# Coverage Path Planning: The Boustrophedon Cellular Decomposition

**Howie Choset**
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
U. S. A.

**Philippe Pignon**
Nomadic Technologies, Inc.,
Mountain View, CA 94043-1605
U. S. A.

## Abstract

Coverage path planning is the determination of a path that a robot must take in order to pass over each point in an environment. Applications include vacuuming, floor scrubbing, and inspection. We developed the boustrophedon cellular decomposition, which is an exact cellular decomposition approach, for the purposes of coverage. Each cell in the boustrophedon is covered with simple back and forth motions. Once each cell is covered, then the entire environment is covered. Therefore, coverage is reduced to finding an exhaustive path through a graph which represents the adjacency relationships of the cells in the boustrophedon decomposition. This approach is provably complete and Experiments on a mobile robot validate this approach.

## 1 Introduction

*Coverage path planning* determines a path that guarantees that an agent will pass over every point in a given environment. This procedure allows for a variety of applications. Naval applications include mine-countermeasure missions and continental shelf oceanographic mapping. Commercial applications include contamination cleanup, floor scrubbing, crop plowing, and bridge inspection. Without a coverage algorithm, these applications cannot be handled. Most current coverage path planners are rudimentary at best because they are based on heuristics. Using such approaches for mine sweeping is akin to performing this operation with a faulty mine detector. Therefore, the coverage path planning algorithm described in this pare is *complete*; that is, in finite time, it will find a coverage path or determine that none exists.

Our approach exploits a geometric structure termed an *exact cellular decomposition*, which is the union of non-intersecting regions composing the target environment. Each region is termed a *cell*, and the union of the cells fills the environment. In each cell, a coverage path can be readily determined, such as simple back-and-forth motions; thus coverage path planning reduces to planning motions from one cell to another. This work will develop a new cellular decomposition, termed the *boustrophedon decomposition* and apply it to coverage path planning.

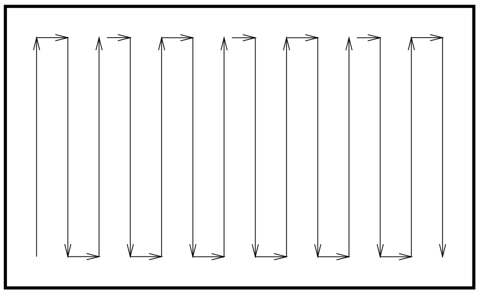

Figure 1. Bountrophedon Path.

Boustrophedon was first used in the English language in 1699 and it literally means, "the way of the ox." Typically, when an ox drags a plow in a field, it crosses the full length of the field in a straight line, turns around, and then traces a new straight line path adjacent to the previous one. By repeating this procedure, the ox is guaranteed to cover (and thereby to plow) the entire field. See Figure 1.

The boustrophedon cellular decomposition is a new type of decomposition where the robot's free space is broken down into cells such that the robot can cover each cell with back-and-forth boustrophedic motions. Once a robot covers each cell, it has covered the entire free region of an environment. This approach has been validated with simulations and a Nomadic 200 mobile robot base.

## 2 Background Work

### 2.1 Prior Work in Coverage

Previous work in coverage include applications such as vacuum cleaning [Colegrave and Branch, 1994] and floor cleaning. In these approaches, the path must be explicitly programmed into the robot; i.e., they do not use an algorithm to generate the coverage path, but instead prescribe one "by hand." Furthermore, these algorithms rely on landmarks deployed in the environment. Some modern agricultural operations represent a significant opportunity for coverage applications whose coverage path is easy to automatically generate. The Demeter project [Ollis and Stentz, 1996] is used to harvest large fields; in this approach the robot simply uses vision to guide its path alongside the previous cut crop line and can only cover rectangular fields.

A floor coverage approach that considers nonholonomic constraints is described in [Hofner and Schmidt, 1995]. In this work, a set of templates is used to cover only a bounded region that is free of obstacles. These templates are used to accommodate the nonholonomic constraints of the robot, and thus may be useful for planning back and forth motions within each cell of the proposed approach. However, its limitation is that it cannot plan paths when obstacles are present.

