value is calculated by subtracting this path length from the value of the goal cell. Since cells not containing obstacles should have a non-negative value, the goal value must be no smaller than the length of the longest possible path in the grid.

A simple calculation will ensure that the constant **GOAL_VALUE** is larger than any possible path length in a $m \times n$ grid. This calculation is shown in formula **[2.1]**.

$$\mathbf{GOAL_VALUE} \leftarrow mn \qquad [2.1]$$

### *2.3 Definitions and Terminology*

The next two chapters provide a detailed explanation of the CREEP algorithm. While many of the terms used are commonplace, they are defined in this section in order to eliminate confusion with identical terms used differently in other works. The first time a term is formally defined it will appear in **boldface** type. Thereafter, any use of that term is assumed to be consistent with the definition. A concise listing of the symbols used throughout this thesis can be found in the front of this thesis.

The distance between two cells, (x1,y1) and (x2,y2), is measured from one cell center to the other. Thus $\delta(x1,y1,x2,y2)$ is the Euclidean measure of distance between two points on a plane.

Two cells, (x1,y1) and (x2,y2), are **adjacent** if

$$\delta(x1,y1,x2,y2) \leq \sqrt{2}$$

The definition includes horizontally, vertically, and diagonally adjacent cells.

Given two adjacent cells, neither of which contains an obstacle, one of two possible relationships exists between them. If no obstacles prevent movement between the cells they are said to be **passable**. That is, the robot can freely pass from one to the other with no possibility of collision. Any two horizontally or vertically adjacent obstacle-free cells are passable. Given two cells, one is **obstructed** from the other if movement between them is inhibited by an

adjacent cell containing an obstacle. As an example, a typical set of four adjacent cells is shown in Figure 2-1. By definition,cell A will always be passable with respect to either of the B cells, provided none of these cells contains an obstacle.

The relationship between cell A and cell C depends on the status of the B cells. If either of the B cells contains an obstacle, cells A and C are obstructed from one another. A robot moving diagonally from A to C will collide with the obstacle contained in cell B.

Every cell containing an obstacle is obstructed from all adjacent cells. The converse is not true; every obstructed cell does not necessarily contain an obstacle, as shown in the previous example.

The **passable neighborhood** of cell (x,y), denoted N(x,y), is the set of all non-obstructed cells adjacent to (x,y). Thus, a passable neighborhood will contain at most eight cells.

Any cell in N(x,y) is called a **passable neighbor** of (x,y). The **maximal neighbor** of (x,y), denoted MN(x,y), is the passable neighbor with the highest value assigned to it. If two or more cells share the same maximum value, one cell will be selected arbitrarily.

A **propagation wave** is a set of cells which are equally distant from the goal. This distance is measured in the

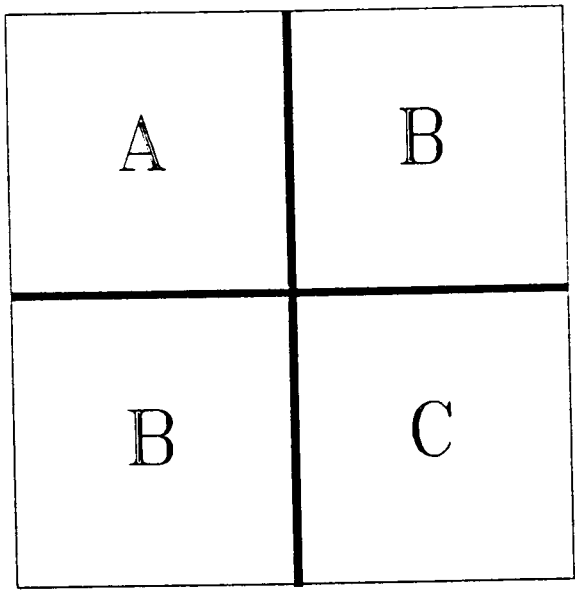



**Figure 2-1: Four Adjacent Cells**

number of cells crossed and not the actual distance travelled. Example waves are shown in Figure 2-2.

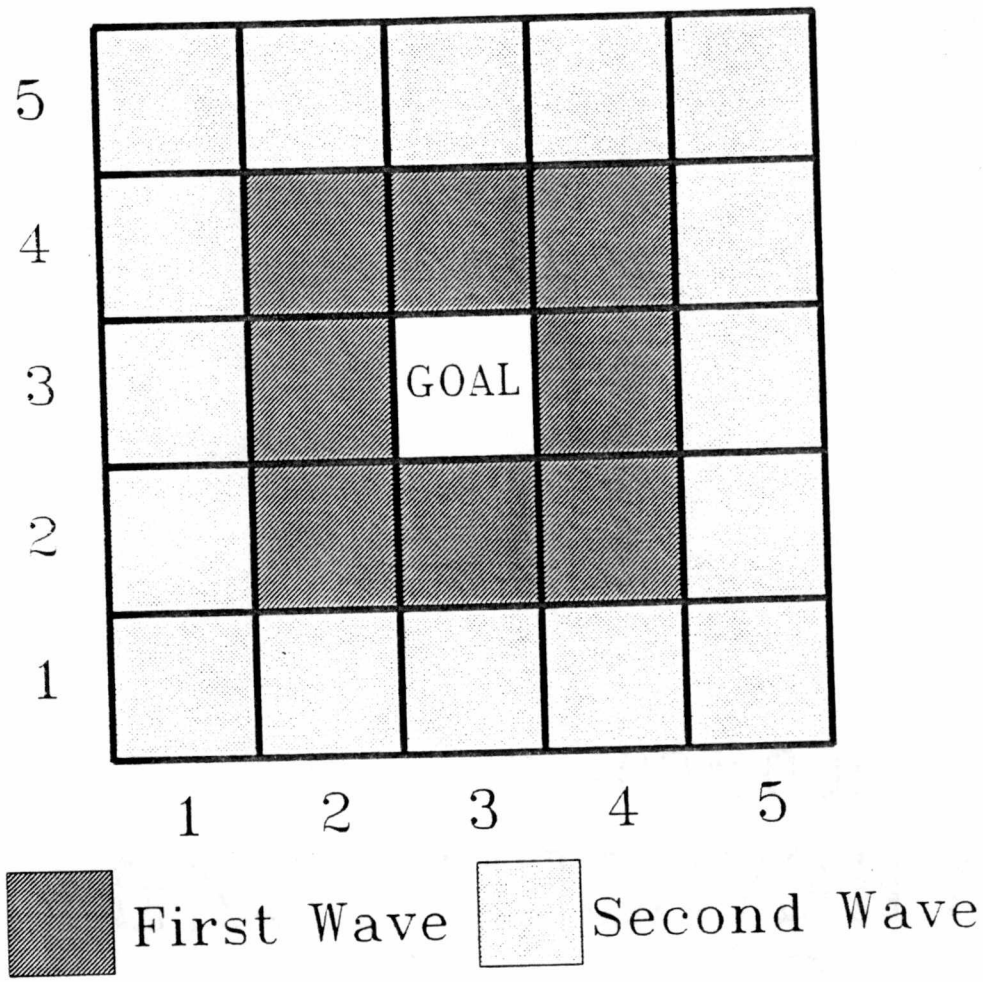



**Figure 2-2: Propagation Waves**

# CHAPTER 3

# PROPAGATION METHODOLOGY

The central component of the CREEP algorithm is the manner in which values are assigned to grid cells. It is necessary to have a complete understanding of the value propagation procedure before presenting the entire CREEP algorithm. Therefore, this chapter provides a detailed explanation of the value propagation technique used. The formal definition is followed by two examples intended to further illustrate the method.

## *3.1 Formal Definition of the Value Propagation Technique*

A pebble dropped into still water sends many waves rippling out from the disturbance. The further a wave travels the weaker it becomes. Likewise, values are assigned to grid cells in a similar fashion. The goal cell is analogous to the spot where the pebble breaks the surface of the water. A single wave is sent out from the goal; each cell the wave crosses is given a value proportional to the strength of the wave.

The algorithm used in value propagation is shown in Figure 3-1. The first step handles initialization. The value of each cell is initialized to zero; all *open* flags

**1.** Initialize

$\forall x \mid 1 \leq x \leq m$ and $\forall y \mid 1 \leq y \leq n$

if $\nu(x,y) > 0.0$

$\nu(x,y) \leftarrow 0.0$

$o(x,y) \leftarrow$ **FALSE**

if $\psi(x,y)$ and $\nu(x,y) < 0$

$\kappa(x,y) \leftarrow$ **TRUE**

else

$\kappa(x,y) \leftarrow$ **FALSE**

$\nu(g1,g2) \leftarrow$ **GOAL_VALUE**

$\kappa(g1,g2) \leftarrow$ **TRUE**

$O \equiv \{ (x, y) \mid (x, y) \in N(g1,g2) \}$

$\forall (x,y) \in O \quad o(x,y) \leftarrow$ **TRUE**

**2.** Do

**(cx,cy)** ≡ First cell in O

**(mx,my)** ≡ $MN(cx,cy)$

$\nu(cx,cy) \leftarrow \nu(mx,my) - \delta(cx,cy,mx,my)$

$\forall (x,y) \mid (x,y) \in N(cx,cy)$

if $\neg o(x,y)$ and $\neg\kappa(x,y)$

$O \leftarrow O + \{(x,y)\}$

$o(x,y) \leftarrow$ **TRUE**

$O \leftarrow O - \{(cx,cy)\}$

$o(cx,cy) \leftarrow$ **FALSE**

$\kappa(cx,cy) \leftarrow$ **TRUE**

While $\{O \neq \emptyset$ AND $\neg\kappa(x,y)\ \forall\ (x,y) \in N(r1,r2)\}$

**Figure 3-1: The Propagation Routine**

are set to **FALSE**. The goal cell and all cells known to contain obstacles maintain the value of **TRUE** for their *closed* flags; all other *closed* flags become **FALSE**. The value of the goal cell is set to the constant **GOAL_VALUE**. More details on the importance of properly calculating the value of this constant are given in Section 2.1.3.

The same step also initializes O. Additions to and deletions from this list are strictly ordered. Each added cell becomes the last item on the list. Only the first cell in the list may be deleted. Thus, O is a first-in first-out (FIFO) structure. It is initialized to consist of all unobstructed cells adjacent to the goal cell. The actual implementation uses a linked list structure for O. Since it is necessary to check membership in this list many times during the propagation process, a flag (*open*) associated with each cell indicates whether or not it belongs to O. This process eliminates the need for a linear search through the linked list each time a membership test is performed.

After initialization, an iterative cycle is invoked. Each iteration begins by removing the first cell from O. This cell will be called the *current* cell, and is represented as **(cx,cy)**. The maximal neighbor, MN**(cx,cy)** is selected, call it **(mx,my)**. The value of the *current* cell is calculated as:

$$v(\mathbf{cx},\mathbf{cy}) \leftarrow v(\mathbf{mx},\mathbf{my}) - \delta(\mathbf{cx},\mathbf{cy},\mathbf{mx},\mathbf{my}) \qquad [3.1]$$

Next, O is updated. Each passable neighbor of **(cx,cy)** that is neither *open* nor *closed* is added to the end of O. Finally, **(cx,cy)** is removed from O, its *open* flag becomes **FALSE**, and its *closed* flag is set to **TRUE**. The process terminates when either of two conditions is met.

First, when the propagation wave has assigned values to all passable neighbors of the robot, all these cells are closed. That is, none of these cells will have new values assigned to them during this invocation of the propagation routine. So, the maximal neighbor of the robot is determined. Since this is exactly what is needed in selecting a cell to move to, no further calculations are needed and the process ends.

When O is empty all possible paths connecting the goal have been explored. If this occurs before all passable neighbors of the robot are closed, no path exists and the CREEP algorithm terminates.

Whenever neighbors are inspected, the first cell considered is directly above the *current* cell. The examination proceeds in a clockwise direction.

The next two sections demonstrate the propagation process just described. A 5 x 5 grid is used for these examples; the goal cell is in the center of the grid. A larger grid would be used in most applications, but for demonstration purposes this size is sufficient.

### *3.2 Example of Value Propagation in an Unobstructed Environment*

In the first example no obstacles exist. Multiplying the dimensions of the grid gives the initial value for the goal cell, 25. All other cell values are set to zero. Since there are no obstacles on the grid, only the goal cell has its *closed* flag set **TRUE**. O initially holds the eight passable neighbors of the goal cell. The initial grid is shown in Figure 3-2.

The first member of O, (3,4), becomes the *current* cell. Cell (3,3) is the maximal neighbor. So, from formula **[3.1]**, the value assigned to **(cx,cy)** is:

$$\begin{aligned} v(3,4) &\leftarrow v(3,3) - \delta(3,4,3,3) \\ &= 25.0 - 1.0 = 24.0 \end{aligned}$$

Three new cells are appended to O. They are: (3,5), (4,5), and (2,5), in that order. The *current* cell is removed from O, its *open* flag becomes **FALSE**, and its *closed* flag is set to **TRUE**. The result of this iteration is portrayed in Figure 3-3.

Again, the *current* cell is set to the first item in O, (4,4). Similarly, MN(4,4) is (3,3), so

$$\begin{aligned} v(4,4) &\leftarrow v(3,3) - \delta(4,4,3,3) \\ &\approx 25.0 - 1.4 = 23.6 \end{aligned}$$

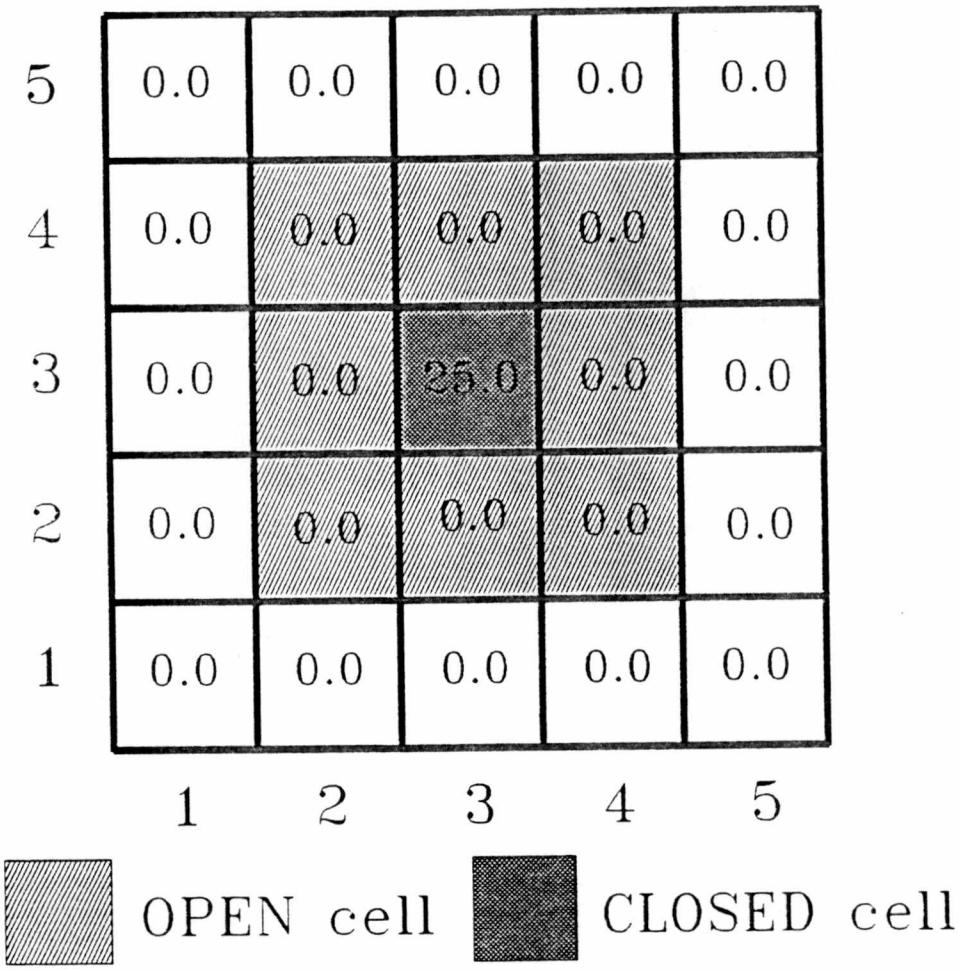



**Figure 3-2: Initial Grid**

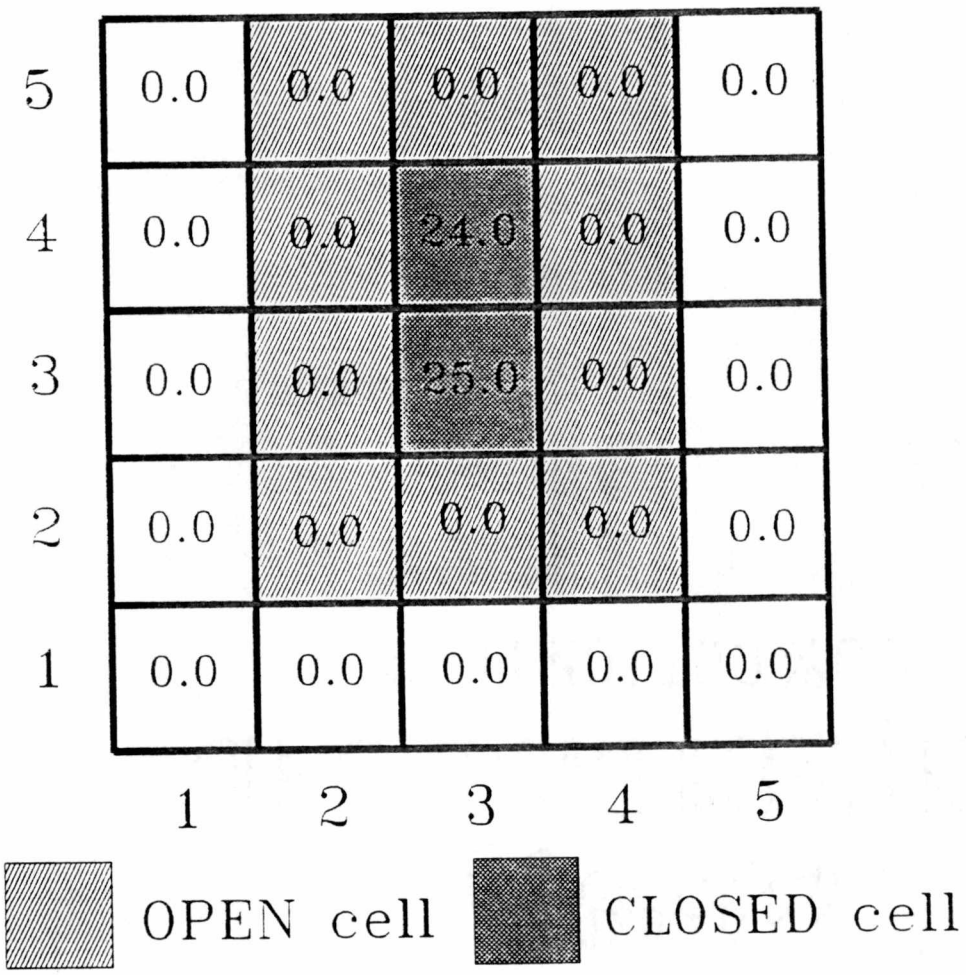



**Figure 3-3: First Cell Value Calculated**

The passable neighbors (5,5), (5,4), and (5,3) are added to O; cell (4,4) is removed and its *open* and *closed* flags are set to **FALSE** and **TRUE**, respectively. The values in the grid after the second iteration are shown in Figure 3-4.

