A Unified Solution to Coverage and Search in Explored and Unexplored Terrains Using Indirect Control

Amir Pirzadeh Wesley Snyder

Center for Communications and Signal Processing
North Carolina State University
Raleigh, NC 27695-7911

Abstract

An algorithm is described which solves coverage and search problems in either explored or unexplored terrains. The coverage problem requires that the robot pass over all parts of the terrain that is free of obstacles. The search problem requires that the robot seek a specified target in a given terrain. The target need not be present in the terrain; in that case, the algorithm will realize the fact after a thorough search. In an explored terrain problem, a model of the terrain is available indicating which regions of the terrain are traversable. In an unexplored terrain problem, such a model is not available, thus forcing the robot to build the model by the use of sensors. It is proven that the algorithm guarantees complete coverage and a thorough search if physically possible. Furthermore, due to its use of indirect control, the algorithm is not complicated, and thus, simple to implement.

1 Introduction

This paper presents a new approach to a general class of problems in autonomous robot navigation. The solution is based on the philosophy of indirect control. By indirect control we mean that the algorithm presented here does not determine a destination for the robot and then calculate a complete path for the robot to take. Instead, the algorithm controls the robot path loosely by placing penalty costs along paths the robot may take, thus directing the robot movement indirectly.

Four different problems are solved here using the indirect control strategy. These are coverage and search problems in either explored or unexplored terrains. The coverage problem requires that the robot pass over all parts of the terrain that is free of obstacles. Possible applications of this problem are automatic vacuum cleaners, automatic lawn mowers, land mine detectors, etc. The search problem requires that the robot seek a specified target in a given terrain.

For either the problem of coverage or of search, we may be faced with one of two possible situations: we may already have a model of the terrain we are covering or searching, or we may have no prior knowledge of the exact locations of obstacles in the terrain. These are referred to here as the explored and unexplored terrain problems respectively.

Conventional path planning methods are based on the assumption that the origin-destination pair is known. Thus, the problem is reduced to that of finding a collision free path from the robot initial position to the destination. As a result, these methods typically involve searching a graph for the optimal or near optimal path [8] [7] [1] [5]. Other methods for solving such problems include the use of Voronoi diagrams [9], algebraic cost functions [6], probability theory [4], and heuristics [2]. Yap [IO] has a good survey on the evolution of path planning methods. The problems solved in this paper, however, do not have a clear destination. The coverage problem has the entire set of non-obstructed space as its destination while the search problem does not have a known destination (if any.)

Following is a list of assumptions made by the algorithm.

- The terrain boundary is assumed to be known to the robot in terms of a relative distance to its initial position. (A rectangular terrain is implied here with no loss of generality.)
- The robot is assumed to have some means for recognizing obstacles and calculating the distance to them. (In the explored terrain problem, this is only required as a safety measure.)
- The robot is assumed to be able to recognize the target when in a direct line of sight. (This only applies to the search problem.)

The underlying strategy of the algorithm is to divide the terrain into cells. We shall refer to a cell in this paper as a unit of area equal to the dimensions of the robot base. In the explored terrain problem, the algorithm knows a priori whether each cell is obstructed or free of obstacles. In the unexplored terrain problem, this information is gathered as the robot navigates the terrain. Nevertheless, by dividing the terrain into cells, we are able to navigate around obstacles of any shape since the problem is now discrete.

Given the discrete terrain model presented above, we also discretize the robot motion. More specifically, we limit the robot movement to orthogonal directions designated as up, down, left, and right. Since we have limited the robot motion (and therefore sight) to the specified four directions, we will be dealing with a four connected problem, that is, a cell is not considered to be connected to its diagonal neighbors.

The following terminology will be used in this paper.

- cell : A unit of area equal to the dimensions of the robot base.
- cell cost : The cost of covering a particular cell.
- direction cost : The cost of traveling in a particular direction.

CH2876-1/90/0000/2113$01.00 @

- free cell : A cell that is completely free of obstacles and, thus, traversable by the robot.
- obstructed cell : A cell that is obstructed and, thus, not traversable by the robot.

The algorithm will be presented in three parts in order to present a lower level understanding of the main components of the algorithm. These three parts are composed of the main component referred to as the base algorithm, and two additional components called “looking ahead in space” and “looking ahead in time”. Furthermore, three sets of results will be provided corresponding to the performance of the base algorithm and the addition of the other two components so as to show the impact of each component to the overall performance.