The coverage algorithm described in [Zelinsky *et al.*, 1993] is well suited to unstructured environments. Although it is complete, it achieves floor coverage in a discretized environment (i.e., it is resolution complete). A similar approach, without proof, with cooperating robots was mentioned in [Kurabayashi *et al.*, 1996]. Finally, Lumelsky et. al. [Hert *et al.*, 1996] produced an algorithm that is similar to the proposed approach in the planar case. Although the proposed algorithm produces nearly the same path as Lumelsky' group's in the planar case, the approach described in this paper is easier to implement because it has two cases whereas their approach contains a series of special cases. Finally, the approach in [Hert *et al.*, 1996] is *not* complete. The primary contribution of the algorithm supplied by [Hert *et al.*, 1996] is that it is incremental, and thus may lead to a sensor based implementation on a mobile robot.

### 2.2 Exact Cellular Decomposition

The approach to coverage used in this paper is an adaptation of an existing complete motion planning scheme, termed an *exact cellular decomposition.* Cellular decomposition is a motion planning technique in which the free configuration space (set of all robot configurations where the robot does not overlap an obstacle) is decomposed into cells such that the union of the cells is the original free space. Each cell can be represented as a node in a graph, where adjacent cells have an edge connecting their corresponding nodes. This graph is called an adjacency graph. If each cell can be covered by the robot, then the floor coverage problem reduces to determining a walk through the adjacency graph that visits each node at least once, i.e., the traveling salesman problem, for which a solution (possibly sub-optimal) always exists.

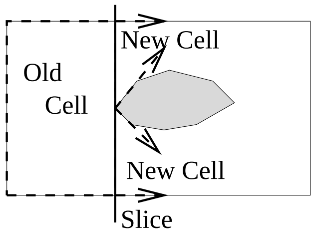


Figure 2. In Event.

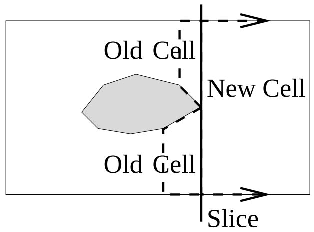


Figure 3. Out Event.

One popular cellular decomposition technique, which can yield a complete coverage path solution, is the trapezoidal decomposition [Latombe, 1991] (also known as the slab method [Preparata and Shamos, 1985]) in which the robot's free space is decomposed into trapezoidal cells. Since each cell is a trapezoid, coverage in each cell can easily be achieved with simple back and forth motions (See Figure 1). Coverage of the environment is achieved by visiting each cell in the adjacency graph.

The trapezoidal decomposition approach assumes that a vertical line, termed a *slice*, sweeps left to right through a bounded environment which is populated with *polygonal* obstacles. Cells are formed via a sequence of *open* and *close* operations which occur when the slice encounters an *event*, an instance in which a slice intersects a vertex of a polygon. There are three types of events: IN , OUT , and MIDDLE . Loosely speaking, at an IN event the current cell is closed (thereby completing its construction) and two new cells are opened (thereby initiating their construction) See Figure 2. An OUT event is the reverse: two cells are closed, and a new one is opened. See Figure 3. The IN event can be viewed as one cell breaking up into two cells, whereas the OUT event is when two cells merge into one. At a MIDDLE event, the current cell is closed, and a new one is formed. The result of these operations is a freespace that is broken down into trapezoidal cells.

The terrain coverage system of VanderHeide and Rao [VanderHeide and Rao, 1995] is based on a trapezoidal decomposition of a planar environment populated with one or two well-separated obstacles. The advantage of this system is that it is sensor based.

Unfortunately, the trapezoidal approach requires too many redundant back and forth motion to guarantee

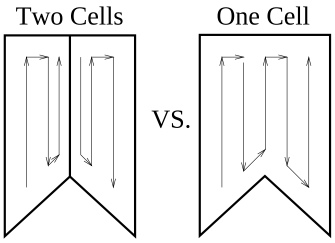


Figure 4. Fewer cells is better.