After eight iterations of the process, the first wave of propagation is complete. That is, the all of the original eight cells from O had values assigned to them. As seen in Figure 3-5, O now contains all the cells in the next "ring" out from the goal. When these cells have values assigned to them, there are no more cells in O, and the propagation process is complete (see Figure 3-6).

The path length between (2,1) and the goal cell is equal to the difference of the cells' values multiplied by the size of the cell. The robot traverses the length of 2.4 cells to reach the goal. This is achieved by following either path [(2,1), (2,2), (3,3)], or path [(2,1), (3,2), (3,3)]. However, a robot in cell (2,1) would chose to move to cell (3,2). As will be discussed in Chapter 4, a move is always made to the maximal neighbor of the robot's current cell.

### *3.3 Example of Value Propagation with Obstacles Present*

Cells (3,2) and (4,2) contain obstacles for this example, so the value is set to -1, and the *closed* flag is **TRUE** for each. This also results in cell (2,2) being obstructed,

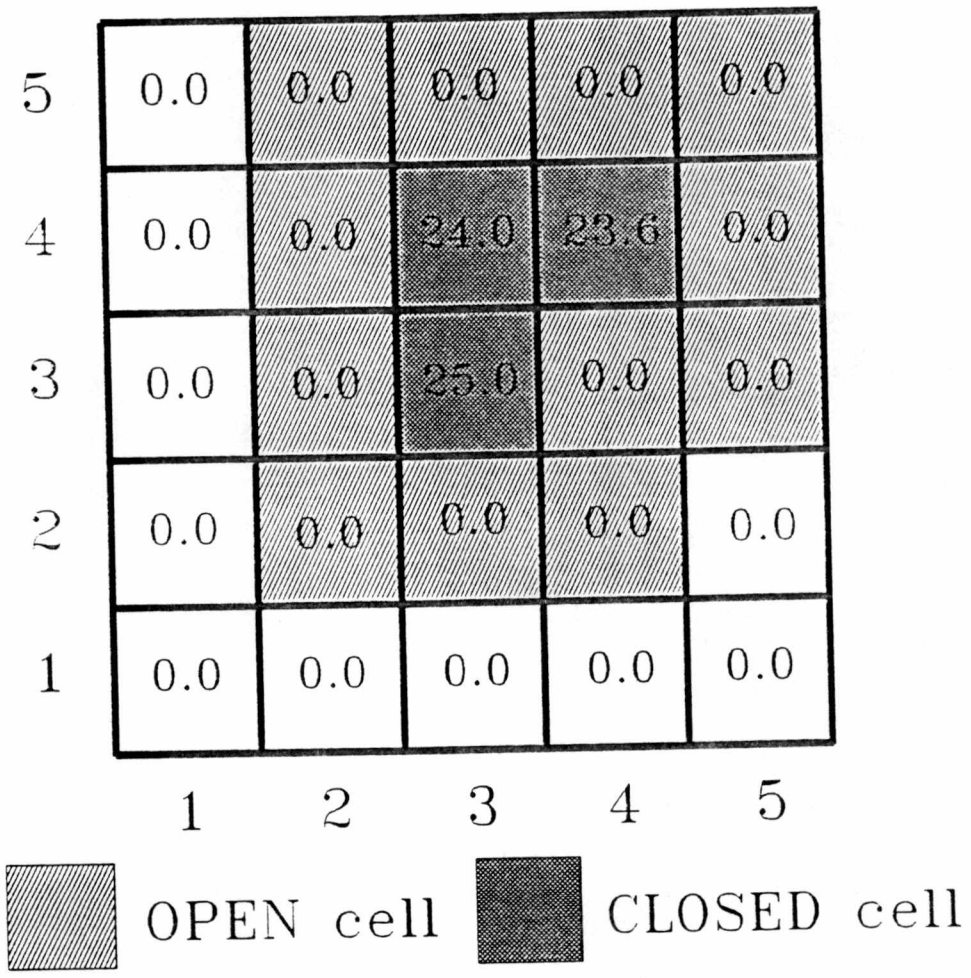



**Figure 3-4: Second Cell Value Calculated**

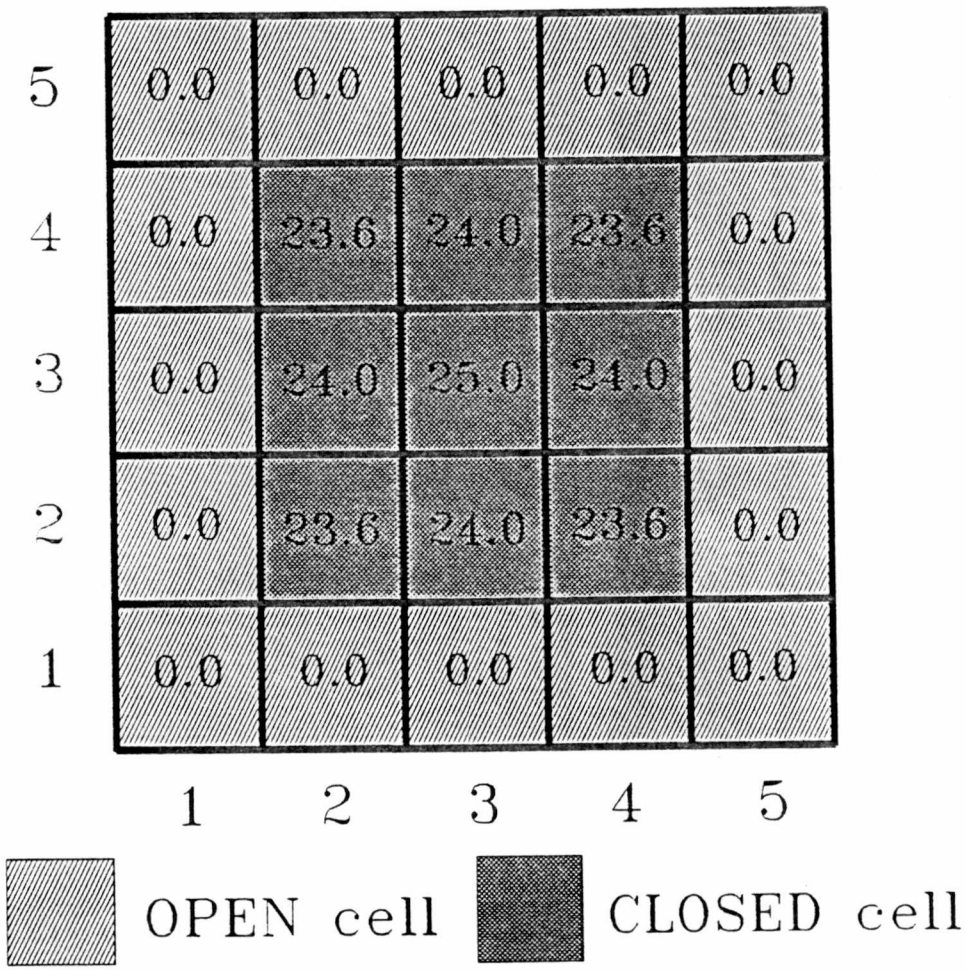



**Figure 3-5: First Propagation Wave Complete**

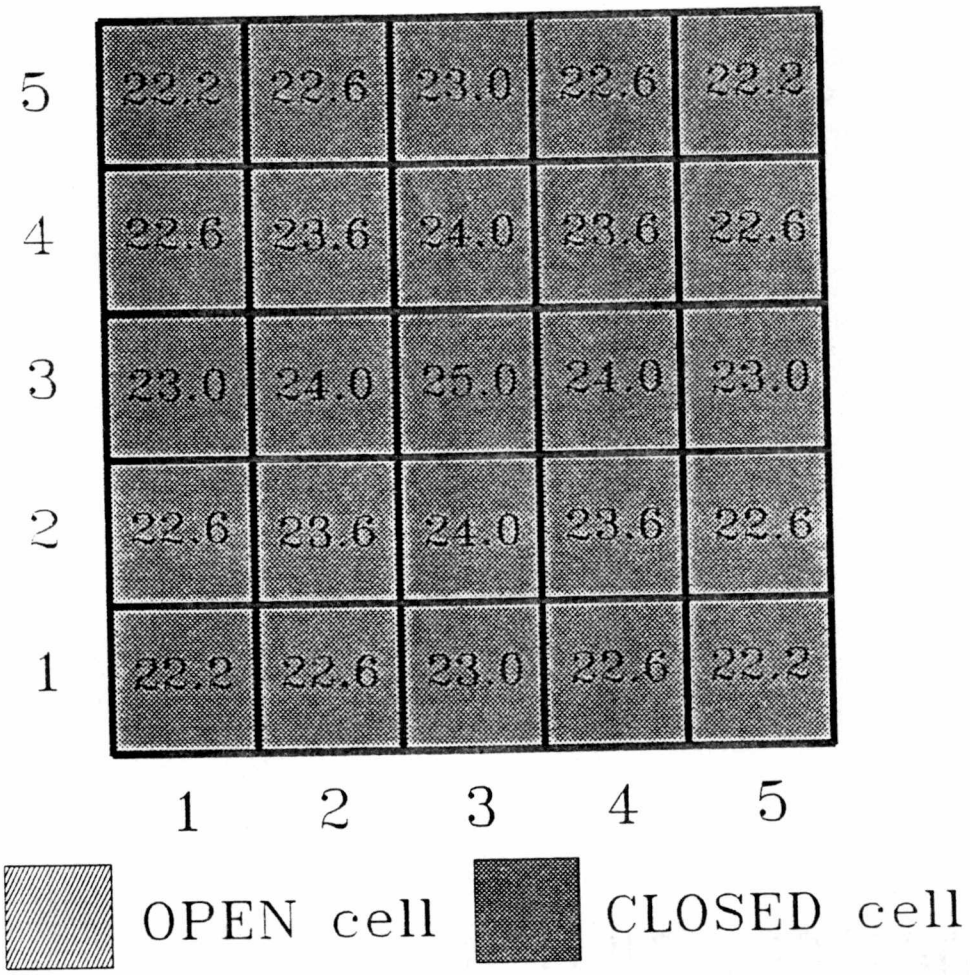



**Figure 3-6: Propagation Complete**

as the robot could not move diagonally between (2,2) and the goal cell. The initial setup is shown in Figure 3-7.

Initially, only 5 cells are in O. Once these cells have values assigned to them, the first wave is complete. A comparison of Figures 3-8 and 3-5 reveals that none of the values of the unobstructed cells have changed. However, O has been affected by the presence of the obstacles.

Cell (2,2) does not receive a value until the second wave of propagation. While cell (3,3) again has the maximum value of all adjacent cells, it cannot be used. Cells (2,2) and (3,3) are obstructed from one another. By definition, (3,3) is not a passable neighbor of (2,2). Therefore, the maximal neighbor of the *current* cell is (2,3), and

$$v(2,2) \leftarrow v(2,3) - \delta(2,2,2,3)$$
$$= 24.0 - 1.0 = 23.0$$

The grid after the completion of the second propagation wave is shown in Figure 3-9.

In the first example all twenty-five cells had values at the end of the second wave of propagation. With obstacles on the grid there are still six cells with a value of zero after the second wave of propagation.

It will take three more propagation waves before all the cell values are calculated. These steps are shown in Figures 3-10, 3-11, and 3-12. show these steps.