We will present the algorithm in detail as it would be applied to the coverage of the explored terrain problem. Section will give the minor modifications necessary for solving search problems and section will give the modifications necessary for unexplored terrains.

2 The Base Algorithm

This component of the algorithm guarantees complete coverage in a coverage problem and a thorough search in a search problem. The necessary conditions for this statement to hold and the corresponding proof will be provided following an explanation of the base algorithm.

The base algorithm:

1. Increment the current cell cost by α.
2. Interrogate neighboring cells to determine the least costly direction in which to travel. Thus, the total cost of traveling in the direction dir is (provided there are no obstacles:)

$$direction\_cost(dir) = cell\_cost(dir).$$

3. Choose the direction according to the priority: up, down, right, left, in case of tie.

In the algorithm, step one tends to make the robot traverse the terrain in such a manner as to minimize the number of times it covers the same free cell; thus, it can be thought of as a cost function. It also insures that the robot will not get caught in an endless loop by penalizing it for going over cells it has covered before. Cell costs should be initialized to zero prior to this step.

Step two calculates the control that will minimize the cost function of step one. This control is indirect because it is not based on the knowledge of the ultimate goal of the problem, but is decentralized in that the goal is to try not to retrace previous steps as much as possible.

The base algorithm is similar in spirit to the algorithm used by Shannon (see [12] for a more detailed explanation of Shannon’s work.) Shannon constructed an electronic mouse that searches a maze for the “cheese”. Shannon’s maze consists of squares each of which holds two bits representing four directions. The mouse is able to search the entire maze by leaving squares 90” to the left of the previously departed direction.

The following Theorem provides the necessary conditions for the base algorithm to guarantee complete coverage[1] and a thorough search[2].

Theorem 1 ***The base algorithm*** guarantees ***complete coverage and a thorough search in the region R if the set of free cells $\mathbf{x} \in R$ have the following property: for all x_i and $x_j \in \mathbf{x}$ there exists a sequence $\{x_i, \dots, x_j\}$ such that x_k and x_{k+1}*** are ***adjacent (in a four connected sense) and $i \leq k < j$.***

We prove the above statement by contradiction. If the algorithm is not to guide the robot along a path of complete coverage, then it must get caught along a circular path. Let

$$\mathbf{y} \subset \mathbf{x}$$

where y is the set of free cells that lies on the circular path. Furthermore, let

$$p = x - y$$

Then, after each pass along the circular path,

$$cell\ cost(y) = cell\text{-}cost(y) + a$$

while ***cell_cost{p}*** remains unchanged. Consequently, based on the condition in Theorem 1, and at some finite time t, there will be a cell $p_i \in$ p adjacent to some cell $y_j \in$ y such that

$$cell\_cost(pi) < cell\_cost(y_j)$$

Thus, step two will deviate the robot from the circular path and cover the cell p_i.

Hence, we have shown that no circular path is possible for an infinite duration of time resulting in complete coverage in finite time. Since in the worst case a search problem approaches a coverage problem, we can also guarantee a thorough search based on the above argument.

We have thus shown that the base algorithm alone is sufficient to guarantee complete coverage and a thorough search. Unfortunately, however, it does not give us an optimal solution. As a matter of fact, in practice, its performance is rather poor. The purpose of each of the remaining two components is to give a more efficient solution.

3 Looking Ahead in Space

This component of the algorithm utilizes information available from cells along the line of sight of the robot. One such datum is whether all the cells in a given direction have been covered. If all the cells in a given direction have been covered, then we may penalize the robot for choosing that direction. This cost function is implemented in Step 2 of the following algorithm.

Another type of information that is available is the distance to an obstacle or boundary in a given direction. This information may be used to break possible up-down or right-left ties. We break such ties by choosing the direction that will bring the robot to an obstacle or boundary more quickly. This causes the robot to finis:, covering smaller regions before diverting to cover larger regions. This operation is performed in step 4 of the following algorithm.

The base algorithm with look ahead in space:

1. Increment the current cell cost by α.
2. Assign an additional cost of β to a direction if all the cells in that direction from the current position to an obstacle or boundary have already been covered.

