The International Journal of Robotics Research

http://ijr.sagepub.com

Path Planning for Robotic Demining: Robust Sensor-Based Coverage of Unstructured Environments and Probabilistic Methods

Ercan U. Acar, Howie Choset, Yangang Zhang and Mark Schervish

The International Journal of Robotics Research 2003; 22; 441
DOI: 10.1177/02783649030227002

The online version of this article can be found at:
http://ijr.sagepub.com/cgi/content/abstract/22/7-8/441

Published by:



http://www.sagepublications.com

On behalf of:



Multimedia Archives

Additional services and information for *The International Journal of Robotics Research* can be found at:

Email Alerts: http://ijr.sagepub.com/cgi/alerts

Subscriptions: http://ijr.sagepub.com/subscriptions

Reprints: http://www.sagepub.com/journalsReprints.nav

Permissions: http://www.sagepub.com/journalsPermissions.nav

Citations (this article cites 9 articles hosted on the SAGE Journals Online and HighWire Press platforms):
http://ijr.sagepub.com/cgi/content/refs/22/7-8/441

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

Ercan U. Acar
Howie Choset
Yangang Zhang
Mark Schervish

Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213 USA

Path Planning for Robotic Demining: Robust Sensor-based Coverage of Unstructured Environments and Probabilistic Methods

Abstract

Demining and unexploded ordnance (UXO) clearance are extremely tedious and dangerous tasks. The use of robots bypasses the hazards and potentially increases the efficiency of both tasks. A first crucial step towards robotic mine/UXO clearance is to locate all the targets. This requires a path planner that generates a path to pass a detector over all points of a mine/UXO field, i.e., a planner that is complete. *The current state of the art in path planning for mine/UXO clearance is to move a robot* randomly *or use simple* heuristics. *These methods do not possess completeness guarantees which are vital for locating all of the mines/UXOs. Using such random approaches is akin to intentionally using imperfect detectors. In this paper, we first overview our prior complete coverage algorithm and compare it with randomized approaches. In addition to the provable guarantees, we demonstrate that complete coverage achieves coverage in shorter time than random coverage. We also show that the use of complete approaches enables the creation of a filter to reject bad sensor readings, which is necessary for successful deployment of robots. We propose a new approach to handle sensor uncertainty that uses geometrical and topological features rather than sensor uncertainty models. We have verified our results by performing experiments in unstructured indoor environments. Finally, for scenarios where some a priori information about a minefield is available, we expedite the demining process by introducing a probabilistic method so that a demining robot does not have to perform exhaustive coverage.*

KEY WORDS—path planning, robust coverage, sensor based, demining, probabilistic coverage

The International Journal of Robotics Research
Vol. 22, No. 7–8, July–August 2003, pp. 441-466,
©2003 Sage Publications

1. Introduction

The problem of detecting mines in a surface-laid minefield using autonomous robots is of interest to civilians and military alike. The use of robots decreases the danger and the cost involved in manual mine detection. Trevelyan (1998) gives an overview of the worldwide demining problem and outlines its challenges. We believe that the main challenge for demining is the development of better mine detection methods. Currently, metal detectors are widely used mine/unexploded ordnance (UXO) detectors. However, there are efforts to build detectors such as infrared imaging, TNT sniffing, etc. Our contribution is to design path planners that direct a mobile robot equipped with such detectors to locate all the mines in a minefield. Despite the great efforts in technology development for mine detectors, it is very likely that detectors will generate false positive or negative results that must also be accommodated by path planners.

The current state of path planning for mine/UXO clearance is to guide a robot randomly or use heuristics (Figure 1(a)). It has been believed that random strategies are suitable for inexpensive robots because of their low sensor and computational power requirements. However, these strategies do not guarantee full coverage. Healey et al. (1995) analyze such a random coverage strategy for "pick up and carry away" type unexploded ordnance clearance scenarios. They characterize performance improvements with random methods by demonstrating that the number and locations of mine disposal areas can expedite the demining process. Gage (1995) also characterizes random strategies by comparing them to complete ones. He shows that random strategies start to become as effective as complete coverage when lots of robots are used or the accuracy of the detector degrades.