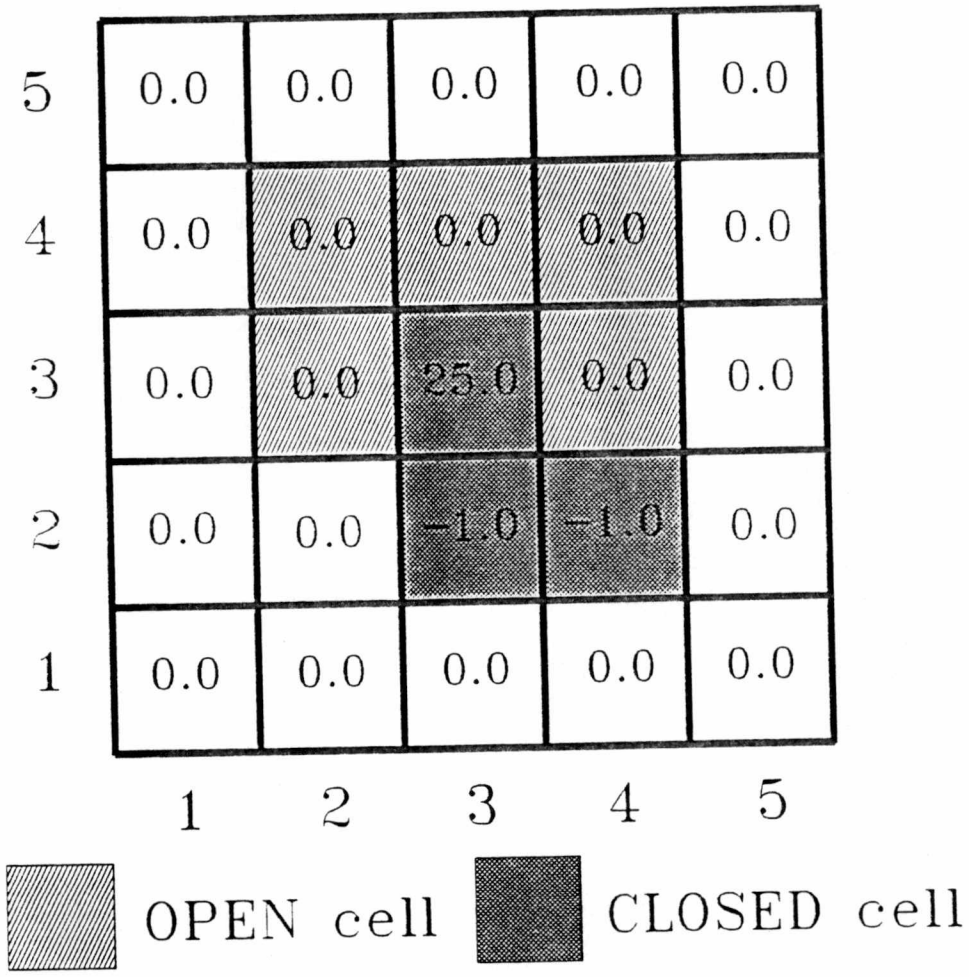



**Figure 3-7: Initial Grid with Two Obstacles**

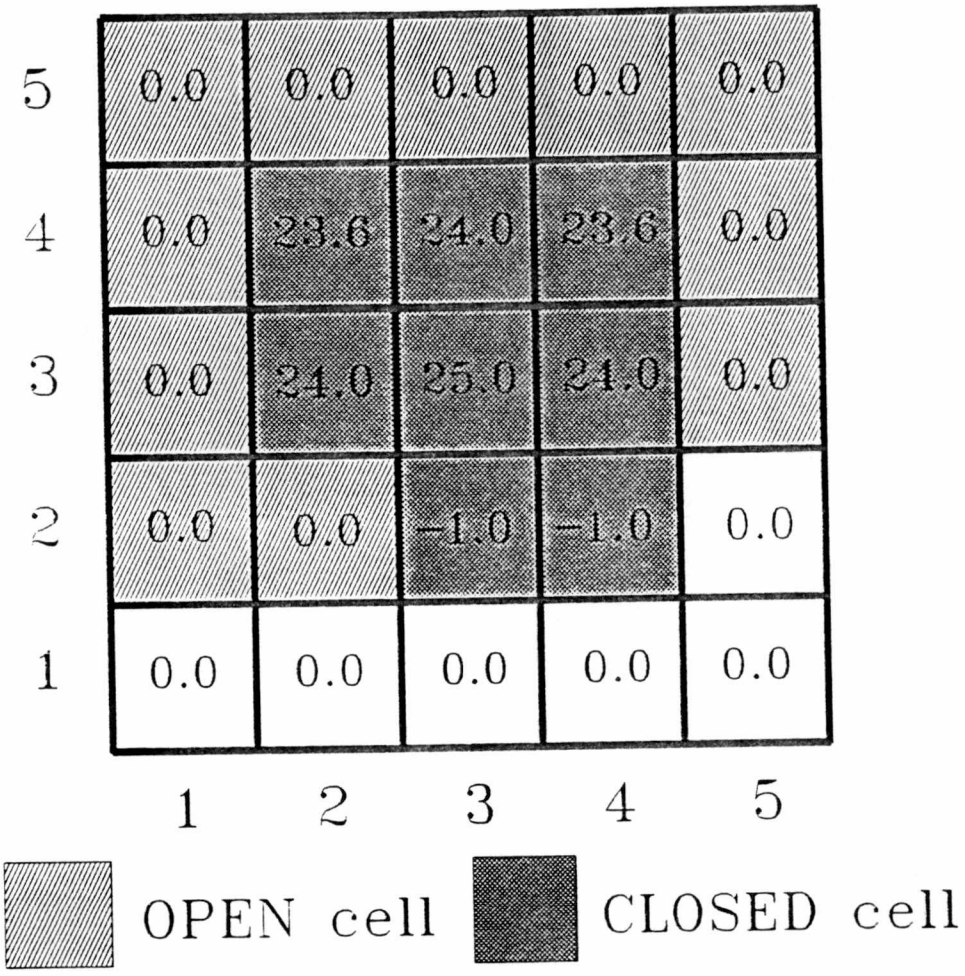



**Figure 3-8: First Wave of Value Propagation with Obstacles**

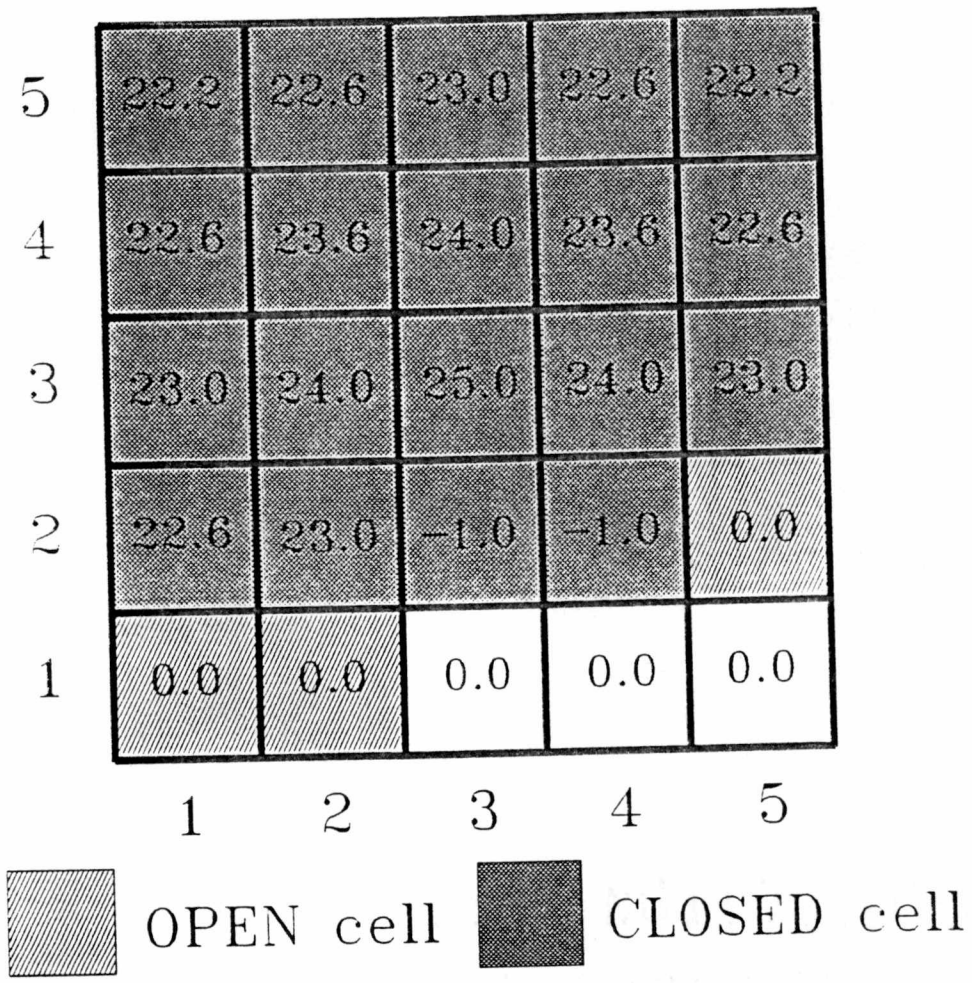



**Figure 3-9: Second Wave of Value Propagation with Obstacles**

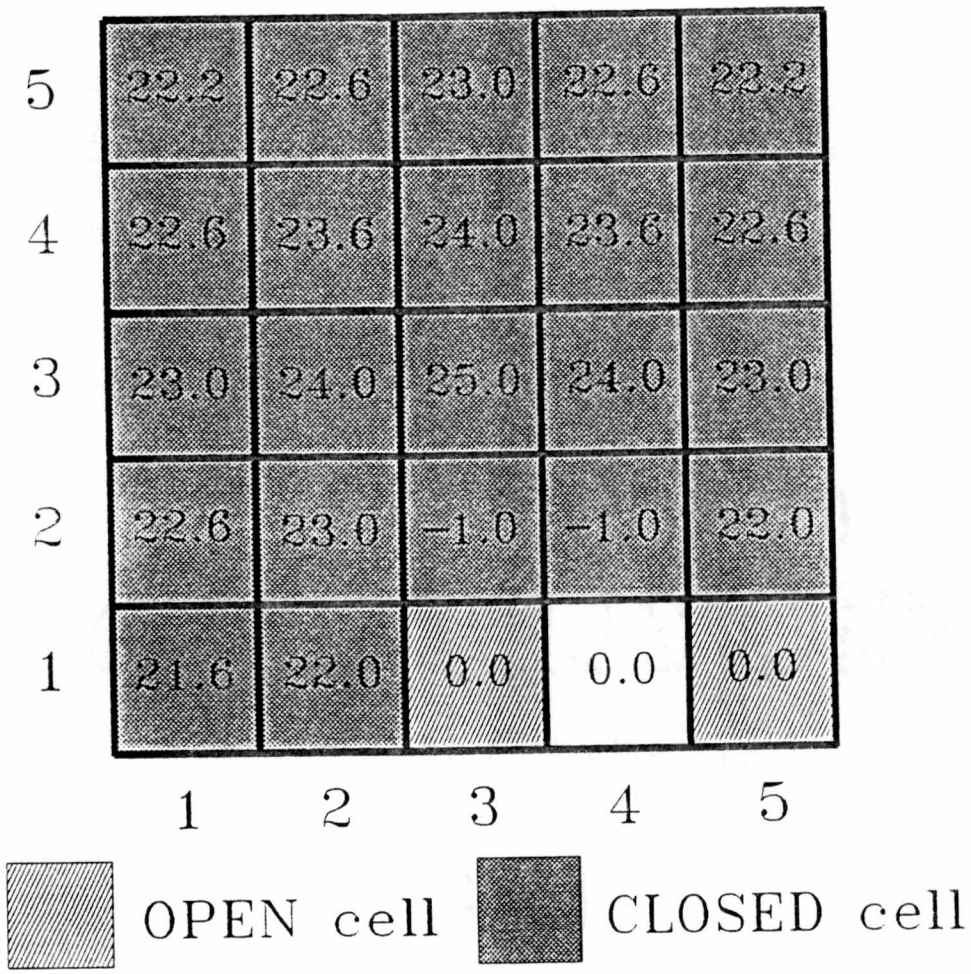



**Figure 3-10: Third Wave of Value Propagation with Obstacles**

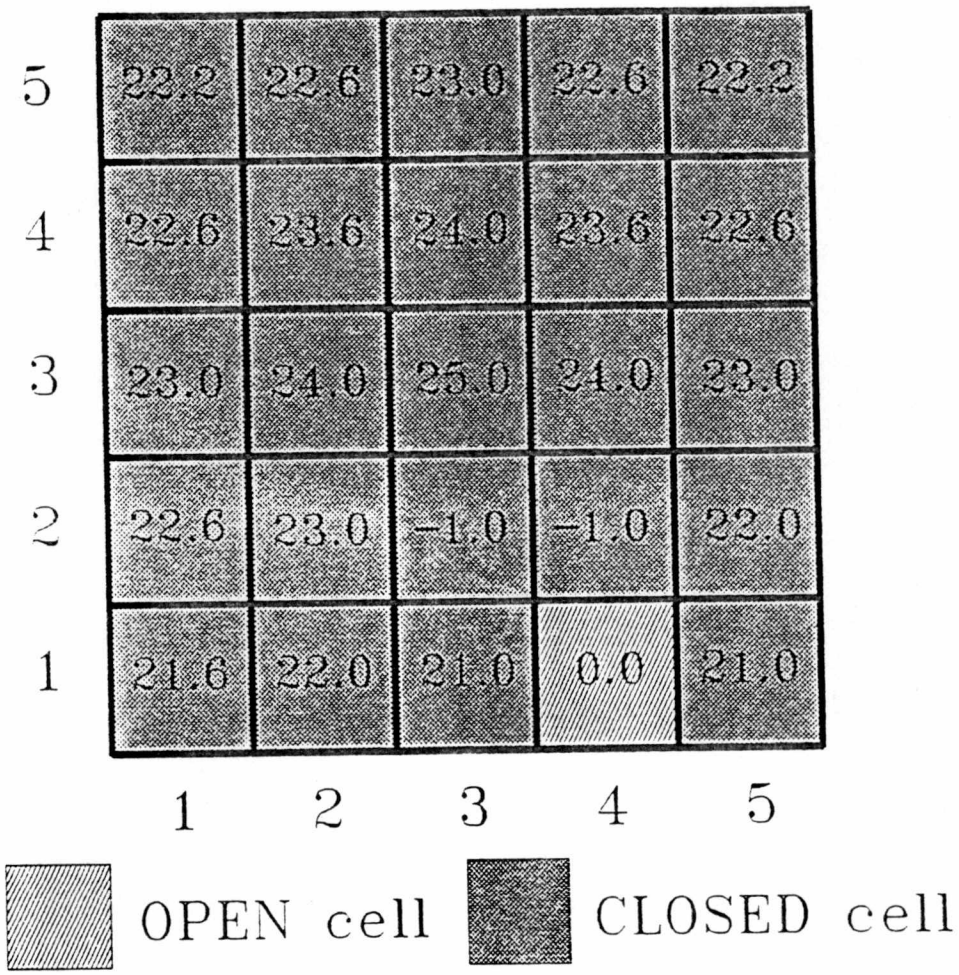



**Figure 3-11: Fourth Wave of Value Propagation with Obstacles**

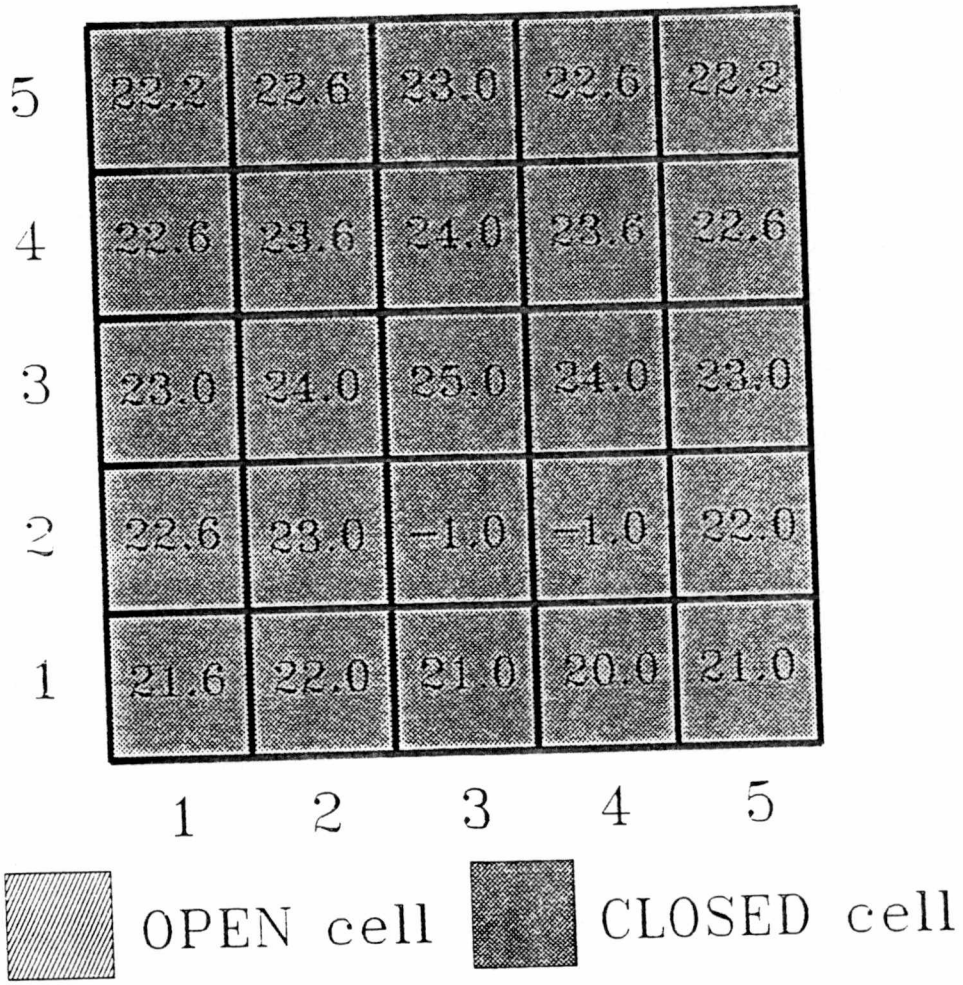



**Figure 3-12: Propagation with Obstacles Complete**

# CHAPTER 4

# ALGORITHM DEFINITION

Having covered the technique of propagation in detail, it is now possible to present a formal definition of the CREEP algorithm as is shown in Figure 4-1. The major steps are discussed in Section 4.1. A brief description of each is provided. The specifics of the scanning, planning, and moving phases of the algorithm, are discussed in Sections 4.2, 4.3, and 4.4, respectively. The concept of a curiosity factor, a modification to the value calculation in the propagation routine, is introduced in Section 4.5. Scenarios where it is useful are presented. A heuristic is introduced to reduce the number of cell values calculated. This heuristic is described in Section 4.6.

## *4.1 Formal Definition of the CREEP Algorithm*

Initially the robot has not performed a sonar sweep of its surroundings, so all *seen* flags are set to **FALSE**. The locations of the goal and robot cells are also established. The propagation routine is invoked to assign cell values. This allows a move to be made in the first pass of the algorithm even if no obstacles are found in the initial scan.

**1.** Initialize

$\forall x \mid 1 \leq x \leq m$ and $\forall y \mid 1 \leq y \leq n$

$\psi(x,y) \leftarrow$ **FALSE**

$v(x,y) \leftarrow 0.0$

$\check{\mathbf{G}} \leftarrow (g1, g2)$

$\check{\mathbf{R}} \leftarrow (r1, r2)$

Do Propagate

**2.** Scan

**NEW_OBSTACLES** $\leftarrow$ **FALSE**

Sonar Sweep

**3.** Plan

If **NEW_OBSTACLES** then

Do Propagate

$(\mathbf{mx}, \mathbf{my}) \leftarrow MN(r1,r2)$

If $v(\mathbf{mx},\mathbf{my}) = 0.0$ then

TERMINATE with *failure*

**4.** Move

$(r1,r2) \leftarrow (\mathbf{mx}, \mathbf{my})$

**5.** If $\check{\mathbf{R}} = \check{\mathbf{G}}$ then

TERMINATE with *success*

else

Go to **2.**

**Figure 4-1: The CREEP Algorithm**

The second step receives sensor data (actual or simulated) and incorporates it into the grid. Any cells that are partially or completely filled by an obstacle are assigned a negative value. Whenever a previously unseen obstacle is found the **NEW_OBSTACLES** flag becomes **TRUE**. The *seen* flag for each cell covered by the sensor sweep is also set to **TRUE**.

If any new obstacles are discovered in the scanning phase, the propagation routine (described in Chapter 3) is invoked. Otherwise, the algorithm proceeds directly to the second part of the planning stage.

When determining where the next move should be made it is first necessary to find the maximal neighbor of the robot, MN(r1,r2). Call this cell **(mx,my)**. If its value, v(**mx**,**my**), is zero then the algorithm terminates and a failure condition is reported. For more explanation of this failure condition refer to Section 4.3.

Step four executes the move found in the previous step. The robot moves from its current location to **(mx,my)**.

The final step of the algorithm checks if the robot has reached the goal location, **G**. If so, execution is halted and success is reported. Otherwise, the scan, plan, move cycle begins again.

### *4.2 Scanning*

The sonar sweep in step 2 is a 360° scan, with readings taken every 15°. Thus, there are twenty-four distance measurements, given in inches, that must be incorporated into the grid model.

A measurement equal to the maximum scanning range indicates that no obstacle was found in that direction. Values less than the maximum imply an obstacle exists the specified distance from the robot.