[1] A region $R \subset \Re^2$ **is said to be completely covered if all the free cells $x \in R$ are traversed at least once by the robot.**

[2] **A region $R \subset \Re^2$ is said to be thoroughly searched if all the free cells $x \in R$ have been interrogated at least once by the robot sensor.**

3. Interrogate neighboring cells to determine the least costly direction in which to travel. Thus, the total cost of traveling in the direction dir is (provided there are no obstacles:)

$$\text{direction } \mathbf{cost}(dir) = \mathbf{cell\ cost}(dir) + \beta \text{ cost}(\&).$$

4. Choose the direction that will bring the robot to a boundary or obstacle more quickly in case of an up-down or right-left tie. In case of any other tie, choose the direction according *to* the following priority: up, down, right, left.

The addition of this component of the algorithm to the base algorithm improves the overall performance significantly (see section 5 However, further improvement in performance is possible by identifying cells that do not need to be covered anymore. The next component of the algorithm recognizes such cells.

4 Looking ahead in time

The purpose of this component of the algorithmis to associate a large cost with covering cells that need not be covered again. By assigning such a large cost to a cell, we will practically be treating that cell as an obstructed cell. Thus, apart from requiring that these cells be covered, we also require that the condition stated in Theorem 1 not be violated. This condition will be violated if the cells penalized are the only ones that provide a traversable path between two other free cells.

Figure 1 shows five possible situations that might occur. Assuming all the cells in column "a" have been covered, only in the first four cases can we associate a large cost with entering any of the cells in column "a" again. In the last case, it might be required to traverse some of the cells in column "a" in order to get to the other part of the terrain. Notice that the difference between the last example and the first four is in the occurrence of the sequence: (free-cell obstructed cell free-cell] adjacent to column "a".

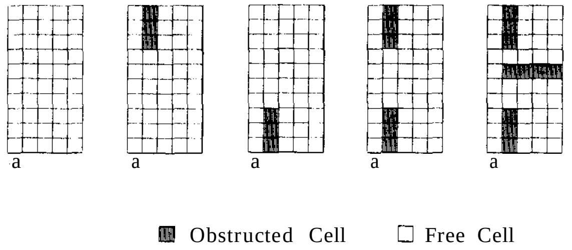

Figure 1: Possible obstacle arrangements

Besides columns or even rows, this idea can be extended as well to *mini-rows* and *mini-columns.* The terms mini-row and mini-column are used to describe free cells that do not necessarily lie between the boundaries, but that lie between a boundary or obstructed cell on one side and a boundary or obstructed cell on the other. Figure 2 shows examples of these.

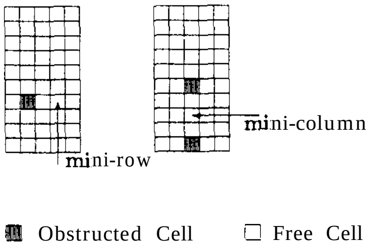

Figure 2: Examples of mini-rows and mini-columns

We shall now provide the conditions necessary for associating a large cost with covering certain cells. We shall refer to cells penalized in this manner as finished cells. Let the set B = *{boundary, obstructed-cell, finished_cell},* and let the set F = {free-cell}. Furthermore, let the operator + on the set **R** (designated $\mathbf{R}^+$) describe the new set consisting of elements formed by concatenating a finite non-zero number of elements from {R}. Parsing of the regular expression $\mathbf{R}^+$ is accomplished by passing elements of the set **R** through a finite state machine. For example, [free-cell obstructed-cell free-cell free_cell] is one element of F+B+F+. Two definitions are now provided for recognizing finished cells. Cells that meet the following criteria should be assigned a cost of γ.

Definition 1: Cells within the mini-row (or mini-column) Γ may be designated finished if all the following criteria are met.

1. All the cells within Γ are covered.
2. The adjacent cells on one side of Γ form an element of $\mathbf{B}^+$.
3. The adjacent cells on the other side of Γ ***do not*** form an element of $\mathbf{F}^+\mathbf{B}^+\mathbf{F}^+$.
4. The robot is ***not*** currently in Γ.

Dednition 2: Cells in between the two rows (or columns) Γ_1 and Γ_2 may be designated finished if all the following criteria are met.

1. All the cells in between Γ_1 and Γ_2 are covered.
2. The cells of Γ_1 form an element of $\mathbf{B}^+$.
3. The cells of Γ_2 ***do not*** form an element of $\mathbf{F}^+\mathbf{B}^+\mathbf{F}^+$.
4. The robot is ***not*** currently in between Γ_1 and Γ_2.