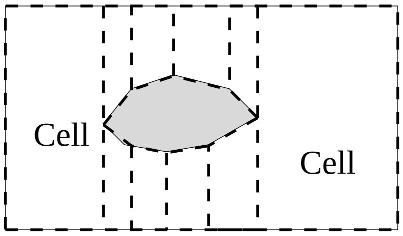


Figure 5. Trapezoidal Decomposition

completeness. In the left hand side of Figure 4 the robot needs to make one additional lengthwise motion to cover the remaining portion of the trapezoidal cell. This can be viewed as part of the cost in guaranteeing the robot exhaustively covers the entire environment. Another drawback of the trapezoidal approach is that it requires the environment to be polygonal.

## 3 Contributions

The boustrophedon cellular decomposition, introduced in this paper, is an enhancement of the trapezoidal decomposition and is designed to minimize the number of excess lengthwise motions, as described in the previous paragraph. In essence, all cells between IN and OUT events are merged into one cell. Compare the trapezoidal decomposition in Figure 5 with the boustrophedon decomposition in Figure 6. Note that the boustrophedon decomposition has a fewer number of cells.

The advantage of having a fewer number of cells is that the number of back-and-forth boustrophedon motions can be minimized. For example, consider two adjacent trapezoidal cells whose widths are each two and a half times the width of the robot. In order to cover each trapezoid, the robot must make three passes, for a

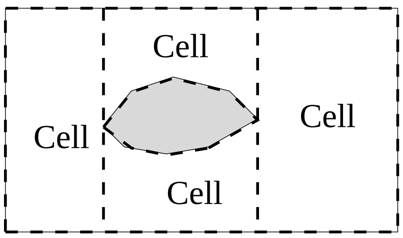


Figure 6. Boustrophedon Decomposition

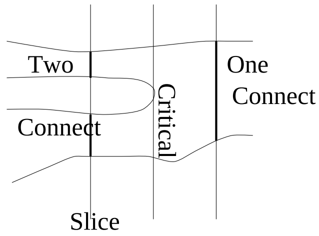


Figure 7. Critical points are points where the connectivity of a slice changes, i.e., they are events.

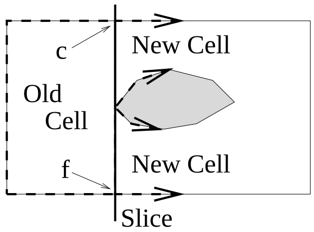


Figure 8. In Event.

total of six lengthwise motions. With the boustrophedon decomposition approach, the two cells are merged into one cell which is a monotone polygon and requires five passes to cover. See Figure 4.

Instead of exploiting the structure of polygons to determine IN and OUT events, this approach will rely on changes in connectivity of a slice to determine the existence of an event. Typically, this is called a critical point, which are used in roadmap motion planning techniques such as Canny and Lin's "Opportunistic Path Planner" (OPP) [Canny and Lin, 1990], [Canny and Lin, 1993], which is itself based on Canny's Roadmap Algorithm [Canny, 1988]. Now, the robot can perform coverage in curved, or even sampled, environments. See Figure 7.

## 4 Algorithm Overview

The boustrophedon cellular decomposition approach is similar to the trapezoidal one. Again, a slice sweeps through a bounded planar environment populated with polygonal obstacles. Just like the trapezoidal decomposition, at an IN event, where the connectivity of the slice increases, the current cell is closed and two new cells are opened (Figure 8). Conversely, at an OUT event, where the connectivity of the slice decreases, the two current cells are closed and one new cell is opened (Figure 9).

The difference between the trapezoidal decomposition and boustrophedon decomposition approach is with the middle events: at the MIDDLE events, do not open nor close a cell, but rather simply update the current cell. Essentially, cells are opened and closed when there is a change in connectivity of slice. See Figure 6.

As the decomposition is computed, the adjacency

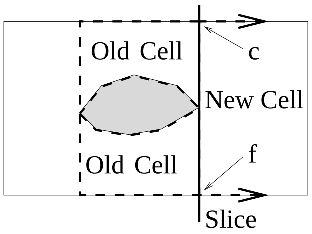


Figure 9. Out Event.