By applying trigonometric functions, the cell in which cell the detected obstacle lies can be determined easily. In Figure 4-2, the length of the hypotenuse of the right triangle shown is equal to the distance, **d**, returned by the sonar reading. The robot, (r1,r2), is at the origin. The angle, $\theta$, is the direction relative to the grid, in degrees, in which the sonar measurement was made. The positive x-axis corresponds to 0°. The x and y coordinates can be calculated using the following formulas:

$$x = \mathbf{d}\cos(\theta) \qquad \text{and} \qquad y = \mathbf{d}\sin(\theta)$$

If non-integer values are calculated, the values of x and y are truncated.

When an obstacle is first discovered the **NEW_OBSTACLES** flag is set to TRUE. This causes the value propagation routine to be invoked.

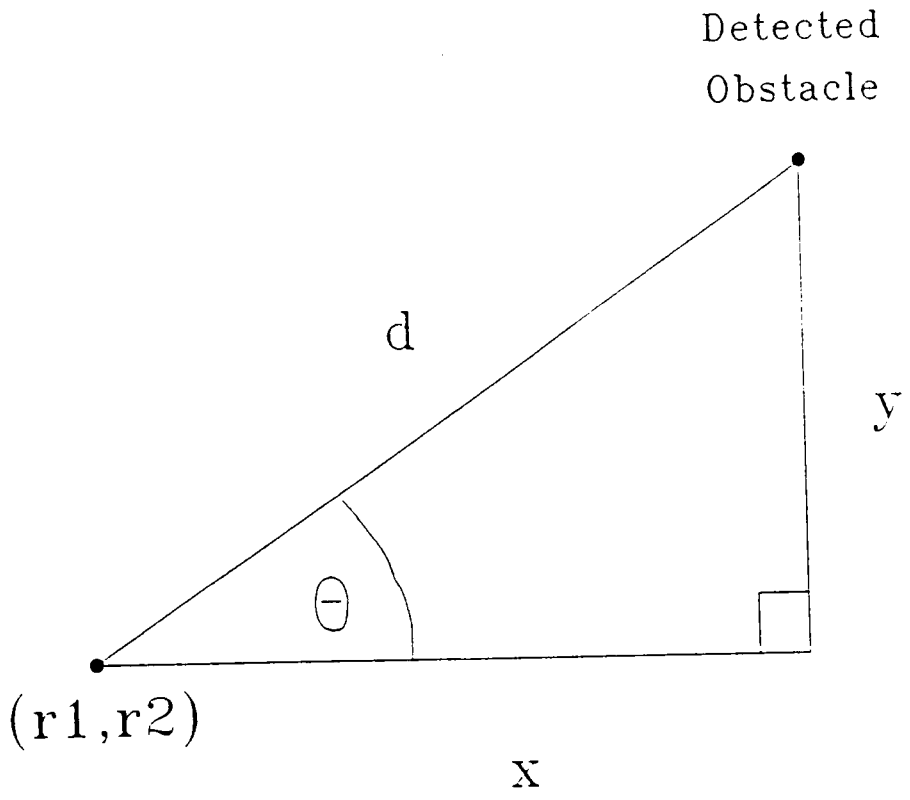



**Figure 4-2: Placing a Detected Obstacle**

### 4.3 Planning

The most significant aspect of the planning stage is the propagation routine, fully described in Chapter 3. Once the propagation is complete (whether it was invoked in this iteration or not), the value of each cell indicates its strength as a candidate path. The higher the value, the shorter the path between that cell and the goal. In order to determine which cell to move to, the robot examines each of its neighbors' values and selects the maximum. A value of zero or less (i.e., a cell containing an obstacle), indicates each adjacent cell has a value of zero or less. Since the maximal neighbor is equal to its value at the beginning of the routine, no new values were propagated to the robot. That is, the propagation wavefront did not reach the robot. Therefore, no opening exists between the robot and the goal, and the algorithm terminates in failure.

If the value of the maximal neighbor is greater than zero, then at least one path remains open between the robot and the goal. In order for the robot to explore the most promising path (based on incomplete information), it should move to the cell that is its maximal neighbor.

It is important to record which direction the robot is facing. The first reading of a scan is always taken directly in front of the robot (0° relative to the robot). On the grid, the positive x-axis represents 0° (see Figure 4-3).

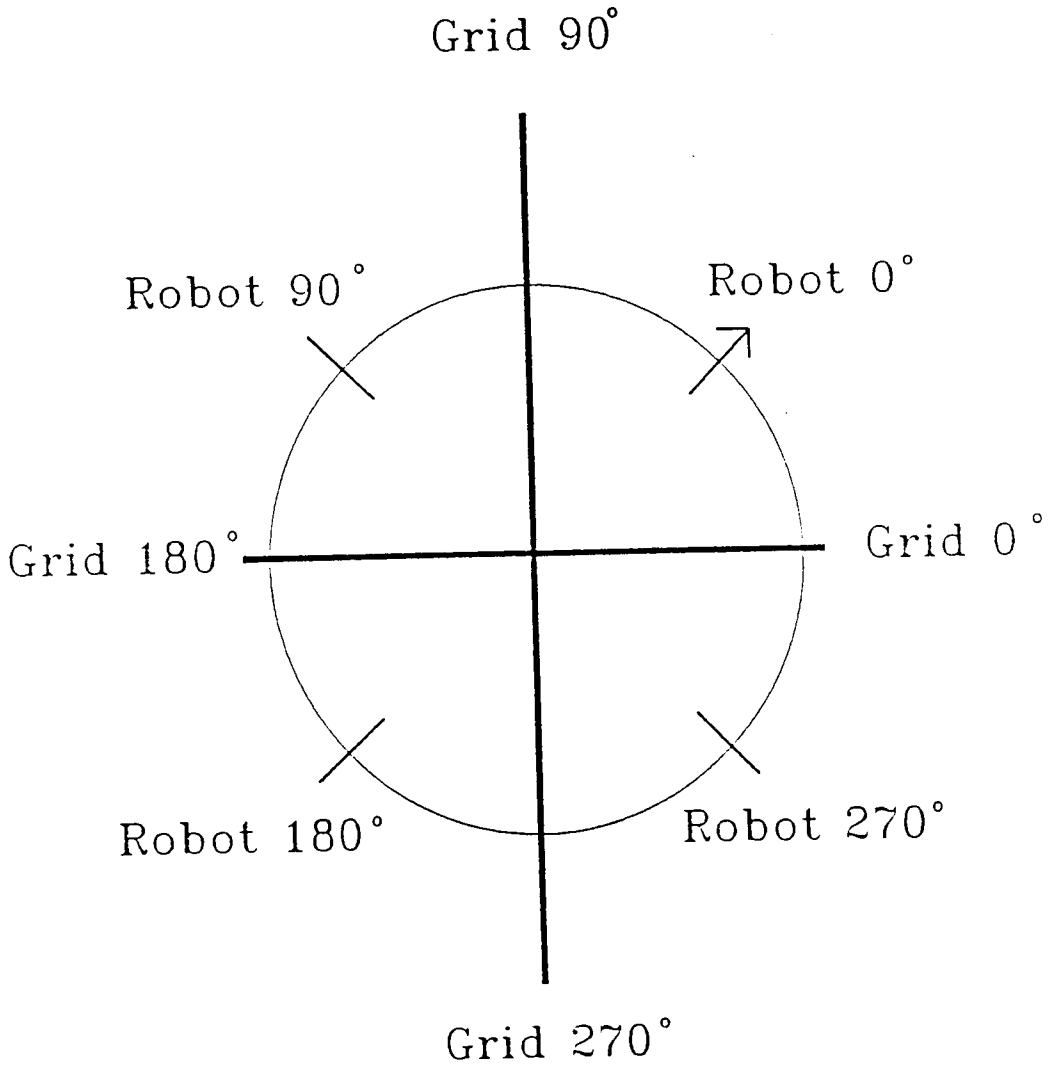



**Figure 4-3: Robot Heading is Forty-Five Degrees**

Thus, in order to correctly incorporate sonar data, the two different directions must be reconciled.

For example, if the robot's heading is 45° when a sonar sweep is performed, the reading used to update the grid at 0° is taken from the robot's 315° measurement. In general,the proper adjustment can be calculated:

$$\gamma = 360 - \rho \qquad [4.1]$$

where, $\gamma$ is the grid degree measurement and $\rho$ corresponds to the robot heading.

### *4.4 Moving*

The cell the robot will move to is determined in the planning phase of the algorithm. If the destination cell lies in the direction the robot is facing, the robot simply moves forward far enough to place it in the center of the destination cell. Otherwise, the robot must first turn to face the direction of its destination prior to moving.

Each cell has at most eight adjacent cells. There are only eight possible directions the robot can travel from one cell to a passable neighbor. Thus, the length of the planned path only approximates the true best path. Mitchell [1987] states, "using an 8-connected grid leads to an

approximation $d$ of Euclidean length $d_E$ such that:

$$d_E \leq d \leq 1.082 d_E \text{ ."}$$

That is, a cell-to-cell path between two points will be no more than 8.2 percent longer than following a straight line connecting the points.

### *4.5 The Curiosity Factor*

Once the robot reaches the goal, some action is usually performed. The robot may need to return to its starting location in order to successfully complete its objective. For example, supplies or data samples may need to be retrieved, reconnaissance photographs delivered, or experiment results reported.

This section assumes the robot has reached its initial goal and needs to return to its original starting location. A path shorter than the one used to arrive at the current location may exist, but it runs through unexplored cells. Undetected obstacles might block the way.

When the robot has little or no information about a region, the possibility of finding a dead end exists. This risk must be weighed against the potential gain an open path in the unexplored area may provide.

There are many scenarios where minimizing the time or distance travelled is critical. For example, if the robot

is retrieving emergency medical supplies, unnecessary delays may be fatal. If the robot's battery level is low, wasting energy in exploration could strand the robot.

In such critical situations it would be better to retrace a known path rather than attempt to make a small savings (in time or distance) by exploring a new region. The exploration itself is time consuming, but finding that the new path is blocked could be disastrous.

A simple way to avoid excursions into unknown regions is to introduce a penalty when calculating values for cells that have not been scanned. This penalty will be called the **curiosity factor** and is calculated

$$\varsigma \leftarrow \frac{\sum_{x,y} \beta( \nu(x,y), -1.0 )}{\sum_{x,y} \beta( \psi(x,y), \text{TRUE} )} \qquad \textbf{[4.1]}$$

Thus, the curiosity factor is simply the ratio of obstacles detected to the number of cells scanned. This ratio reflects the expectation probability, based on past experience, of finding obstacles in the unknown region.

In order to incorporate this factor into value propagation it is necessary to redefine formula **[3.1]** as follows:

$$\nu(\mathbf{cx},\mathbf{cy}) \leftarrow \nu(\mathbf{mx},\mathbf{my}) - (\varsigma + \delta(\mathbf{cx},\mathbf{cy},\mathbf{mx},\mathbf{my})) \qquad \textbf{[4.2]}$$

The number of obstacles found and cells scanned are unchanging once the value propagation begins. Accordingly, the curiosity factor is calculated when propagation is invoked, not each time a cell value is calculated. Examples demonstrating the curiosity factor are given in Chapter 6.

If no obstacles have been found, the curiosity factor is zero and the cell value calculations are unaffected. The more cluttered the environment, the higher the curiosity factor becomes. In a map with many obstacles, the returning robot will be less likely to attempt exploration of an unknown region than in a less cluttered environment.

### *4.6 Reducing Cell Value Calculations*

The propagation routine, described in Chapter 3, calculates a value for each cell the propagation wave encounters. Reducing the number of cell value calculations performed speeds the path planning process, thereby decreasing the overall execution time of the algorithm.

A heuristic rule, intended to reduce calculations in the propagation routine has been investigated. An initial call to the propagation routine assigns values to all cells in the grid. Subsequent invocations of the propagation routine apply the heuristic rule, which restricts cell value calculations only to cells the robot has scanned. The propagation routine, as previously defined, calculates values for both scanned and unscanned cells.

The curiosity factor, described in Section 4.5, and the cell value calculation saver, described in this section, are contrasted with the formal definition of the CREEP algorithm in Chapter 6. Several examples are presented demonstrating the effects of these two modifications. An analysis of the different methods is also provided at the end of Chapter 6.

# CHAPTER 5

# ALGORITHM ANALYSIS

An analysis of the CREEP algorithm is presented in this chapter. A discussion of the advantages provided by the wavelike propagation technique follows the formal analysis.

## *5.1 Formal Analysis*

The CREEP algorithm consists of two loops, one imbedded in the other. The inner loop is the propagation routine, the outer, the main algorithm, itself.

The propagation routine is analyzed in terms of the number of cells processed. The grid is finite, and each cell is processed only once. Thus, the time complexity of value propagation is O(n), where n is the total number of cells in the grid.

The CREEP algorithm consists of three steps: scan, plan, and move. The scan and move routines are simple routines and run on order O(1). If no new obstacles are detected, the plan routine is also O(1). However, each time a new obstacle is discovered, the propagation process is invoked. Clearly, in a grid with n cells, obstacles will be detected in no more that n-1 cells (the robot occupies one cell). Hence the outer loop of the algorithm operates on

order O(n-1). This is equivalent to running in O(n) time.

The time complexity of nested loops are multiplied to determine the overall complexity of the algorithm. Thus, the CREEP algorithm operates on order $O(n \times n) = O(n^2)$.

### *5.2 Wavelike Qualities of the Value Propagation in the CREEP Algorithm*

The wavelike manner of value propagation has many strengths. First, if there is an opening between the robot and the goal, a wave will reach the robot. That is, if a path exists, this algorithm will find it.

PROOF

Since the grid has a finite number of cells, if a path exists, it must be of finite length, $\ell$. Here $\ell$ is a measure of the number of cells traversed and not the distance travelled. The first propagation wave examines all paths of length 1, the second, all paths of length 2, the third, 3, and so on. The propagation terminates on one of two conditions. In the first, termination caused by exhausting all members of O implies no path exists. A path is assumed to exist, so this case is disregarded. In the second case, if all neighbors of the robot are *closed*, a wave has reached the robot (i.e., all paths of length $\ell$ have been examined) and a path has been found. The proof is complete.

Second, based on current grid information the robot will explore the most direct route to the goal. However, the robot will not necessarily follow the shortest global

path, but instead, will pursue the path that appears shortest *based on the obstacles it has already encountered.*

PROOF

The first-in first-out constraint of operations on the open list, O, forces the graph search to be breadth-first. A breadth-first search is guaranteed to find the shortest solution path [Nilson, 1980].

Third, recovery from a dead end is easily accomplished. If the path currently under exploration is a dead end, one or more new obstacles will be detected that block the path the robot was exploring. The detection of obstacles always invokes the propagation routine, which will cause new calculations to be made incorporating new obstacles data. The new cell values will lead the robot out of the dead end to explore the next best path. Thus, no special exception handling routine or dead end recovery strategies are necessary.

Fourth, the wavelike motion accounts for distance needed to travel around known obstacles. As a wave encounters and moves around an obstacle, its strength decreases. The decrease is directly proportional to the increased distance the robot must travel to move around the obstacle.

Finally, when two waves meet after being split by one or more obstacles, only the stronger of them will survive. This results from assigning a value to a cell based on the value of its maximal neighbor.

# CHAPTER 6

# CREEP SIMULATION AND ANALYSIS

The CREEP algorithm has been simulated on a Northgate 286 computer using the C programming language. Five different obstacle configurations are used for the examples presented in this chapter. Each configuration has four cases associated with it. The first case runs the unmodified algorithm described in Chapters 3 and 4. The second case implements the curiosity factor, described in Section 4.5. The cell value calculation saver, described in Section 4.6, is implemented for the third case of each configuration. Finally, the curiosity factor and cell value calculation saver are combined for Case 4.

Figures associated with the different examples will include an identifier in the caption, "Example n, Case m." This means the figure belongs to Example number n, Case number m. If multiple case numbers are provided it means the same path is followed for each case listed.

An analysis of all simulation examples is presented at the end of this chapter. A table lists statistics for each of the twenty example cases.

In the figures, obstacles are the black rectangles. The robot location is represented by a gray box with an

arrow indicating its heading. Arrows appearing by themselves mark the path of the robot. The direction of the arrow indicates the heading of the robot when it entered that cell. The goal is depicted as a plain gray box. The same symbols will be used in all the graphics figures in this chapter.

Several figures show the cell values at different points during the robot's travel. Cells containing obstacles have a value of -1. The goal value is shown having a value of 99.

The goal value of 99 shown in these examples does not reflect the value stored in the grid. The **GOAL_VALUE** constant is actually 23 × 25 = 575. However, the grid values are reduce by 476, and all decimal values are truncated before displaying. Thus, all numbers displayed on the screen are, at most, two digits long. This preserves an on-screen correlation between the cells of the text and graphics displays.

The propagation routine terminates when all of the robot's passable neighbors have been assigned values (or when **O** is empty). Hence, many cells may not be assigned values during propagation; these cells are shown with a value of zero.

### *6.1 Example One*

The complete obstacle layout used in the first example is shown in Figure 6-1. This configuration will demonstrate the flexibility of the algorithm. The robot will enter and explore the two inverted U-shaped enclosures in its attempt to reach the goal. Once these enclosures are explored and found not to lead to the goal, the robot leaves them and continues along another route to the goal.

The 15° spacing between sonar readings results in a zig-zag in the robot's path (Figure 6-2). From cell (20,12) the robot detected obstacles in cells (18,18), (20,18), and (22,18). Cells (19,18) and (21,18) were unscanned by the sonar. The robot moves so it can go through the gap in cell (21,19), but finds it to contain an obstacle in the subsequent sonar scan. Now the path through cell (19,18) appears better, but this path is also found to be blocked.