The cells marked "finished cell candidate" **in Figure 3 satisfy the requirements of Definition 1 provided that requirements 1 and 4 of that definition are also met. Similarly, the cells marked "finished cell candidate" in Figure 4 satisfy the requirements of Definition 2 provided that requirements 1 and 4 of that definition are also met.**

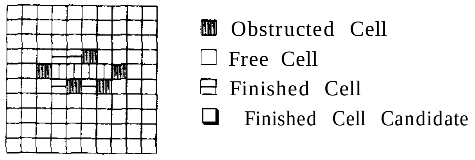

Figure 3: Example of recognizing finished cells using Definition 1

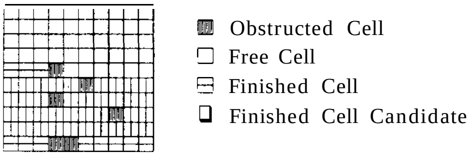

Figure 4: Example of recognizing finished cells using Definition 2

Given the above definitions, we may now present the complete algorithm for the robot's decisions at step k. We designate the mini-row and mini-column that the robot currently occupies as Γ^k_{mr} and Γ^k_{mc} respectively. Similarly, we designate the mini-row and mini-column that the robot previously occupied as Γ^{k-1}_{mr} and Γ^{k-1}_{mc} respectively. The base algorithm with look ahead in space and look ahead in time:

1. Increment the current cell cost by α.

2. Assign an additional cost of β to a direction if all the cells in that direction from the current position to an obstacle or boundary have already been covered.

3. Use Definition 1 to see whether the cells of Γ^{k-1}_{mr} or Γ^{k-1}_{mc} may be declared finished. If so, then associate a cost of γ to these cells. (Skip this step in the first iteration.)

4. Use Definition 1 to see whether the cells of Γ^k_{mr} or Γ^k_{mc} may be declared finished except for requirement 4 of that definition. If so, then associate a cost of 6 with traveling in a direction that would keep the robot in that mini-row or mini-column.

5. Use Definition 2 to detect any other finished cells and associate a cost of γ with any found.

6. Interrogate neighboring cells to determine the least costly direction in which to travel. Thus, the total cost of traveling in the direction dir is (provided there are no obstacles:)

$$\begin{aligned} direction\_cost(dir) &= cell\_cost(dir) + \beta\_cost(dir) + \\ &\quad \gamma\_cost(dir) + \\ &\quad \delta\_cost(dir). \end{aligned}$$

7. Choose the direction that will bring the robot to a boundary or obstacle more quickly in case of an up-down or right-left tie. In case of any other tie, choose the direction according to the following priority: up, down, right, left.

In both Definitions 1 and 2 above, we require that the robot not be on any of the cells that we are testing. This requirement is necessary because otherwise, it is possible for other cells around the robot to be declared finished as well, causing finished cell costs to cancel leaving the robot with no new information. Step 4 detects this situation and provides incentive for the robot to leave the mini-row or mini-column so that finished cells can form. By doing this, we are effectively looking one time unit ahead to the next robot movement, encouraging the formation of finished cells.

The parameters for the algorithm should be assigned values according to the following relationship:

$$\alpha < \beta < \delta < \gamma$$

signifying the order of importance of the corresponding cost functions. A difference factor of three in costs works well in experiments.

5 Results

The algorithm was simulated on five different terrains and the results are given in Figure 5. The color code indicates the final robot position, the obstacle arrangement, and the number of times each free cell was covered. In each case, the robot was initialized to begin at the lower left hand corner of the terrain.

The first column of pictures shows the performance of the base algorithm. Note that in each of the five cases, the base algorithm was able to cover the entire set of free cells regardless of the obstacle arrangement. The second column shows the performance of the base algorithm with look ahead in space, and the third column shows the performance of the base algorithm with look ahead in space and time.

Ideally we would like the robot to cover each cell only once. However, given the obstacle arrangement, even an optimal algorithm may have to retrace part of its path. Two statistical measurements were made to compare the performances. Table 1 gives the average number of times each cell was covered. A more meaningful measure of the performances, however, is to calculate the RMS deviation of each cell coverage from 1. Thus in an ideal case, this number should be zero, indicating that all cells were covered exactly once. We calculate the deviation as:

	Mean		
Terrain	Alg 1	Alg 2	Alg 3
1	1.626	1.106	1.084
2	2.534	1.687	1.307
3	3.516	1.693	1.405
4	2.681	1.864	1.319
5	2.639	1.551	1.356

Table 1: Average cell coverage

$$\begin{aligned} \sigma &= \sqrt{E\{(x-1)^2\}} \\ &= \sqrt{E\{x^2\}-2E\{x\}} + I \end{aligned}$$

where x is the number of times a cell was covered, $E\{x\}$ is the mean value of x, and $E\{x^2\}$ is the mean value of x^2. Thus, it can be seen in Table 2 that the performance of the base algorithm has increased with the addition of each component., resulting in a σ closer to the ideal value of 0.