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

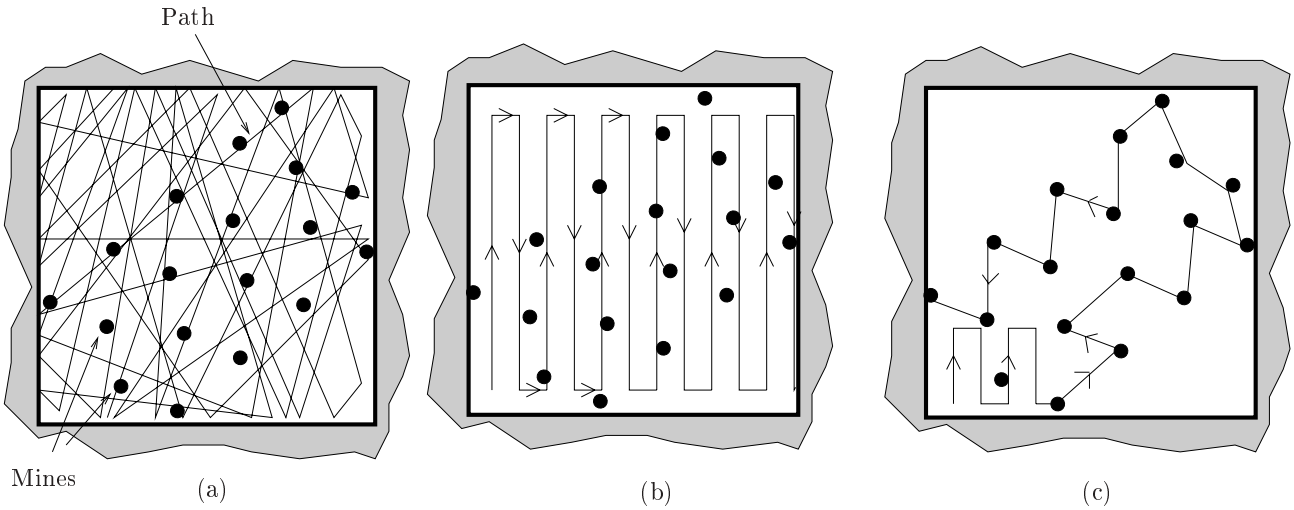


Fig. 1. The path followed by the robot in a minefield using (a) random, (b) complete and (c) probabilistic methods. (a) The robot moves around randomly in the minefield until it recovers all of the mines, which takes a considerably long time. (b) Complete coverage using back-and-forth motions guarantees that the robot will recover all the mines in less time than the random strategy requires. (c) The robot first covers a small portion of the field to extract parameters of the mine-laying pattern. Once the mine pattern is determined, the robot visits each mine location rather than covering the whole space exhaustively.

To ensure complete coverage, in our prior work we used a coordinated approach (Figure 1(b)). Our coordinated method achieves complete coverage in a considerably shorter time than the random methods, particularly in the presence of obstacles. There exists a variety of coordinated coverage algorithms (Zelinsky et al. 1993; Cao, Huang, and Hall 1988; Lumelsky, Mukhopadhyay, and Sun 1990; Hert, Tiwari, and Lumelsky 1996) using different approaches and making certain assumptions about the obstacle configurations and the sensors. Our earlier work is based on a geometric structure called cellular decomposition (Latombe 1991; Chazelle 1984), which is the union of non-overlapping subregions of the free space, called *cells*. We define our cells such that simple back-and-forth motions cover each cell, and thus complete coverage is reduced to finding an exhaustive walk through the adjacency graph that encodes the topology of the cells (Acar et al. 2002). Our framework allows us to design a path planner that uses feasible sensor systems and works in more general obstacle configurations. We do not need to place a grid on the space or to assume that only polygonal obstacles exist. These attributes of the framework become especially important for demining because minefields are inherently unstructured spaces.

For a priori unknown spaces, our prior work has prescribed a method that incrementally constructs the cellular decomposition while achieving complete coverage (Acar and Choset 2002). We verified our method in planar structured indoor environments populated with cardboard walls making the planar space "friendly" to our sonar ring. In this paper, we extend our early work, as a first step towards outdoors,[1] to the case of unstructured indoor environments where "bad" sensor readings often occur. Previously, researchers (Erdmann 1986; Lozano-Perez, Mason, and Taylor 1984) have developed algorithms that deal with uncertainty. In these algorithms, bounds on sensor uncertainty are utilized to determine the regions from which certain motions are ensured to reach a desired goal. In this work, rather than using error bounds, we identify geometrical and topological features of our algorithm that can be used to overcome failures due to sensor uncertainty without making sophisticated data processing.

Thus far, we have outlined complete coverage which does not require a priori information at all. Next, we describe a probabilistic algorithm to utilize the prior information. If we know certain information about the minefield, e.g., the mine-laying pattern, we can expedite the demining process (Figure 1(c)). For example, when ground vehicles or humans place mines, they tend to follow a regular pattern. This information, which can reflect the structure of a minefield, can be exploited to guide a robot opportunistically when the resources are limited. For example, the mines can be laid out using a regular pattern that has rows and columns. By simply identifying the intended inter-row and inter-column spacing, the robot can bypass exhaustively covering the entire region by driving to the probable mine locations. In this paper, we overview an algorithm where the robot covers a small sample of the minefield, and then "decodes" the parameters that describe the

1. In the future we are planning to implement our algorithms on outdoor demining robots developed by The Naval Explosive Ordnance Disposal Technology Division (DeBolt et al. 2000).