The robot eventually reaches the goal. Once there, the robot's original starting location is established as the new goal, and the robot attempts to return to this new goal. All obstacle information is maintained. The return path (Figure 6-3) shows how the robot avoids repeating the exploration needed in the initial trip. Since nearly complete information about the robot's surroundings is now available, the return path is much shorter. This demonstrates how the algorithm incorporates previous obstacle information.

The curiosity factor is used for Case 2 of this

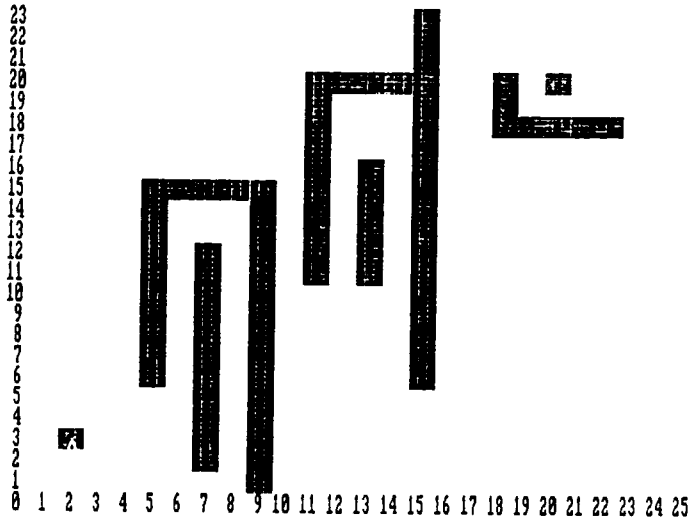


Figure 6-1: Example 1, Case 1,2,3,4 Obstacle Configuration

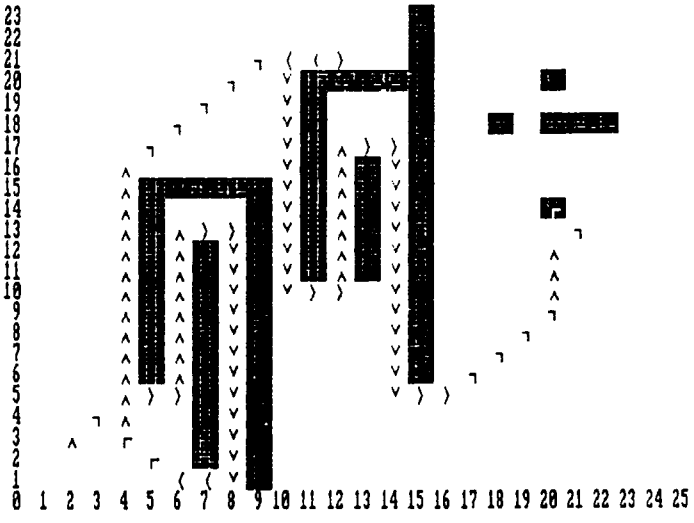


Figure 6-2: Example 1, Case 1,2,3,4 Cause of a Zig-Zag

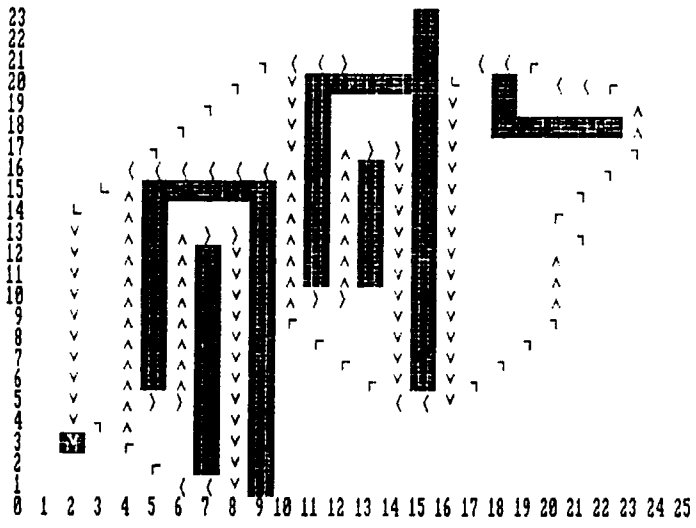


**Figure 6-3: Example 1, Case 1,3,4 The Complete Path**

obstacle configuration. The only deviation from the path in Cases 1, 3, and 4 is shown in Figure 6-4. Since fewer unscanned cell values are calculated for the path passing through the opening at cell (11,16), that path is the most promising. However, once an obstacle is detected there, a path very similar to the one generated by the other three cases is followed. The return trip (Figure 6-5) is identical.

### *6.2 Example Two*

The purpose of this example is to demonstrate the effect of the curiosity factor. In reaching the initial goal, all four cases result in the path shown in Figure 6-6.

At this point a decision must be made whether to return by the entirely known path or to move through an unexplored region with the opportunity of making a small savings in path length. Case 1 does not use the curiosity factor. Examining the cell values (Figure 6-7) reveals that the path through the unexplored region appears slightly better than the path entirely know to the robot. Cell (18,2) is not known to contain an obstacle. Thus, the robot could move diagonally from (19,2) to (18,1). However, cells (2,19) and (1,18) are obstructed from each other by the obstacle in cell (2,18). Based on the current information available to the robot, the path through the unexplored region is better.

However, while exploring the new territory, the robot

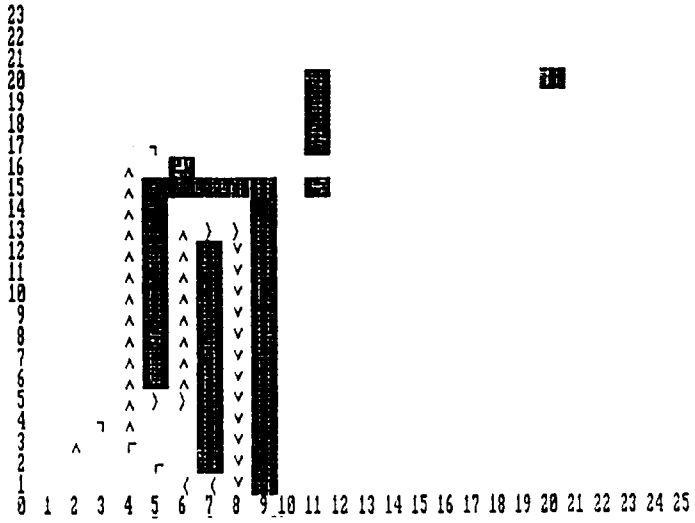


**Figure 6-4: Example 1, Case 2 A Path Deviation**

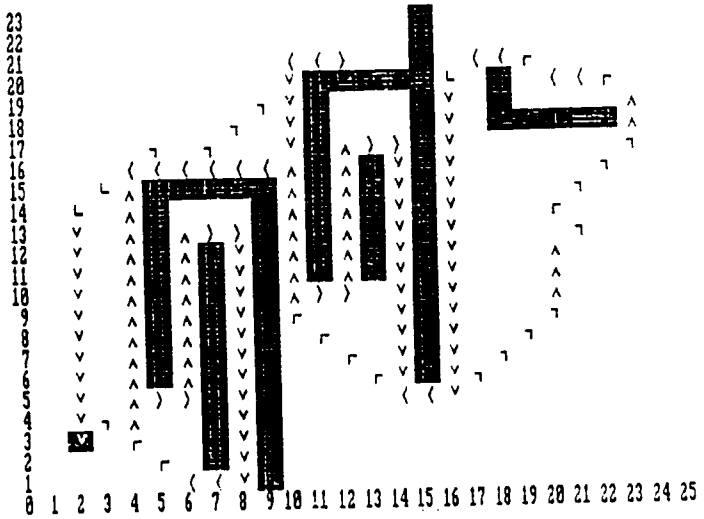


**Figure 6-5: Example 1, Case 2 The Complete Path**

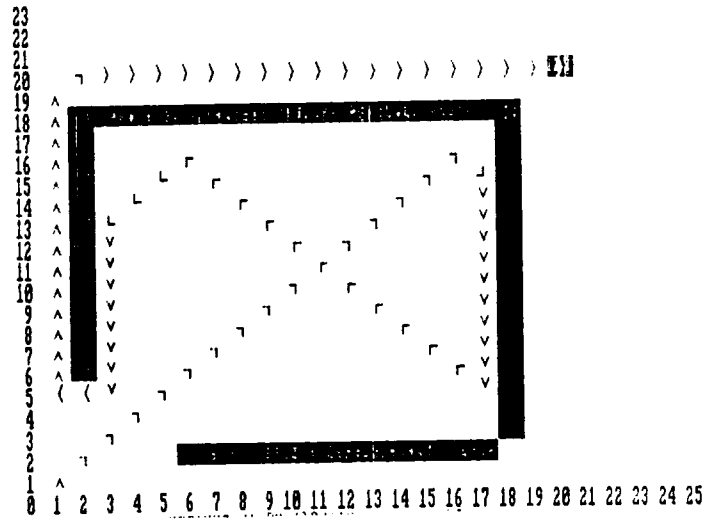


**Figure 6-6: Example 2, Case 1,2,3,4 At The Initial Goal**

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 23 | 77 | 76 | 76 | 75 | 75 | 74 | 73 | 72 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 | 63 | 62 | 61 | 60 | 0 | 0 | 0 | 0 | 0 |
| 22 | 78 | 77 | 77 | 76 | 75 | 74 | 73 | 72 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 | 63 | 62 | 61 | 60 | 0 | 0 | 0 | 0 | 0 |
| 21 | 79 | 78 | 78 | 77 | 76 | 75 | 74 | 73 | 72 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 | 63 | 62 | 61 | 60 | 0 | 0 | 0 | 0 |
| 20 | 80 | 79 | 78 | 77 | 76 | 75 | 74 | 73 | 72 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 | 63 | 62 | 62 | 61 | 61 | 60 | 60 | 60 |
| 19 | 81 | 80 | 79 | 78 | 77 | 76 | 75 | 74 | 73 | 72 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 | 63 | 63 | 62 | 62 | 61 | 61 | 61 |
| 18 | 82 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 64 | 64 | 63 | 63 | 62 | 62 | 62 |
| 17 | 83 | -1 | 82 | 81 | 81 | 80 | 80 | 80 | 79 | 79 | 78 | 78 | 78 | 77 | 77 | 76 | 76 | -1 | 65 | 65 | 64 | 64 | 63 | 63 | 63 |
| 16 | 84 | -1 | 83 | 82 | 82 | 81 | 81 | 81 | 80 | 80 | 79 | 79 | 79 | 78 | 78 | 77 | 76 | -1 | 66 | 66 | 65 | 65 | 64 | 64 | 64 |
| 15 | 85 | -1 | 84 | 83 | 83 | 82 | 82 | 82 | 81 | 81 | 80 | 80 | 80 | 79 | 79 | 78 | 77 | -1 | 67 | 67 | 66 | 66 | 65 | 65 | 65 |
| 14 | 86 | -1 | 85 | 84 | 84 | 83 | 83 | 83 | 82 | 82 | 81 | 81 | 81 | 80 | 79 | 78 | 77 | -1 | 68 | 68 | 67 | 67 | 66 | 66 | 66 |
| 13 | 87 | -1 | 86 | 85 | 85 | 84 | 84 | 84 | 83 | 83 | 82 | 82 | 82 | 81 | 80 | 79 | 78 | -1 | 69 | 69 | 68 | 68 | 67 | 67 | 67 |
| 12 | 88 | -1 | 87 | 86 | 86 | 85 | 85 | 85 | 84 | 84 | 83 | 83 | 82 | 81 | 80 | 79 | 78 | -1 | 70 | 70 | 69 | 69 | 68 | 68 | 68 |
| 11 | 89 | -1 | 88 | 87 | 87 | 86 | 86 | 86 | 85 | 85 | 84 | 83 | 82 | 81 | 80 | 79 | 78 | -1 | 71 | 71 | 70 | 70 | 69 | 69 | 69 |
| 10 | 90 | -1 | 89 | 88 | 88 | 87 | 87 | 87 | 86 | 86 | 85 | 84 | 83 | 82 | 81 | 80 | 79 | -1 | 72 | 72 | 71 | 71 | 70 | 70 | 70 |
| 9 | 91 | -1 | 90 | 89 | 89 | 88 | 88 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 | 79 | -1 | 73 | 73 | 72 | 72 | 71 | 71 | 71 |
| 8 | 92 | -1 | 91 | 90 | 90 | 89 | 89 | 89 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 | -1 | 74 | 74 | 73 | 73 | 72 | 72 | 72 |
| 7 | 93 | -1 | 92 | 91 | 91 | 90 | 90 | 89 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 | -1 | 75 | 75 | 74 | 74 | 73 | 73 | 72 |
| 6 | 94 | -1 | 93 | 92 | 92 | 91 | 90 | 89 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 | -1 | 76 | 76 | 75 | 75 | 74 | 73 | 72 |
| 5 | 95 | 94 | 94 | 93 | 93 | 92 | 91 | 90 | 89 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | -1 | 77 | 77 | 76 | 76 | 75 | 74 | 73 |
| 4 | 96 | 95 | 95 | 94 | 93 | 92 | 91 | 90 | 89 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | -1 | 78 | 78 | 77 | 76 | 75 | 74 | 73 |
| 3 | 97 | 96 | 96 | 95 | 94 | 93 | 92 | 91 | 90 | 89 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | -1 | 79 | 79 | 78 | 77 | 76 | 75 | 74 |
| 2 | 98 | 97 | 96 | 95 | 94 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 81 | 80 | 79 | 78 | 77 | 76 | 75 | 74 |
| 1 | 99 | 98 | 97 | 96 | 95 | 94 | 93 | 92 | 91 | 90 | 89 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 | 79 | 78 | 77 | 76 | 75 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

**Figure 6-7: Example 2, Case 1,3 The Return Trip Values**

discovers an obstacle not only in cell (18,2), but one completely blocking its path in cell (9,1). This forces the robot to backtrack and return by the path it initially rejected (Figure 6-8).

Cases 3 and 4 (reduced value calculations and the curiosity factor combined with reduced value calculations) have paths identical to the one shown in Figure 6-8. Case 3 does not use the curiosity factor so this is understandable.

Case 4 uses the curiosity factor, but since cell values calculations are performed only on scanned cells, the penalty is not incorporated into the grid cell values. Only if the robot becomes stuck, as described in Section 4.6, and a full propagation is performed, would the curiosity factor have an impact.

By implementing the curiosity factor in the situation shown in Figure 6-6, the backtracking done in the previous cases can be avoided. The underlying cell values are shown in Figure 6-9. The obstacle in cell (18,2) is still unknown to the robot. Since the curiosity factor penalizes travel through unexplored cells, the path through the known territory is better. The robot returns to its starting location by moving in the familiar area (Figure 6-10).