Although Figure 5 shows the results of the coverage of explored terrain problem, the algorithm generates the same results for the coverage of unexplored terrain problem! For the search problem, however, the performance of the algorithm is dependent on whether a terrain model is available or not.

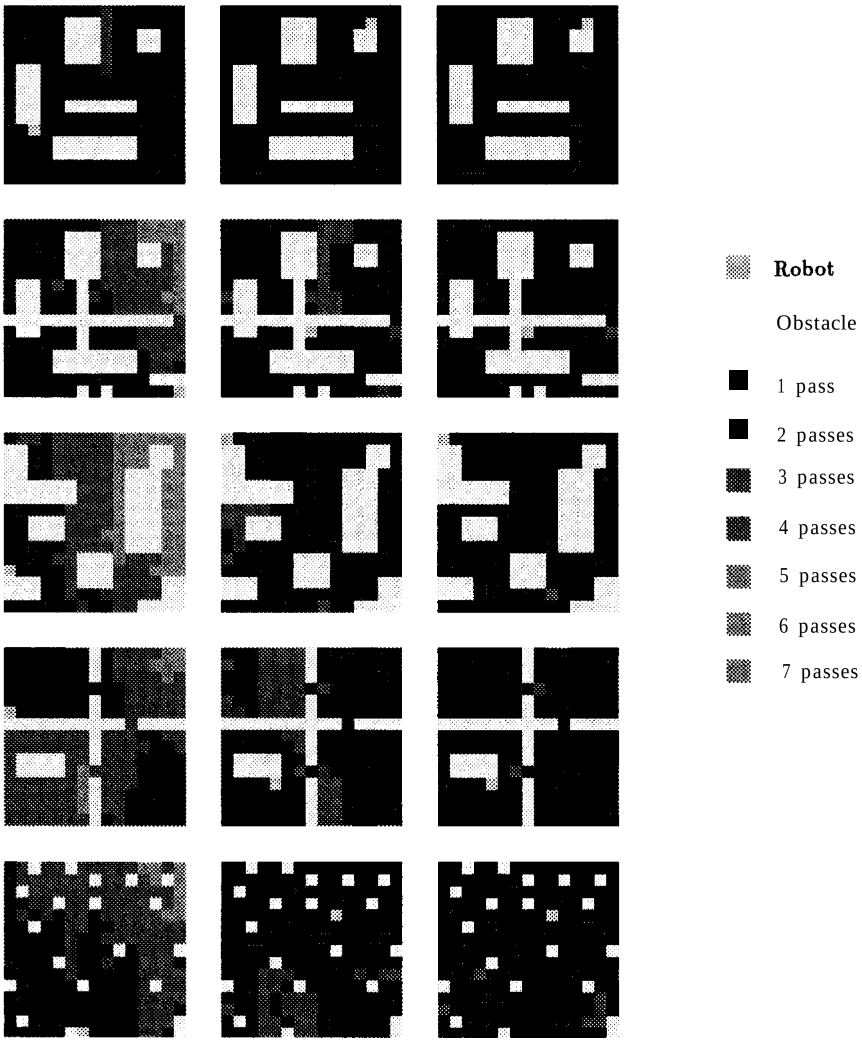

Figure 5: Results for the coverage problem

6 Modifications for Search

As opposed to the coverage problem, the search problem does not require the robot to physically cover every free cell in the terrain. On the contrary, it is only required to inspect free cells until the target is located. The following step will make this modification.

0. "Look" in all four directions for the target. If the target is detected in that direction, proceed in that direction skipping the other steps. Otherwise, label the cells that were inspected as having been covered. Use Definition 1to check for the formation of finished cells starting with the mini-row and mini-column of the farthest cell inspected and working back. Associate a cost of γ with any finished cells found.