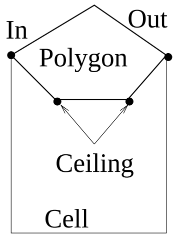


Figure 10. Floor and Ceiling of Cell

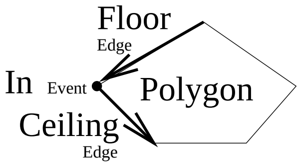


Figure 11. In Event.

graph is also determined. Again, each cell is a node in the graph and an edge connects the nodes of adjacent cells. A depth-first-like graph search algorithm outputs a path list that represents an exhaustive walk through the adjacency graph. A walk through the path list constitutes an exhaustive walk through the adjacency graph.

Finally, the actual path for the robot to take is computed using the above described path list. When the robot enters an "uncleaned" cell, the boustrophedic motion is planned, and then a path to the next cell in the path list in planned. When the robot enters a "cleaned" cell, it simply plans a path through that cell to the next cell in the path list. These two actions are repeated until the end of the path list is reached, i.e., until each cell has been cleaned.

## 5 Algorithm Specifics

This section contains details for implementing the boustrophedon decomposition in a known polygonal environment. Current work includes implementing the algorithm in a curved environment using sensors. The configuration space obstacles are computed by enlarging the obstacles by the radius of the robot, which is circular. The resulting generalized polygons (sequences of segments and arcs of circles) are then approximated by polygons.

### 5.1 Events

In our implementation of the boustrophedon decomposition method, we replaced the middle event with two more types of events: FLOOR and CEILING . The FLOOR events correspond to vertices that are on the *top* of the polygonal obstacle and the CEILING events correspond to vertices that are on the *bottom* of the obstacle. In this way, the FLOOR and CEILING events correspond to the floor and ceiling, respectively, of the cells that are being incrementally generated. See Figure 10.

The input to the algorithm is a list of polygons whose vertices are listed in counter-clockwise order. The algorithm first creates a list of events from the list of polygons. The polygons are considered in no special order, but in our implementation we make a generic assumption that no two IN nor two OUT events have the same $x$-coordinates.

Recall that an event is a vertex of a polygon and some additional information; specifically, the event structure contains the location of the event, its type, and pointer(s) to the edge (or edges) that is (are) associated with it. The event structure has up to two types of pointers to edges: floor pointers and ceiling pointers. An IN event's ceiling pointer points to the next edge emanating from the event and the floor pointer points to the previous edge terminating at the event. See Figure 11. Conversely, the OUT event's floor pointer points to the next edge emanating from it and the ceiling pointer points to the edge terminating at the event. A CEILING event only has a ceiling pointer which points to the edge emanating from the event' a FLOOR event only has a floor pointer which points to the edge terminating at the event.

When considering a particular polygon, the algorithm first finds the IN event of the polygon. The algorithm walks through the vertex list of a polygon until it encounters the left-most vertex. This vertex, and its related information, is inserted into an events list. Since the vertices are ordered in a counter clockwise fashion, the next sequence of vertices are CEILING events. Recall that although these vertices correspond to the underside of the polygon, and they are CEILING events because they correspond to the ceiling of the cell that is immediately below the polygon.

The algorithm walks through the polygon list, inserting each vertex as a CEILING event, until the algorithm encounters the right-most vertex. This vertex, and its associated information, is inserted into the events list as an OUT vertex. The remaining vertices correspond to FLOOR events.

As the events are encountered, they are inserted into an ordered events list sorted by the $x$-coordinate of the event. The insertion process is $O(n \log n)$ where $n$ is the total number of edges (or vertices) in polygonal environment.

## 5.2 Cells

A cell can be represented by two lists: a list of floor edges and a list of ceiling edges, both of which bound the cell. Therefore, the cell structure contains two pointers to a lists of edges: a floor pointer and a ceiling pointer. The cell structure also contains a linked list of pointers to neighboring cells. Finally, the cell structure has two flags: visited and cleaned, which are used later on in the algorithm.