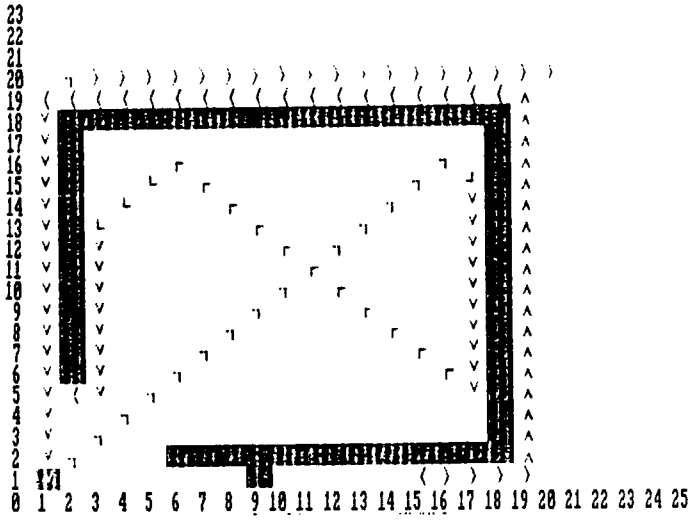


Figure 6-8: Example 2, Case 1,3,4 The Complete Path

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 23 | 77 | 76 | 76 | 75 | 75 | 74 | 73 | 72 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 | 63 | 62 | 61 | 60 | 0 | 0 | 0 | 0 | 0 |
| 22 | 78 | 77 | 77 | 76 | 75 | 74 | 73 | 72 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 | 63 | 62 | 61 | 60 | 0 | 0 | 0 | 0 | 0 |
| 21 | 79 | 78 | 78 | 77 | 76 | 75 | 74 | 73 | 72 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 | 63 | 62 | 61 | 60 | 0 | 0 | 0 | 0 |
| 20 | 80 | 79 | 78 | 77 | 76 | 75 | 74 | 73 | 72 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 | 63 | 62 | 61 | 60 | 59 | 58 | 57 | 57 |
| 19 | 81 | 80 | 79 | 78 | 77 | 76 | 75 | 74 | 73 | 72 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 | 63 | 62 | 61 | 60 | 59 | 58 | 58 |
| 18 | 82 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 61 | 61 | 60 | 60 | 60 | 59 | 59 |
| 17 | 83 | -1 | 82 | 81 | 81 | 80 | 80 | 80 | 79 | 79 | 78 | 78 | 78 | 77 | 77 | 76 | 76 | -1 | 62 | 62 | 61 | 61 | 61 | 60 | 60 |
| 16 | 84 | -1 | 83 | 82 | 82 | 81 | 81 | 81 | 80 | 80 | 79 | 79 | 79 | 78 | 78 | 77 | 76 | -1 | 63 | 63 | 62 | 62 | 62 | 61 | 60 |
| 15 | 85 | -1 | 84 | 83 | 83 | 82 | 82 | 82 | 81 | 81 | 80 | 80 | 80 | 79 | 79 | 78 | 77 | -1 | 64 | 64 | 63 | 63 | 63 | 62 | 61 |
| 14 | 86 | -1 | 85 | 84 | 84 | 83 | 83 | 83 | 82 | 82 | 81 | 81 | 81 | 80 | 79 | 78 | 77 | -1 | 65 | 65 | 64 | 64 | 64 | 63 | 62 |
| 13 | 87 | -1 | 86 | 85 | 85 | 84 | 84 | 84 | 83 | 83 | 82 | 82 | 82 | 81 | 80 | 79 | 78 | -1 | 66 | 66 | 66 | 65 | 64 | 64 | 64 |
| 12 | 88 | -1 | 87 | 86 | 86 | 85 | 85 | 85 | 84 | 84 | 83 | 83 | 82 | 81 | 80 | 79 | 78 | -1 | 67 | 67 | 66 | 66 | 66 | 65 | 65 |
| 11 | 89 | -1 | 88 | 87 | 87 | 86 | 86 | 86 | 85 | 85 | 84 | 83 | 82 | 81 | 80 | 79 | 78 | -1 | 69 | 68 | 68 | 67 | 67 | 66 | 66 |
| 10 | 90 | -1 | 89 | 88 | 88 | 87 | 87 | 87 | 86 | 86 | 85 | 84 | 83 | 82 | 81 | 80 | 79 | -1 | 70 | 69 | 69 | 68 | 68 | 67 | 67 |
| 9 | 91 | -1 | 90 | 89 | 89 | 88 | 88 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 | 79 | -1 | 71 | 70 | 70 | 69 | 69 | 68 | 68 |
| 8 | 92 | -1 | 91 | 90 | 90 | 89 | 89 | 89 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 | -1 | 72 | 71 | 71 | 70 | 70 | 70 | 69 |
| 7 | 93 | -1 | 92 | 91 | 91 | 90 | 90 | 89 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 | -1 | 73 | 73 | 72 | 72 | 71 | 71 | 70 |
| 6 | 94 | -1 | 93 | 92 | 92 | 91 | 90 | 89 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 | -1 | 74 | 74 | 73 | 73 | 72 | 71 | 70 |
| 5 | 95 | 94 | 94 | 93 | 93 | 92 | 91 | 90 | 89 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | -1 | 75 | 75 | 74 | 74 | 73 | 72 | 70 |
| 4 | 96 | 95 | 95 | 94 | 93 | 92 | 91 | 90 | 89 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | -1 | 76 | 76 | 75 | 74 | 73 | 72 | 71 |
| 3 | 97 | 96 | 96 | 95 | 94 | 93 | 92 | 91 | 90 | 89 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | -1 | 77 | 77 | 76 | 75 | 74 | 73 | 71 |
| 2 | 98 | 97 | 96 | 95 | 94 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 79 | 79 | 77 | 76 | 75 | 74 | 73 | 72 |
| 1 | 99 | 98 | 97 | 96 | 95 | 94 | 93 | 91 | 90 | 89 | 88 | 87 | 86 | 85 | 84 | 82 | 81 | 80 | 79 | 78 | 77 | 76 | 75 | 73 | 72 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

Figure 6-9: Example 2, Case 2 Cell Values for Return Trip

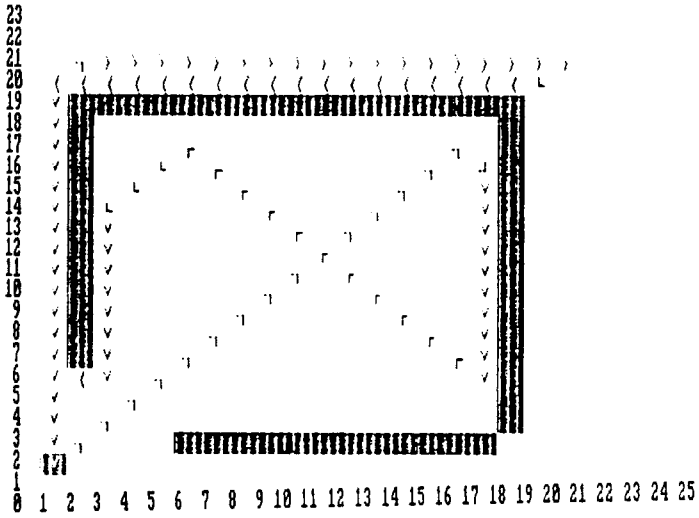



**Figure 6-10: Example 2, Case 2 The Complete Path**

### 6.3 Example Three

The curiosity factor will not always result in the robot travelling through known regions. This example shows how a path through unknown territory may be selected even when the curiosity factor is activated. The obstacle configuration used for the cases of this example is shown in Figure 6-11.

Again, incomplete information caused by gaps in the sonar sweep makes the robot appear indecisive, as shown in Figure 6-12 by the zig-zag near cell (13,8). After determining cells (18,6) and (18,8) contain obstacles, the robot begins exploring another region and finds the goal. The entire path, including the return trip, is shown in Figure 6-13.

The path is identical regardless of whether or not the curiosity factor is applied. This indicates that the potential savings in time or energy spent exploring the unknown path is much greater than the time needed to retrace the known path. That is, the potential benefit of finding an open path through the unexplored area outweighs the estimated risk (based on past experience) of exploring the unknown territory and finding it blocked.

### 6.4 Example Four

This example demonstrates the power of the algorithm by successfully navigating through a complex, maze-like

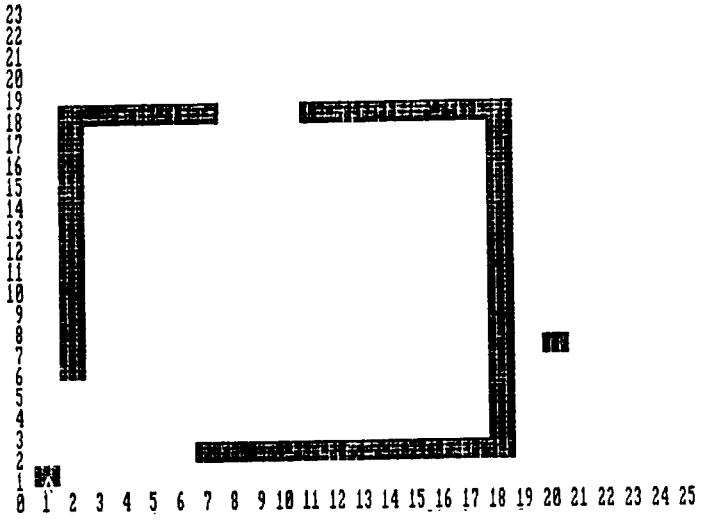


Figure 6-11: Example 3, Case 1,2,3,4 Obstacle Configuration

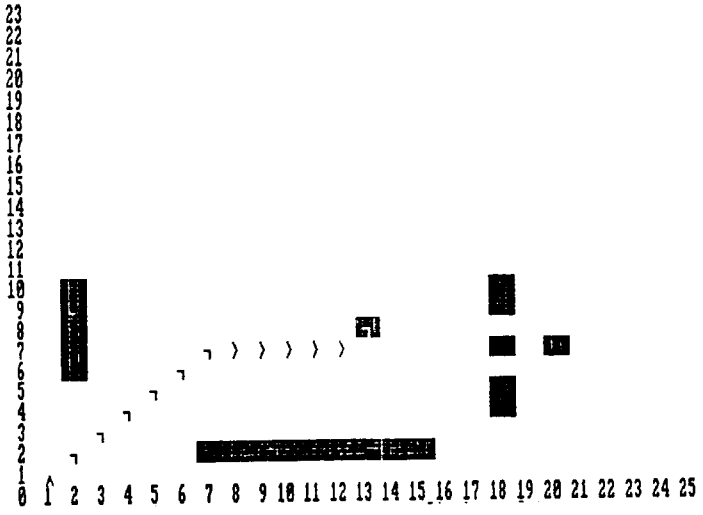


Figure 6-12: Example 3, Case 1,2,3,4 A Misleading Hole

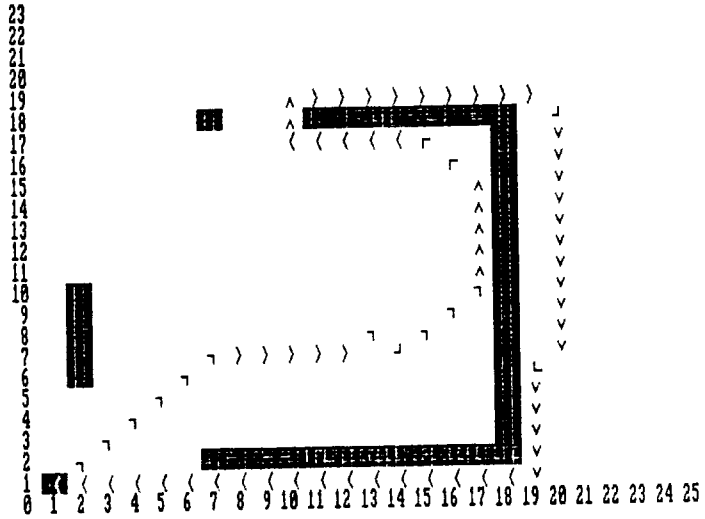


Figure 6-13: Example 3, Case 1,2,3,4 The Complete Path

obstacle configuration (Figure 6-14). Once again, Case 3, implementing the heuristic rule to reduce cell value calculations, produces the identical path generated by propagating values throughout the whole grid. Yet, a comparison of the cases in the analysis (Section 6.6) shows the heuristic reduces the total number of calculations performed by over eighty-five percent. The path to the initial goal is shown in Figure 6-15. The entire path found by all four cases is shown in Figure 6-16.

### *6.5 Example Five*

The obstacle configuration is shown in Figure 6-17. The purpose of this example is to demonstrate how dead end recovery works. Also, the previous four examples have similar or identical paths generated by each of the four cases. Each case in this example produces a unique path. The reasons for the differences will be examined.

In Case 1, the robot discovers a dead end on its first attempt to reach the goal. The first step of dead end recovery is shown in Figure 6-18.

The complete path is shown in Figure 6-19. Since the curiosity factor is not being used for this case, the robot attempts to cross the unexplored region, looking for a new route back to its original starting location. This new route is found to be blocked, so the robot finds its return path through previously explored territory. Since obstacle

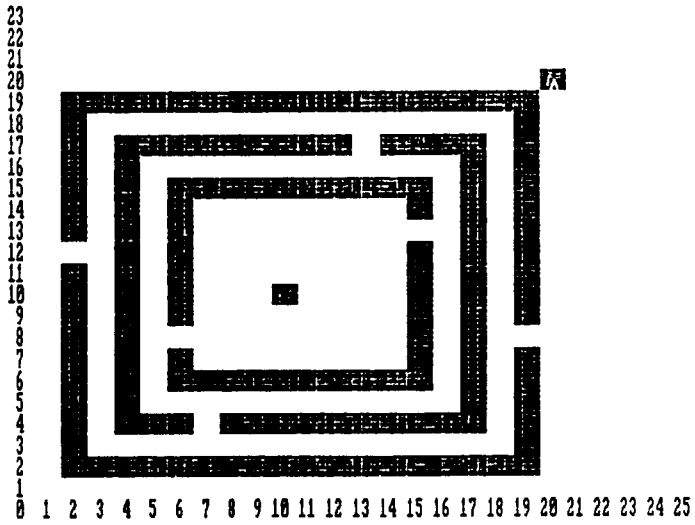


Figure 6-14: Example 4, Case 1,2,3,4 Obstacle Configuration

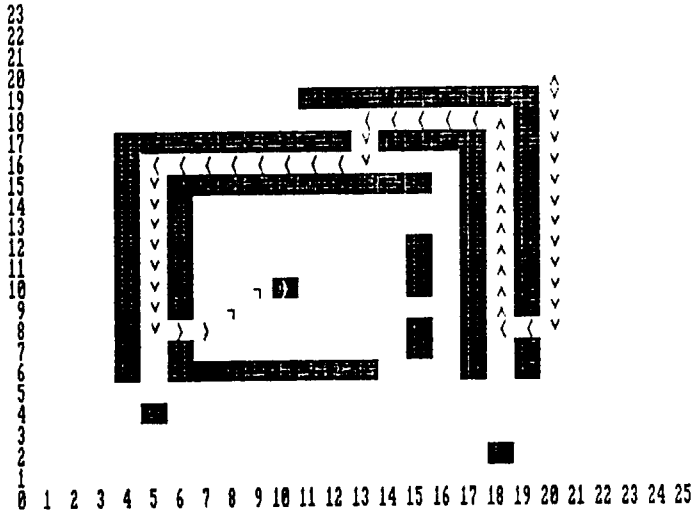


Figure 6-15: Example 4, Case 1,2,3,4 At the Initial Goal

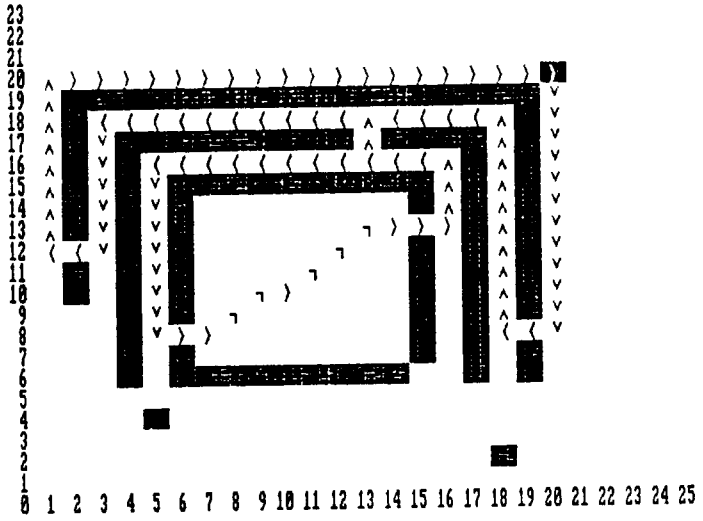


Figure 6-16: Example 4, Case 1,2,3,4 The Complete Path

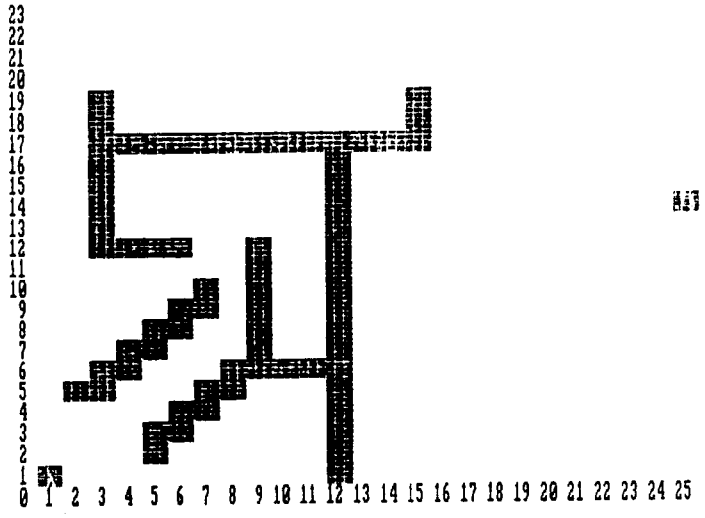


Figure 6-17: Example 5, Case 1,2,3,4 Obstacle Configuration

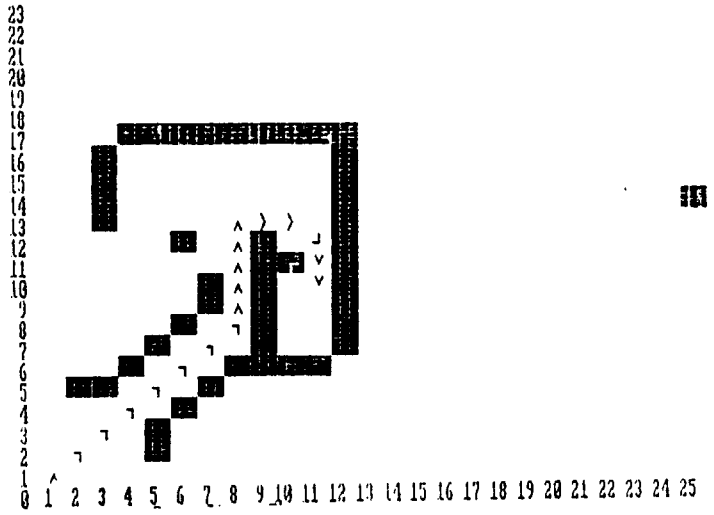


Figure 6-18: Example 5, Case 1,2,3,4 Dead End Recovery

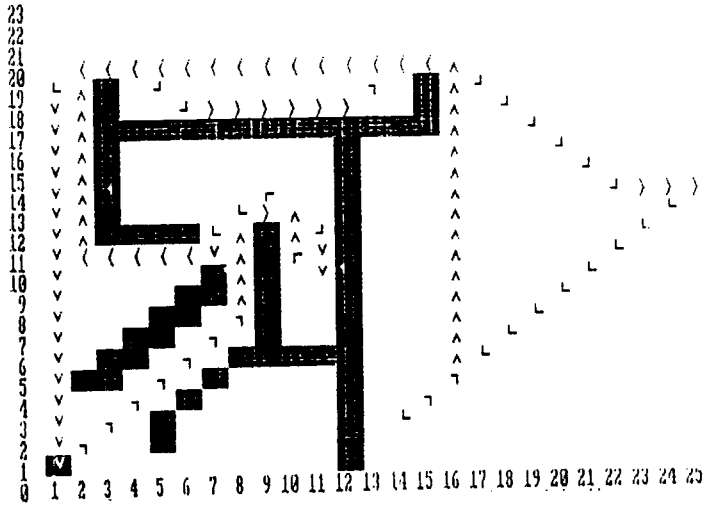


Figure 6-19: Example 5, Case 1 The Complete Path

information is maintained, the robot avoids having to explore the U-shaped configuration near the top of the grid.