Note that there is no need to be aware of the existence of the target or know which direction it is in. If the stopping condition (see section) is met without the detection of the target, it may be concluded that the target is not present.

Terrain	σ		
	Alg 1	Alg 2	Alg 3
1	0.832	0.326	0.289
2	1.988	0.953	0.565
3	2.936	0.922	0.647
4	2.056	1.187	0.583
5	1.963	0.913	0.644

Table 2: RMS deviation of cell coverage from 1

Results are not provided for the search problem since the performance is so dependent on both the location of the target in the terrain, and on the range of the sensor (especially in the search in the unexplored terrain problem.) However, since the search problem is a subset of the coverage problem, the performance of the search algorithm is always better than or equal to the performance of the coverage problem.

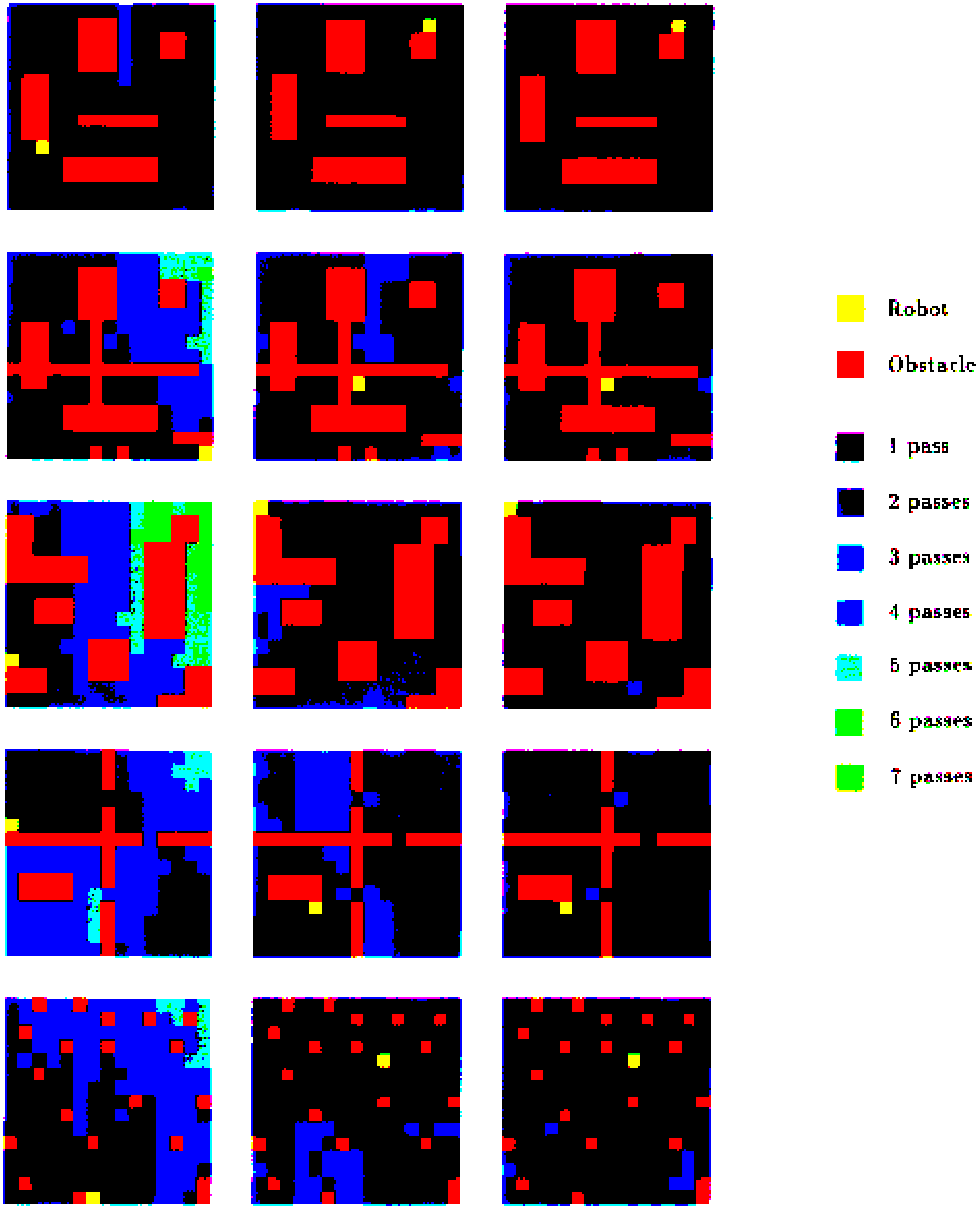

Figure 5: Coverage in Explored and Unexplored Terrains

7 Modifications for Unexplored Terrains

In the more frequent case of unexplored terrains, we may no longer assume that we have a model of the terrain. As a result, we require the robot to build such a model, as it explores the terrain, for its immediate and possible future use. This model will be stored in the form of a two dimensional array where each array element holds information on a corresponding cell in the terrain. We will start by initializing the array elements to NOT-SEEN. Then, at the beginning of each iteration, prior to the execution of any of the steps of the algorithm, perform the following step:

-1. "Look" in all four directions. Put the information obtained from each cell, for example FREE-CELL, OBSTRUCTED-CELL, or TARGET, into the corresponding array element.

Only enter information available from cells that fall within the robot sensor range. Note that for unexplored terrain problems, the algorithm requires a robot sensor range of at least one cell. This requirement is explained in the next section.

Another modification that must be made is in determining finished cells. When determining whether a group of cells is finished, information may be required from a cell that is marked NOT-SEEN. In that case, do not declare that group of cells finished.

8 The Stopping Conditions

The stopping condition for explored terrain problems is obvious since all the free cells are known. Thus, for the coverage of the explored terrain problem, stop when all the free cells have been covered. For the search in the explored terrain problem, since we label the cells that were inspected as having been covered, the stopping condition is the same.