The cells of the boustrophedon decomposition are computed in an incremental manner via a sweepline approach. Sweeping the environment is akin to visiting, in order, each of the events of the events list because the events list is already sorted.

The first cell is the left-most cell. It is assumed in our implementation that the left most cell is artificially opened before the actually sweeping procedure (i.e., sweeping starts to the left of the left-most IN event). It is also assumed that the environment is bounded above by an edge and bounded below by an edge. So, the first cell's floor and ceiling pointers point to these bounding edges.

The first real event is IN event. At the IN event, the intersection of the slice and floor and the intersection of the slice and the ceiling of the current cell are determined. Denote these points at $f$ and $c$. See Figure 8. Typically, the floor and ceiling of the current cell have multiple edges, so the intersection points $f$ and $c$ are the end points of the *last* floor and ceiling edges, respectively, of the current cell. Now, all of the floor and ceiling segments of the current cell are determined and the cell is considered to be closed.

Next, two new cells are to be opened: a bottom cell and a top cell. The starting point of the first edge in the floor of the bottom cell is the point $f$ and the starting point of the first edge in the ceiling of the bottom cell is the event. The floor pointer of the bottom cell is set to the floor pointer of the previously closed cell and the ceiling pointer of the bottom cell is set to the ceiling pointer of the open event. Conversely, for the top cell, the starting point of the first edge in the floor is the event whereas the starting point of the first edge in the ceiling is the point $c$. Here, the new floor pointer is set to the floor pointer of the event and the new ceiling pointer is set to the ceiling pointer of the previously closed cell.

When a FLOOR event is encountered, the floor pointer of the current cell is updated. Specifically, the floor edge associated with the event is added to the floor edge list of the current cell. Similarly, when a CEILING event is encountered, the ceiling edge associated with the event is added to the ceiling edge list.

Finally, when an OUT event is encountered, two cells are closed and a new cell is opened. Again, let the bottom cell and the top cell denote the two cells that are closed at the OUT event and the new cell denote the cell that is opened at the IN event. Let $f$ be the intersection of the current slice with the current edge in the floor list of the bottom cell, and let $c$ be the intersection of the current slice with the current edge in ceiling list of the top cell. The point $f$ is the end point of the last segment in the floor list of the bottom cell and the event location is the end point of the last segment in the ceiling list of the bottom cell. Likewise, the event location is the end point of the last segment in the floor list of the top cell and $c$ is the end point of the of the last segment in the ceiling list of the top cell. Once all of the floor and ceiling segments of the bottom and top cells are determined, the bottom and top cells are closed. See Figure 9.

Next, a new cell is to be opened. The starting point of the first floor segment is $f$ and the starting point of the first ceiling segment is $c$. The floor pointer of the new cell is set to the floor pointer of the previous bottom cell and the ceiling pointer of the new cell is set to the ceiling pointer of the previous top cell.

The cell neighbor list is also incrementally constructed. Recall, each cell has a pointer to a list of neighboring cells which is updated at the IN and OUT events. The goal is to insert the neighboring cells into the neighbor list so that neighboring cells are in counter-clockwise order around the current cell. At an IN event, the current cell is split into two new cells: a bottom and a top. First, a pointer to the top cell is inserted to the front of the current cell's neighbor list and then a pointer to the bottom cell is inserted to the front of the new neighbor list. The result is

$$\text{neighborlist} = \text{bottom} \rightarrow \text{top} \rightarrow \text{old neighborlist}.$$

At an OUT event, a bottom and a top cells are merged into a new cell. First, a pointer to the top cell is inserted to the end of the neighbor list of the new cell and then a pointer to the bottom cell is inserted to the end of the neighbor list. The result is

$$\text{neighborlist} = \text{old neighborlist} \rightarrow \text{top} \rightarrow \text{bottom}.$$

This process produces a neighbor list whose elements are adjacent cells ordered in a counter clockwise fashion starting from the lower right of the current or new cell.