Case 2 uses the curiosity factor. Again, the robot discovers a dead end and is forced to backtrack. This is shown in Figure 6-20. However, now the curiosity factor inhibits exploration and the robot retraces its path before entering new territory. This new area also leads to a dead end, as shown in Figure 6-21. The path leading to the goal is shown in Figure 6-22.

Once again, the robot attempts to return to its starting location. The exploration leading to the second dead end (Figure 6-21) gives the robot complete information of its surroundings. Hence, the return path is optimal and is shown in Figure 6-23.

Cases 3 and 4 follow the same route to the goal as shown for Case 1 (Figure 6-19). However, the return trips are different.

The return path followed for Case 3, using the cell value calculation saver, is shown in Figure 6-24. An explanation of this erratic path is also provided.

The cell values calculated after the robot has reached its initial goal, cell (25,14), but prior to beginning its return trip are shown in Figure 6-25. The values calculated from cell (16,5) are shown in Figure 6-26. The values shown in gray represent cells that have been scanned during the return trip. Only these values will change. All cell

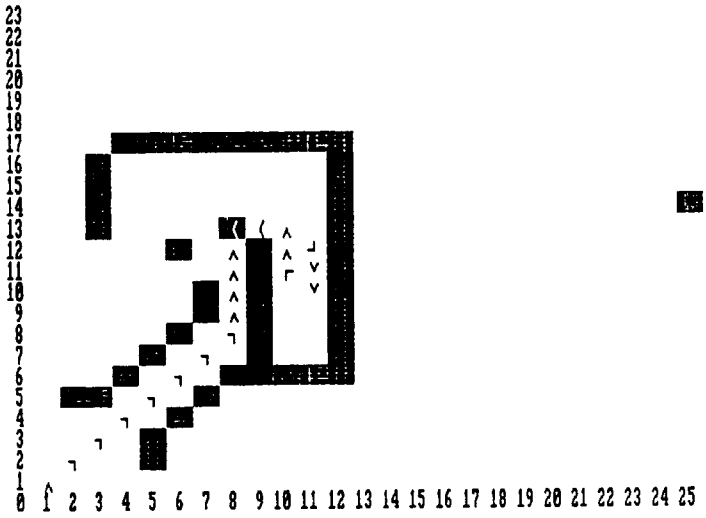


Figure 6-20: Example 5, Case 2 The First Dead End Avoided

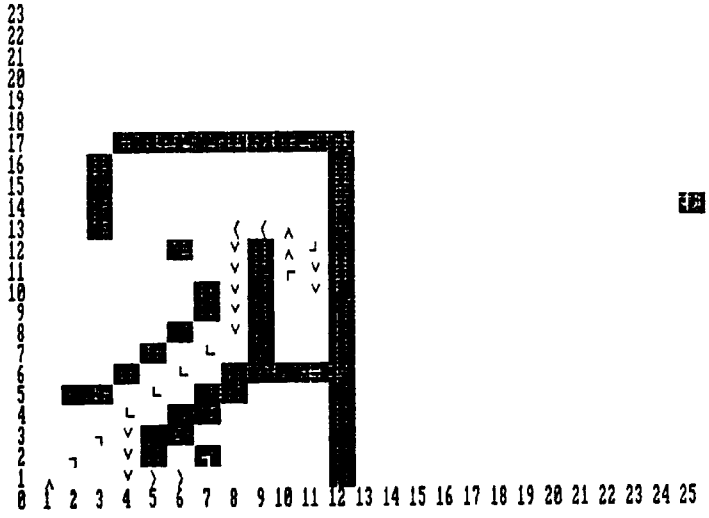


Figure 6-21: Example 5, Case 2 A Second Dead End Found

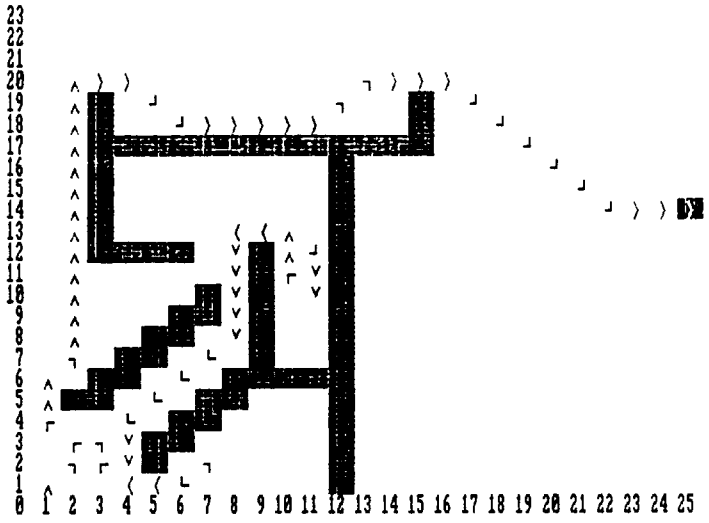


Figure 6-22: Example 5, Case 2 At the Initial Goal

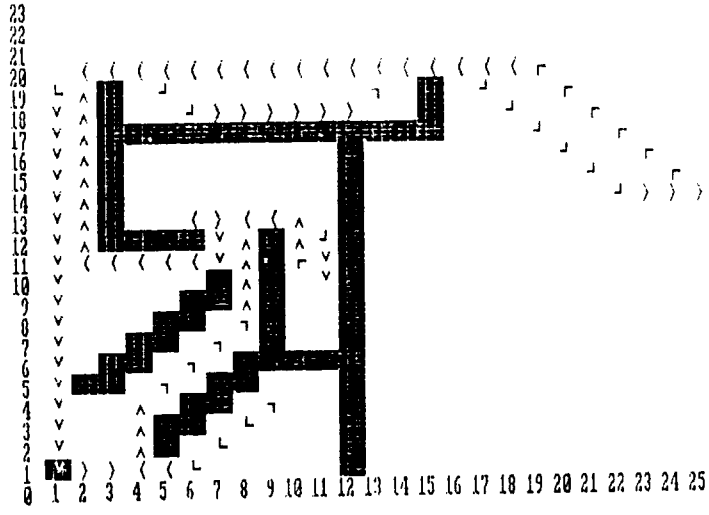


Figure 6-23: Example 5, Case 2 The Complete Path

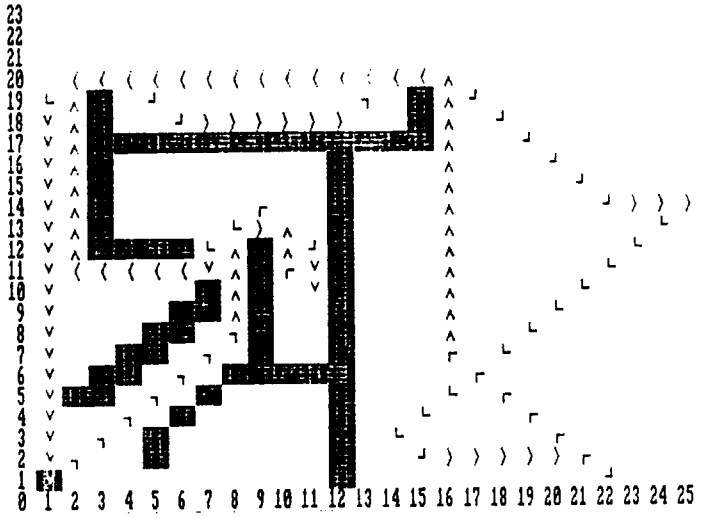


Figure 6-24: Example 5, Case 3 The Complete Path

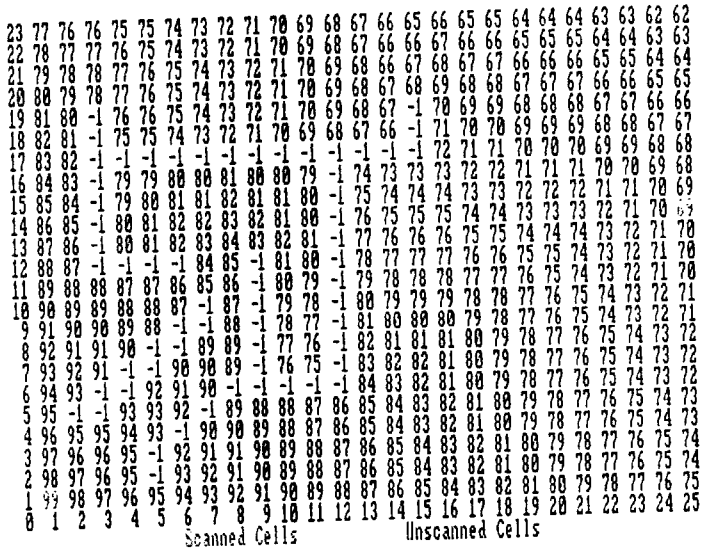



Figure 6-25: Example 5, Case 3 Cell Values Before Returning

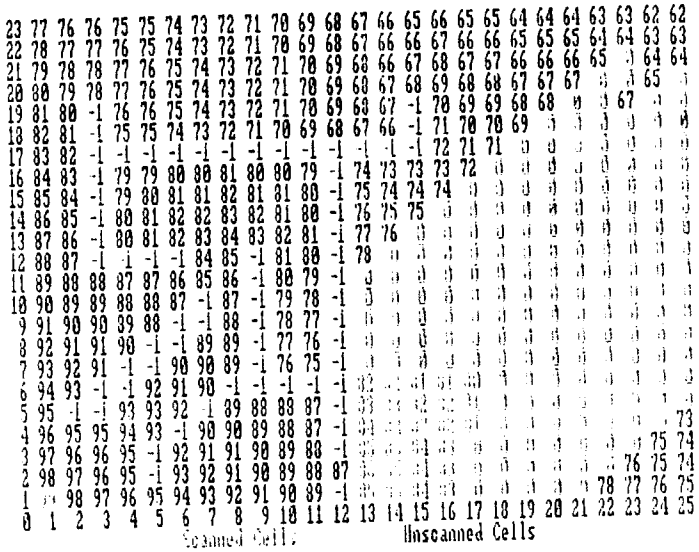



**Figure 6-26: Example 5, Case 3 Cell Values, Robot at (16,5)**

values shown in black will remain as they appeared in Figure 6-25.

From cell (14,3) the robot discovers an obstacle in cell (12,2) blocking its path. New values are propagated through cells the robot has seen since beginning its return trip (i.e., those cells shown in gray). Value calculations begin along the boundary between the seen and unseen territories. The small group of unseen cells in the lower right of the grid cause the robot to move directly away from the goal it is trying to reach. This pocket of unseen cells affect some of the cell value calculations, as shown in Figure 6-27.

The robot follows this false trail to cell (22,1) (Figure 6-24). At this point the robot is stuck. There are no neighboring cells with values greater than the cell the robot is in (i.e., no cell to move to). Since no new obstacles were detected, the propagation routine will not be invoked. For Cases 2, 3, and 4 a special invocation of the propagation routine, calculating all cell values, is performed before failure is announced. The values resulting from this special propagation are shown in Figure 6-28. The robot returns to its original starting location without further difficulties.

The path followed in Case 4 is shown in Figure 6-29. Unlike Case 2 (Figure 6-21), the obstacles in cells (12,1) through (12,5) are not discovered until the return trip.

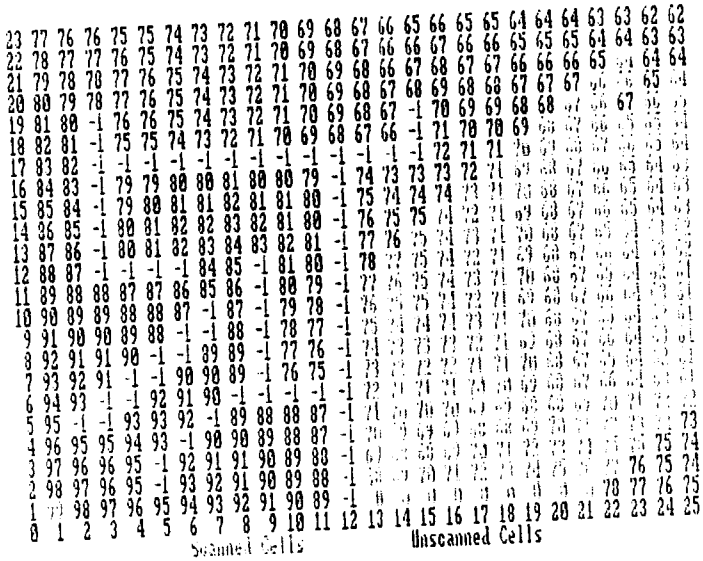



**Figure 6-27: Example 5, Case 3 Values Leading Robot Astray**

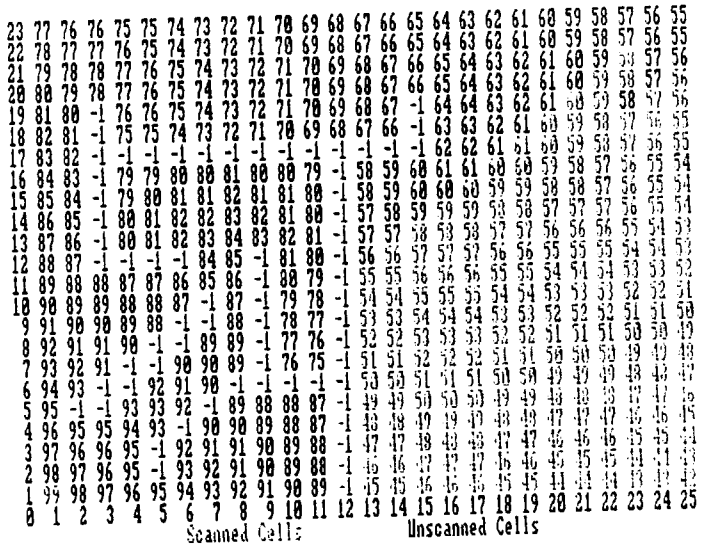



**Figure 6-28: Example 5, Case 3 Values After Robot is Stuck**

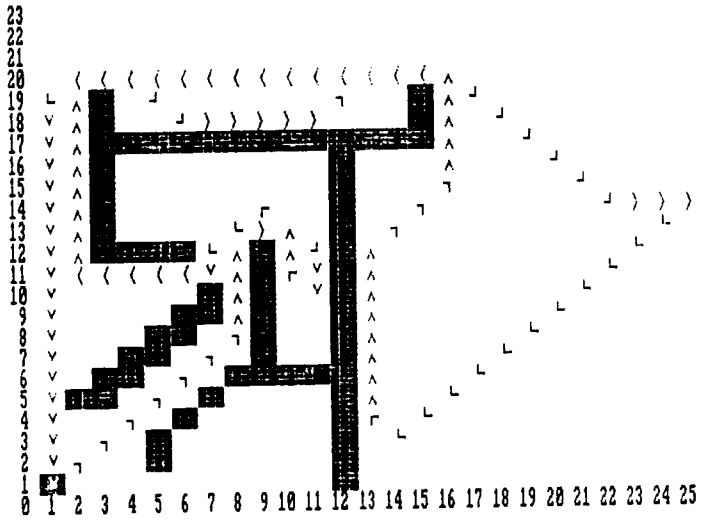


Figure 6-29: Example 5, Case 4 The Complete Path

### *6.6 Analysis of Example Cases*

Statistics are provided for each of the twenty cases in Table 6-1. The first two columns identify which example and case the remaining statistics are associated with. The "Moves" column indicates the number of cells the robot visited during its trip. The total number of times the value propagation routine is called is shown under the column labelled "Propagations." The total number of cell values calculated is given under the "Total Cells Calculated" column. The final column presents the percentage improvement compared to Case 1, in terms of the number of cell values calculated.

The statistics presented for Examples 1 and 3 demonstrate the power of the cell value calculation heuristic. A path identical to that found for Case 1 is generated in each example, but the number of calculations performed is dramatically reduced. Example 2 demonstrates the effectiveness of the curiosity factor in reducing the path length. When it is combined with the heuristic, additional computational savings are gained. Example 4 shows that in cases where the curiosity factor does not impact path length, combining it with the cell value calculation heuristic may require more calculations than using the heuristic alone. A similar situation occurs in Example 5. However, the shorter path length of Case 4 may be better than the reduced calculation load in Case 3.

Case 4 performance in Examples 4 and 5 is worse than that in Case 3. It is necessary to describe the method used for each of the cases to understand this performance decrease.

In the Case 3 simulations, once the robot reaches the initial goal, the *seen* flag of all cells that do not contain obstacles are reset to **FALSE**. This reduces the number of cell value calculations while still maintaining all obstacle information in the grid.

The curiosity factor used in Case 4 penalizes movement through unknown territory. Resetting *seen* flags to reduce the number of cell value calculations would also cause movement through the entire grid to be penalized. This is undesirable. Thus, Case 4 does not reset *seen* flags, thereby causing more cell value calculations than some Case 3 scenarios.