As opposed to the explored terrain problems, the unexplored terrain problems do not have an obvious stopping condition. The problem lies in the robot's not being able to "see" every cell in the terrain due to obstruction. Figure 6 shows examples of these "invisible" cells. Hence, we will not be able to stop when all the cells within the boundary have been identified and all the free cells covered or inspected. However, assuming a sensor range of at least one cell, the stopping condition for the coverage in unexplored terrain problem is simple: repeat until all the identified free cells have been covered.

We prove that the above condition is sufficient by contradiction. Let $\mathbf{x} \in R$ denote the set of free cells in the terrain R. Let $\mathbf{y} \subset \mathbf{x}$ be the set of free cells that are covered and let $\mathbf{p} \subset \mathbf{x}$ be the set of free cells that are still marked NOT-SEEN. Based on the condition in Theorem 1, there must be a cell $p_i \in \mathbf{p}$ adjacent to a cell $y_j \in$ y. In that case, with a sensor range of at least one cell, p_i would have been identified as a free cell and the algorithm would have continued.

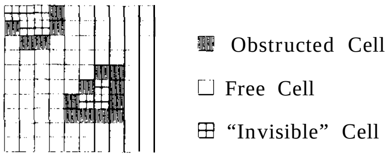

Figure 6: Example of "invisible" cells

For the search in unexplored terrain problem, the stopping condition may be determined with the addition of the following operations:

- When "looking" in each of four directions (step 0), if an obstacle or boundary is detected by the sensor, indicate for all free cells from the current position to the obstacle or boundary whether they were scanned horizontally or vertically.

- Tag the current cell as having been scanned in both the horizontal and vertical directions.

The stopping condition is now simple: repeat until all the identified free cells have been scanned in both the horizontal and vertical directions. If this condition is met without the detection of the target, it may be concluded that the target is not present in the terrain.

We prove that the above condition is sufficient by contradiction again. Let $\mathbf{x} \in R$ denote the set of free cells in the terrain R. Let y $\subset$ x be the set of free cells that were scanned both horizontally and vertically and let $\mathbf{p} \subset \mathbf{x}$ be the set of free cells that are still marked NOT_SEEN. Based on the condition in Theorem 1, there must be a cell $p_i \in \mathbf{p}$ adjacent to a cell $y_j \in \mathbf{y}$. If y_j was tagged because of the second operation, then p_i would have been identified as a free cell (assuming a sensor range of at least one cell.) Otherwise, if y_j was tagged because of the first operation, p_i would have been scanned in at least one direction since we tag all cells until detecting an obstacle or boundary. As a result, the algorithm would have continued. More informally, we argue that by tagging cells only if an obstacle or boundary is detected, we are guaranteed to have "looked" around every corner for the target.