After all of the events in the events list are visited, all of the cells and their adjacency relationships are computed; in effect, the boustrophedon decomposition and its adjacency graph have been determined.

## 5.3 Coverage

With the decomposition and adjacency graph, the robot can now plan a path that covers the environment. This is done in two steps: a path is found in the in the adjacency graph that visits each node, and then the explicit robot motions are computed within each cell (i.e., node).

The determination of an optimal path that visits each node in a generic graph is the classical traveling salesman problem which is an NP-complete problem. A *path list*, a list that represents an exhaustive walk through the adjacency graph, is computed using a depth-first-like search algorithm.

1. Start with any cell in the decomposition. Insert it into the path list. Mark it as visited.
2. Go to the first unvisited cell in the neighbor list of the current cell (i.e., go to the first counter-clockwise unvisited cell). Insert this cell into the beginning of the path list and mark it as visited.
3. Repeat this procedure (i.e., goto step 2) until a cell with all visited neighbors is encountered.
4. At this point, back track until a cell with unvisited neighbors is encountered. This back tracking is achieved by walking forward through the path list, inserting each element that is visited *to the front* of the path list, until an element with an unvisited neighbor is encountered. Insert this element to the front of the path list and repeat the above procedure (i.e., goto step 2).
5. If no cell with an unvisited neighbor is found during the back-tracking process, then all cells in the adjacency graph have been visited.

Using the path list, the motions for the robot are computed in a two-step process. First, if a cell is not cleaned, the cell is marked cleaned and the actual boustrophedon (back and forth) motion for the robot is computed for the cell. Typically, the step size (i.e., the distance between two parallel line segments) for the boustrophedon motion is about the width of the robot. Second, a path to the next cell in the path list is determined. If the cell is already cleaned, then a path to the next cell in the path list is planned.

## 6 Simulation and Experiments

Figure 12 contains a floor plan of two obstacles. Figure 13 and 14 contains intermediate results of the floor coverage algorithm. In Figure 13, two cells are already covered and in Figure 14, all but two cells are covered. The final coverage result can be seen in Figure 15.

It can be seen in Figure 15 that this approach has some problems near obstacle edges that form acute angles with the boustrophedon back and forth paths. To partially alleviate this problem, when the robot travels through a cleaned cell, it travels near the boundary of an obstacle. Future implementation will include one

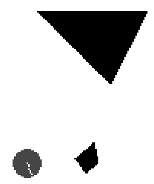

Figure 12. Floor plan bounded above and below by line segments. Black polygons are obstacles.

Figure 13. Two cells have been covered.

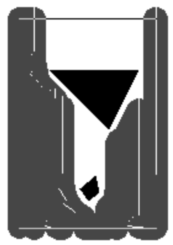

Figure 14. Coverage almost complete.

additional pass where each obstacle is specially circumnavigated. For applications such as vacuuming, this additional approach is reasonable because most vacuuming applications require a different sucking mechanism near the boundary of the environment.

The algorithm was also run on a Nomadic Technologies mobile robot base in a carpeted environment with cardboard obstacles that was represented by the above simulation. The implementation works in small environments but had some problems with dead reckoning in larger ones. In particular, not all lines were perfectly parallel. This approach can benefit from the vision based techniques used in the Demeter project [Ollis and Stentz, 1996]. Also, out experiments showed that this approach is extremely sensitive to initial conditions and

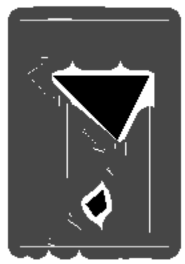

Figure 15. Complete coverage.

thus future work must make this procedure robust to small changes in initial conditions.

## 7 Conclusion and Future Work

This paper describes a new type of floor coverage algorithm that is complete. That is, the robot, in theory, is guaranteed to follow a path such that every point in the environment is passed over by the robot. The algorithm is based on a new type of exact cellular decomposition approach termed the boustrophedon cellular decomposition. Boustrophedon means the way of the ox; boustrophedon motion is back and forth ox-like motions. A robot can easily plan a boustrophedon motion in each cell of the boustrophedon decomposition. Once each cell is covered, the entire environment is covered.