**Table 6-1: Analysis of Example Cases**

| Example | Case | Moves | Propagations | Total Cells Calculated | Improvement from Case 1 |
|---|---|---|---|---|---|
| 1 | 1 | 157 | 53 | 17934 | -- |
| 1 | 2 | 157 | 53 | 17934 | -- |
| 1 | 3 | 157 | 53 | 2362 | 87% |
| 1 | 4 | 157 | 53 | 2362 | 87% |
| 2 | 1 | 164 | 28 | 10184 | -- |
| 2 | 2 | 120 | 26 | 9363 | 8% |
| 2 | 3 | 164 | 28 | 3071 | 70% |
| 2 | 4 | 120 | 26 | 2859 | 72% |
| 3 | 1 | 75 | 25 | 8195 | -- |
| 3 | 2 | 75 | 25 | 8195 | -- |
| 3 | 3 | 75 | 25 | 2552 | 69% |
| 3 | 4 | 75 | 25 | 2552 | 69% |
| 4 | 1 | 111 | 70 | 24121 | -- |
| 4 | 2 | 111 | 70 | 24121 | -- |
| 4 | 3 | 111 | 70 | 3411 | 86% |
| 4 | 4 | 111 | 70 | 6265 | 74% |
| 5 | 1 | 122 | 38 | 16182 | -- |
| 5 | 2 | 124 | 35 | 14806 | 9% |
| 5 | 3 | 132 | 40 | 3605 | 76% |
| 5 | 4 | 122 | 40 | 4266 | 71% |

# CHAPTER 7

# IMPLEMENTATION ON THE HERO 2000

The simulation program demonstrates the power and flexibility of the CREEP algorithm. The same program is used in conjunction with a Heathkit Zenith HERO 2000 robot to implement the algorithm in a real-world scenario.

While it would be possible to implement the CREEP algorithm using the BASIC language, several considerations led to the remote control implementation in conjunction with the simulation program, instead. First and foremost, the memory capacity of the robot is too small to store the entire program, let alone the grid representation of its surroundings. Second, the simulation program shows the robot, its path, the goal, and obstacles in an easily understood graphic display. Grid cell values can be displayed at any point during the simulation, thereby showing the effects of new obstacles on value propagation. These are the main reasons for connecting the HERO and the simulation program via a radio remote link.

This chapter describes the characteristics, commonly used instructions, remote operation, sonar system, and mobility of the HERO 2000. Following this, the software stored in the robot's memory is discussed.

### 7.1 The HERO 2000

The HERO 2000 is a versatile, programmable, mobile robot. Its features include three modes of control (direct, remote, and link), a rotating torso, a manipulator with five degrees of freedom, a speech synthesizer, and a four component sensory system (two sonar, one light, and one temperature sensor). Of these features, the ones used for the implementation of the CREEP algorithm are the sonar system, the robots ability to move, and the link mode of control (described in Section 7.1.3).

#### 7.1.1 The HERO BASIC Instruction Set

The HERO 2000 can execute programs written in BASIC. Several special commands have been added to the typical BASIC instruction set. These commands allow easy control of several of the robot's systems, such as the motors, sonar, speech synthesis, and others.

#### 7.1.2 Remote Control Mode

A communications link is established between the HERO 2000 and the PC executing the simulation. The HERO is equipped with a radio transmitter/receiver. With it, data is transmitted between the robot and a remote console, designed specifically for this purpose. The console is connected to the PC through the RS-232 serial port. All commands issued from the simulation program are relayed from

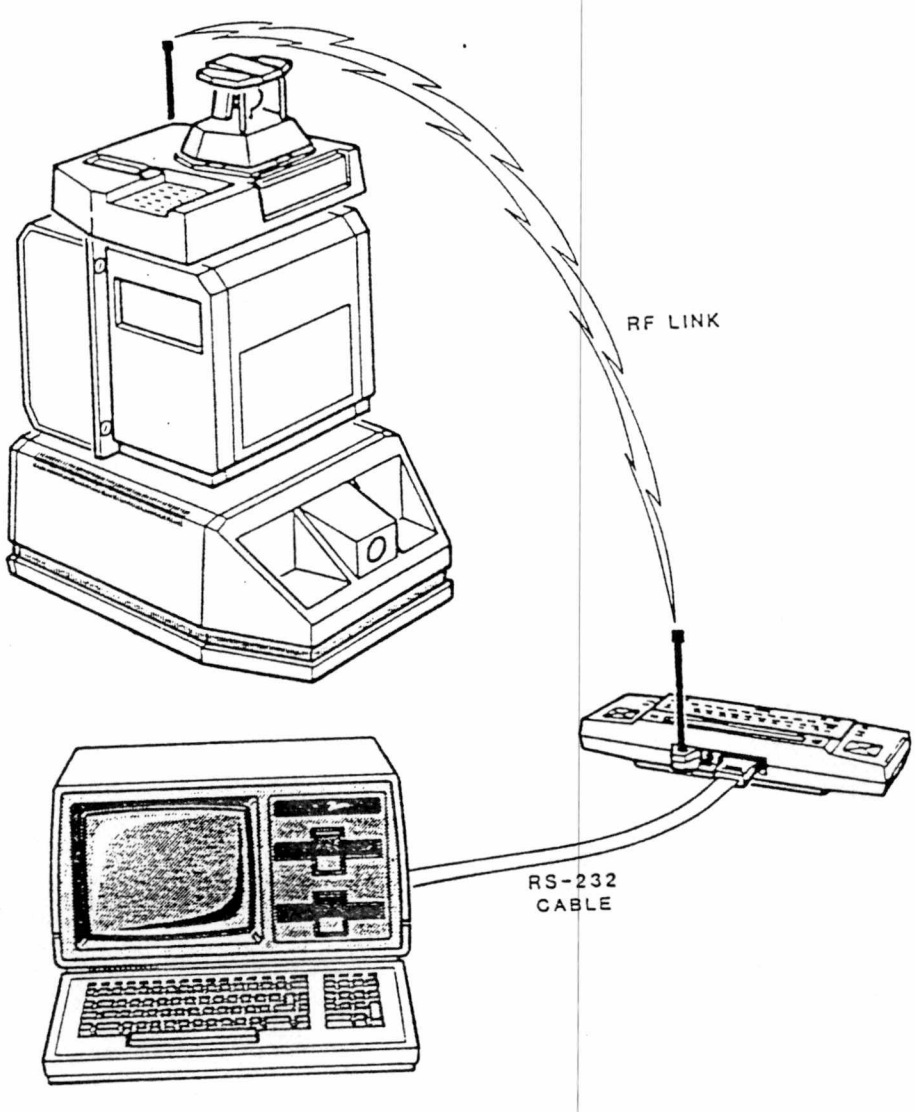



**Figure 7-1: The HERO 2000 in Link Mode**

the PC through the remote console to the robot. All data requested by the PC is sent from the robot, back to the console, which, in turn, sends it to the PC for interpretation. This communication link is illustrated in Figure 7-1.

### *7.1.3 Sonar Capabilities*

The HERO 2000 is equipped with two sonar sensors. One is located on the base of the robot and is capable of taking readings only in the direction the robot is facing. The second sensor is located on top of the robot's torso. Using a rotating reflector, the robot is capable of performing sonar assessments in a 360° sweep. The reflector turns in 15° increments, allowing a total of twenty-four readings to be gathered.

### *7.1.4 Mobility*

The HERO's movements are controlled by two motors, each driving a wheel. These motors can be controlled as a single unit (using the BASE command) or individually (with the LFT and RHT commands). Two parameters are associated with each of these instructions; one specifies the distance to travel, the other, the speed of movement.

### *7.2 The Control Program*

A BASIC program is stored in the HERO's memory. It waits for receives instructions from the PC simulation program, acts on those instruction, and, if necessary, transmits data back to the PC.

The robot remains idle until a single digit command is received by from the PC. The expected value is a number specifying the action the robot is to perform. A '1' or a '3' requests a light scan or a base sonar reading, respectively. Neither of these functions is currently used in the CREEP algorithm.

If a '2' is received, the robot performs a circular sonar sweep. Twenty-four sonar measurements are made, each separated from the previous scan by 15°. As the values are acquired, they are stored in memory. When the entire scan is complete, the twenty-four values are transmitted to the PC.

A number between '4' and '7' initiates movement. A '4' indicates a left turn, '5' a right turn, '6' forward motion, and '7' movement in reverse. Before any action is taken, the robot receives a second number from the PC which specifies the distance the robot is to move.

The program also uses the speech synthesizer to broadcast what action robot is trying to perform. This gives a definite indication as to what portion of the program is executing.

### *7.3 Implementation Analysis*

The hardware of the HERO 2000 is not state of the art. The communication link between the robot and the PC slows execution time. The delays imposed by the robot hardware and communication mode are examined in this section.

The implementation of the CREEP algorithm uses the simulation program for value propagation and path planning. While cell processing still affects the time requirements of the computer program, the robot hardware adds new factors to the analysis.

The sonar scan consists of twenty-four individual readings. A measurement is made and stored until all twenty-four values are transmitted to the PC. The sonar sweep requires approximately 5 seconds, the transmission 12 more.

Once a move has been determined by the simulation program, it must be communicated to the robot. The robot must turn, if necessary, and move to the next cell. Instructing the robot to turn requires 3 seconds; executing the turn takes between 2 and 5 seconds depending on distance the robot must turn (45°, 90°, or 135°). Instructing and executing a move requires 4 seconds when moving parallel to the axes or 5 seconds moving diagonally.

In the implementation of the CREEP algorithm the scan, plan, move cycle is dominated by the time required to perform the sonar sweep and transmit the data. Movement is

also considerably slower than in the simulation. Future research can explore ways to speed these phases of the implementation.

# CHAPTER 8

# CONCLUSIONS AND SUMMARY

## *8.1 Conclusions*

The power and flexibility of the CREEP approach has been demonstrated. The search technique assures that if a path exists, it will be found. Also, the best path, based on current information, will be explored first. The planned path avoids collisions with obstacles. Recovery from dead ends is easily accomplished; invoking the propagation routine allows the robot to leave the blocked region and begin exploration of a new area.

It is often desirable to inhibit exploration when planning a path back to the original starting location. Incorporating a curiosity factor into cell value calculations can improve return trip performance.

Restricting cell value calculations only to cells that have been scanned by the robot greatly reduces the calculation time. In most cases, the time saved does not come at the expense of a longer path.

The feasibility of implementing the CREEP algorithm has been demonstrated using a HERO 2000 robot. The mobility and sonar sensor of the robot meet the requirements needed to implement the CREEP approach. The time required to conduct

the sonar sweep is the predominant factor in the implementation. Future research should investigate methods of reducing these time requirements.

### *8.2 Summary*

Autonomous mobile robots can aid people in wide variety of tasks. In scenarios where the environment is hazardous or lethal a robot can replace human workers, thus removing them from life threatening situations.

This thesis presents an algorithm that allows a mobile robot, equipped with a sensing device, to navigate through unfamiliar terrain. The algorithm consists of three phases: scan, plan, and move.

The scanning phase acquires data about the area surrounding the robot. A circular sonar sweep is performed and twenty-four readings indicate the distance between the robot and any detected obstacles. A two dimensional grid is used to store this data and model terrain the robot has investigated.

This grid is divided into equal-sized cells. Cell size selection criteria include, but are not limited to, the distance between the robot and goal, computing power of the robot, size of the robot, and the range and accuracy of the robot sensor system.

The planning phase assigns values to cells based on their distance from the goal. The order in which cell

values are calculated mimics a wave rippling out from a pebble dropped into still water. The propagation wave emanates from the goal in all directions and decreases in strength the further it travels. A value equivalent to the wave strength is assigned to each cell the wave crosses. The propagation stops either when all cells adjacent to the robot have been assigned values or when no more cells can be reached by the wave.

The move phase consists of finding the maximal neighbor of the cell in which the robot resides. If this cell's value is greater than zero, the robot moves to it. Otherwise, the propagation wave could not reach the robot. That is, no path exists between the robot and goal, so the algorithm terminates.

If the robot successfully reaches the goal, it attempts to return to its starting location. All obstacle information is maintained.

A curiosity factor can be incorporated in the cell value calculation formula. It penalizes paths that cross unexplored regions. The penalty is the ratio of the number of cells containing obstacles to the total number of cells scanned.

A heuristic is introduced to reduce the time required for propagation. Values are calculated only for cells the robot has scanned.

The CREEP approach is guaranteed to find a path if one

exists. The algorithm operates on order $O(n^2)$, where n represents the number of cells in the grid. The assignment of cell values in the propagation routine is the main contributor to the time complexity of the algorithm.

A Heathkit Zenith HERO 2000 robot is used to demonstrate the feasibility of implementing the algorithm. The rotating sonar is used to collect environmental data. The simulation program incorporates this real data into the grid. Using the new data a move is planned and communicated to the robot. The robot executes the move and begin another scan, plan, move cycle with a new sonar sweep.

# BIBLIOGRAPHY

## BIBLIOGRAPHY

Ali, A., Abidi, M., "A 2-D and 3-D Path Planning Algorithm for Robot Navigation and Manipulation," Technical Report, Dept. of Electrical and Computer Engineering, University of Tennessee, Knoxville, 1987

Arkin, C., Murphy, R., "Autonomous Navigation in a Manufacturing Environment," IEEE Transactions on Robotics and Automation, Vol. 6, No. 4, August 1990, pp 445-454

Balaguer, C., Marti, A., "Collision-free Path Planning Algorithm for Mobile Robot Which Moves Among Unknown Environment," IFAC Proceeding Series 1989, Number 10, pp 261-266

Chatergy, R., "Some Heuristics for the Navigation of a Robot," The International Journal of Robotics Research, Vol. 4, No. 1,Spring, 1985, pp 59-65

Chavez, R., Meystel, A., "Structure of Intelligence for an Autonomous Vehicle," Proceedings of the International Conference on Robotics, IEEE, 1984, pp 584-591

Daily, M., Harris, J., Keirsey, D., Olin, K., Payton.

D.,Reiser, K., Rosenblatt, J., Tseng, D., Wong, V., "Autonomous Cross-Country Navigation with the Autonomous Land Vehicle," 1988 IEEE International Conference on Robotics and Automation, Vol. 2, pp 718-726

DeSaussure, G., Kammer, D.W., Weisbin, C.R., "A Real-time Expert System for the control of Autonomous Robot Navigation in the Presence of Moving Obstacles," Remote Systems and Robotics in Hostile Environments International Topical Meeting, April, 1987

Elfes, A., "Using Occupancy Grids for Mobile Robot Perception and Navigation," IEEE Computer, June 1989, pp 46-57

Gaffney, J.E., "Navigation in Space, the Future, and Artificial Intelligence," ION National Space Meeting, March 1972, pp 79-85

Graglia, P., Meystel, A., "Planning Minimum Time Trajectories in the Traversability Space of a Robot," Proceedings of the 1987 IEEE Conference on Robotics and Automation, IEEE, 1987

Hinkel, R., Kneiriemenk, T., "Spatial Uncertainty Management for a Mobile Robot and its Role in Expecta-

tion-Based Perception," IFAC Proceeding Series 1989, Number 10,pp 279-284

Hubbard, J.D., Larson, L.P., "HERO 2000: Programming and Interfacing," Heath Company, 1986

Ilari, J., Torras, C., "2D Path Planning: A Configuration Space Heuristic Approach," The International Journal of Robotics Research, Vol. 9, No. 1, February, 1990, pp 75-91

Lee, C.Y., "An Algorithm for Path Connections and Its Applications," IRE-Transactions on Electronic Computers, Vol. EC-10, No. 3, Sept. 1961, pp 346-365

Matthies, L., Elfes, A., "Integration of Sonar and Stereo Range Data Using a Grid-Based Representation," IEEE, Jan. 1988, pp 727-733

Mehrotra, R.,Krause, D.M., "Obstacle-free Path Planning for Mobile Robots," Third International Conference on Image Processing and Its Applications, July 1989, pp 431-435

Mitchell, J.S.B., Payton, D.W., Keirsey, D.M., "Planning and reasoning for Autonomous Vehicle Control,"

International Journal of Intelligent Systems, Vol. 11, pp 129-198

Mueller, H., Neubauer, H., Mlynski, D.A., "Path Assignment to a Set of Autonomous Guided Vehicles," IFAC Proceeding Series 1989, Number 10, pp 267-270

Nilson, N.J., "Principles of Artificial Intelligence," Tioga Publishing Co., Palo Alto, CA p 76

Parodi, A., "Multi-Goal Real-Time Global Path Planning for an Autonomous Land Vehicle Using a High-Speed Graph Search Processor," IEEE International Conference on Robotics and Automation, 1985, pp 161-167

Shiller, Z., Gwo, Y. "Dynamic Motion Planning of Autonomous Vehicles," IEEE Transactions on Robotics and Automation, Vol. 7, No. 2, April 1991, pp 241-249

Witkowski, C.M., "A Parallel Processor Algorithm for Robot Rout Planning," Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 1983, pp 827-829

Wong, E., Fu, K., "A Hierarchical Orthogonal Space Approach to Three-Dimensional Path Planning," IEEE

Journal of Robotics and Automation, Vol. RA-2, No. 1,
March 1986, pp 42-53

## VITA

Craig William Peterson was born in Windom, Minnesota on January 4, 1964. He graduated from Windom Area High School in 1982. In 1986 he received a BA degree from Gustavus Adolphus College, majoring in Mathematics and Computer Science.

After working for three years as a programmer/analyst in Minneapolis, he returned to academia. In August, 1989 he enrolled at the University of Tennessee Space Institute. In August, 1991 he was awarded a Master of Science degree in Computer Science from the University of Tennessee, Knoxville. He is currently an environmentally conscious, athletically active, and politically aware citizen.