Note that the stopping condition efficiency for the search problem depends on the robot sensor range. In the worst case, with a robot sensor range of just one cell, the entire terrain must be covered before we can conclude that the target is not present. With longer sensor range, a better chance exists of detecting an obstacle or boundary so that we may tag cells to have been scanned both horizontally and vertically without covering those cells.

When the stopping condition is met in the coverage of unexplored terrain problem, the remaining cells that are still marked NOT-SEEN should be identified as OBSTRUCTED-CELL in order to complete the terrain model. In the case of the search in unexplored terrain problem, the above method may be used to obtain a complete terrain model only if the target is not found, thus guaranteeing that the entire terrain was thoroughly searched (explored.)

9 Conclusion

An algorithm is presented in this paper that is both simple to understand and simple to implement, yet it solves four different problems, each of which is a conceivably difficult problem. One reason for the simplicity of this algorithm is in its use of indirect control. By using indirect control, we are not attempting to solve the problem explicitly, but we are allowing the algorithm to solve the problem for us. This strategy is consistent with minimization-search approaches in that the robot motion is always a step in the direction of steepest descent. Following the indirect control philosophy, however, we allow the algorithm to build the energy surface dynamically rather than specifying that surface a ***priori.*** Using this approach, we were able to guarantee complete coverage and a thorough search in terrains where the free cells are connected as defined in Theorem 1.

Thinking in terms of indirect control when designing an intelligent system helps for two reasons. The first reason is that the algorithm is simplified because there are fewer variables involved (as opposed to planning a complete path for the robot, for example.) The second reason is that indirect control algorithms are more intuitive. For example, describing planetary motion about the Sun in terms of a distortion of space time is more intuitive than describing it in terms of Kepler's laws. By the same analogy, we are distorting the terrain on which the robot navigates by associating costs with traversing various regions of it. As a result, the robot is forced to "fall" towards cells with lower cost just as the planets "fall" towards the Sun due to a distortion of space time.

We note that the coverage of explored terrain problem is equivalent to the well known traveling salesman problem constrained by only allowing the salesman to travel to adjacent cities and defining cities to be free cells. Also, we note that the end result of our algorithm on unexplored terrains is equivalent to the end result of region growing[3] or connected component algorithms used in computer vision. At the end of a coverage of unexplored terrain problem, for example, the terrain is divided into traversable and non-traversable regions. Although the result is the same (connected components,) direct use of either raster scan methods from vision [ll] or graph-theoretic methods [3] is not appropriate to this problem, since the robot cannot "scan across" obstacles, and the graph must be built as part of the exploration process.

References

[l] R. Brooks. Solving the find-path problem by good representation of free space. ***IEEE Trans. Systems, Man and Cybernetics,*** 13(2):190--197, 1982.

[2] R. Chattergy. Some heuristics for the navigation of a robot. The International J. of Robotics Research, 4(1):59-66, 1985.

[3] J. E. Hopcroft and J. D. Ullman. Set merging algorithms. ***SIAM J. Comput., 2(4):294-303,1973.***

[4] Sungtaeg Jun and Kang G. Shin. A probablistir approach to collision-free robot path planning. 1988 ***IEEE International Conference on Robotics and Automation, 1:220-225,*** 1988.

[5] S. Kambhampati and L. S. Davis. Multiresolutional path planning for mobile robots. ***IEEE J. of Robotics and Automation,*** R.A-2:135-145, 1986.

[6] Yutaka Kanayama. Least cost paths with algebraic cost functions. ***1988 IEEE*** International Conference on ***Robotics*** *and* Automation, 1:75-80, 1988.

[7] T. Loxano-Perez and M. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. C. ***A CM,*** 22:560-570, 1979.

[8] E. Moore. Shortest path through a maze. ***Annals of the Computation Lab. of Harvard Univ., 30:285-292,*** 1959.

[9] P. F. Rowat. Representing spatial experience and solving spatial problems in a simulated robot environment. Doctoral ***thesis, University of British Columbia,*** 1979.

[10] J. T. Schwartz and C. Yap, editors. ***Advances in Robotics.*** Lawrence Erlbaum Associates, 1987.

[11] W. E. Snyder and C. D. Savage. Content-addressable read/write memories for image analysis. ***IEEE Trans. Computers, 31:963-968, 1982.***

[12] I. Sutherland. A method for solving arbitrary-wall mazes by computer. ***IEEE Trans. Computers, C-18(*** 12):1092-1097, 1969.

[3] Region growing **refers to the** grouping **of individual pixels into distinct regions.**