Experimental results on a mobile robot has validated this approach and pointed out some avenues of future work. This approach sometimes skips portions of the environment near the boundaries of obstacles because of the discretization of the side step. If the robot has a shorter side step, these uncovered areas are reduced. There is a tradeoff time/coverage here. Nevertheless, a simple obstacle following algorithm, after the main coverage is completed, will alleviate this problem. Such a solution is consistent with normal vacuum cleaning where the portion of the floor near the walls requires an additional pass with a vacuum cleaner.

Near-term research also includes improving the boustrophedon cellular decomposition by allowing for the definition of bigger (and thus fewer) cells. For example, in Figure 6 the bottom three cells can be merged into one cell in which boustrophedon motion can be planned. Furthermore, since a cell is open or closed when there is a change in the connectivity of the sweep line, this algorithm can be easily modified to environments with curved obstacles.

Also, there are issues in optimization that need be considered. The first issue deals with the development of metrics which gauge heuristic graph searches of the adjacency graph. Such metrics include: path length, area of re-covered floor space, time, etc. Another optimization issue deals with determining the angle for the sweepline; some environments may be better suited to a horizontal sweep line.

Another issue deals with material removal. In the case of the vacuum cleaner, this point is not important. However, in the case of snow removal, a snow removing robot may have to plan optimal paths to transfer the snow from the coverage site. This suggests the use of multiple robots: one robot to remove the snow from the ground, and another robot to transfer the snow to a central dumping zone.

The long-term goal of this work is in sensor based floor coverage, which is the determination of a coverage path from solely line of sight sensor information. Sensor based floor coverage is useful even when full knowledge of the world is available to the robot, but is too cumbersome to input into the robot. For example, an automatic vacuum cleaner would not be a marketable product if each user had to program a house CAD model (if it existed) into the robot. The sensor based approach will be based on Rimon and Canny's roadmap work which uses critical points to guarantee the connectivity of a roadmap.

## References


[Canny and Lin, 1990] J.F. Canny and M.C. Lin. An Opportunistic Global Path Planner. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1554–1559, Cincinnati, Ohio, 1990.

[Canny and Lin, 1993] J.F. Canny and M.C. Lin. An Opportunistic Global Path Planner. *Algorithmica*, 10:102–120, 1993.

[Canny, 1988] J.F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1988.

[Colegrave and Branch, 1994] J. Colegrave and A. Branch. A Case Study of Autonomous Household Vacuum Cleaner. In *AIAA/NASA CIRFFSS*, 1994.

[Hert *et al.*, 1996] S. Hert, S. Tiwari, and V. Lumelsky. A Terrain-Covering Algorithm for an auv. *Autonomous Robots*, 3:91–119, 1996.

[Hofner and Schmidt, 1995] C. Hofner and G. Schmidt. Path planning and guidance techniques for an autonomous mobile cleaning robot. *Robotics and Autonomous Systems*, 14:199–212, 1995.

[Kurabayashi *et al.*, 1996] D. Kurabayashi, J. Ota, T. Arai, and E. Yoshida. Cooperative Sweeping by Multiple Mobile Robots. In *Int. Conf. on Robotics and Automation*, 1996.

[Latombe, 1991] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[Ollis and Stentz, 1996] M. Ollis and A. Stentz. First Results in Vision-Based Crop Line Tracking. In *IEEE International Conference on Robotics and Automation*, 1996.

[Preparata and Shamos, 1985] F.P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985. p198-257.

[VanderHeide and Rao, 1995] J. VanderHeide and N. S. V. Rao. Terrain coverage of an unknown room by an autonomous mobile robot. Technical Report ORNL/TM-13117, Oak Ridge National Laboratory, Oak Ridge, Tennessee, 1995.

[Zelinsky *et al.*, 1993] A. Zelinsky, R.A. Jarvis, J.C. Byrne, and S. Yuta. Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot. In *Proceedings of International Conference on Advanced Robotics*, pages pp533–pp538, Tokyo Japan, November 1993.