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

minefield based on information acquired in the covered area. Note that our complete coverage algorithm does not require any a priori information about the minefield contrary to the probabilistic algorithm. The theoretical details and derivation of this algorithm are omitted here. Once the parameters are determined the robot can visit each mine location and does not need to cover the entire field.

2. Complete Versus Random Coverage

In this section, we compare the performance of a complete strategy to a random one. Before doing so, we review an exact cellular decomposition method that allows us to devise our complete coverage path planner. The cellular decomposition is formulated such that cells with simple structure can be formed. This simple structure enables us to guide a robot using back-and-forth motions to cover each cell (and thus ensuring complete coverage is trivial by visiting all cells one by one).

2.1. Complete Coverage Using Exact Cellular Decompositions in Terms of Critical Points

The cellular decomposition is built upon "a slicing method" (Canny 1988) that sweeps a slice $\mathcal{CS}_\lambda$ (a line segment) throughout the configuration space[2] $\mathcal{CS} \subset \Re^2$ of a circular robot. The slice is the pre-image of a real-valued function $h(x) = x_1$, i.e., $\mathcal{CS}_\lambda = \{x \in \mathcal{CS} | h(x) = \lambda \text{ where } \lambda \in \Re\}$. As we sweep the slice (by varying λ), we locate the points on the obstacle boundaries where the connectivity of the slice in the free space changes. These points are termed the critical points. In actuality these are the critical points that correspond to the extrema of $h|_{\partial \mathcal{CC}}(x)$, the restriction of h to the union of the obstacle boundaries $\partial \mathcal{CC}$. Let the gradient of $h(x)$ be denoted as $\nabla h(x)$; this is the vector which points in the direction that maximally increases h. In our early work (Acar et al. 2002), we have shown that at a critical point of $h|_{\partial \mathcal{CC}}(x)$, $\nabla h(x)$ and surface normals of the obstacles $\mathcal{CC}$ are parallel to each other (Figure 2). We have described how to use the critical points (connectivity changes) to form cells using Morse theory (Milnor 1963). Basically we have identified the slices that contain critical points and portions of obstacle boundaries between the critical points to form cells. In Figure 2, we show a cellular decomposition and its Reeb graph (Reeb 1946) representation that describes the topology of the cellular decomposition. The Reeb graph represents the critical points as nodes and cells as edges unlike the conventional adjacency graph that has cells as nodes and neighboring information represented as edges. We use the Reeb graph because we want to encode the topologically meaningful events (critical points) as nodes. Generically, we can characterize each cell by two critical points and represent it with an edge between two nodes.[3] Therefore, "left" and "right" most cell boundaries are defined by one critical point each.

Once we have the cellular decomposition, path generation is simple. In each cell, the robot performs back-and-forth motions. Once the robot covers one cell, it has essentially completed "tracing" an edge of the Reeb graph. Each edge of the graph has only two nodes and each node may have one, two or zero edges connected to it. Each edge can correspond to an uncovered or covered cell. If there is an uncovered cell, i.e., unexplored edge, the robot moves to the cell and starts to cover it. The robot concludes that it has completely covered the space when all the edges of the Reeb graph have been explored.

Figure 2 contains cellular decomposition of a space and depiction of the coverage path. With this coverage path, the robot passes the mine detector over all the points at least once. However, depending on the detector's error rate, the robot can follow the same path more than once to minimize the number of missed mines. Essentially, a demining robot using the coverage method that we have just described incrementally constructs the Reeb graph while covering the space. We explain methods for incremental construction that rely on realistic sensors in Section 3. Our incremental construction methods are proven to be complete and achieve complete coverage in considerably less time than random coverage as explained in the next section.

2.2. Random Coverage

Currently, the state of the art for robotic demining and UXO clearance is to drive the robot randomly in a field. Even though random coverage could be suitable for systems that do not have sophisticated sensing systems, achieving complete coverage of a field is not guaranteed. To establish the trade-offs between random versus complete coverage, we provide the analysis of simulations. In our simulations, we used the specifications supplied by our collaborators at The Naval Explosive Ordnance Disposal Technology Division. We place 100 mines/UXOs uniformly on a 100×100 ft^2 area. We set the width of the detector to 1 ft and the speed of the robot to 1 ft s^{-1}. The robot's rotation speed is fixed at 0.1 rev s^{-1}. Random search uses the following procedure.[4] The robot moves along a straight line until it senses[5] an obstacle in front of it; then it rotates some random amount. When the robot detects a mine/UXO, it picks up the mine/UXO, delivers it to the boundary of the field and drops it. We assume that the robot

2. Even though we present our formulation in the configuration space, the robot never constructs the configuration space ahead of time. It achieves coverage on-line using workspace distance measurements.

3. Note that dealing with non-generic configurations such as two critical points on the same slice or non-isolated critical points is an implementation detail.

4. Note that we have tested various random strategies and found that the results are close to each other.

5. We use the term *sensor* for the sensing system that the robot is using to locate the obstacles and *detector* to refer to the mine/UXO detector.

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

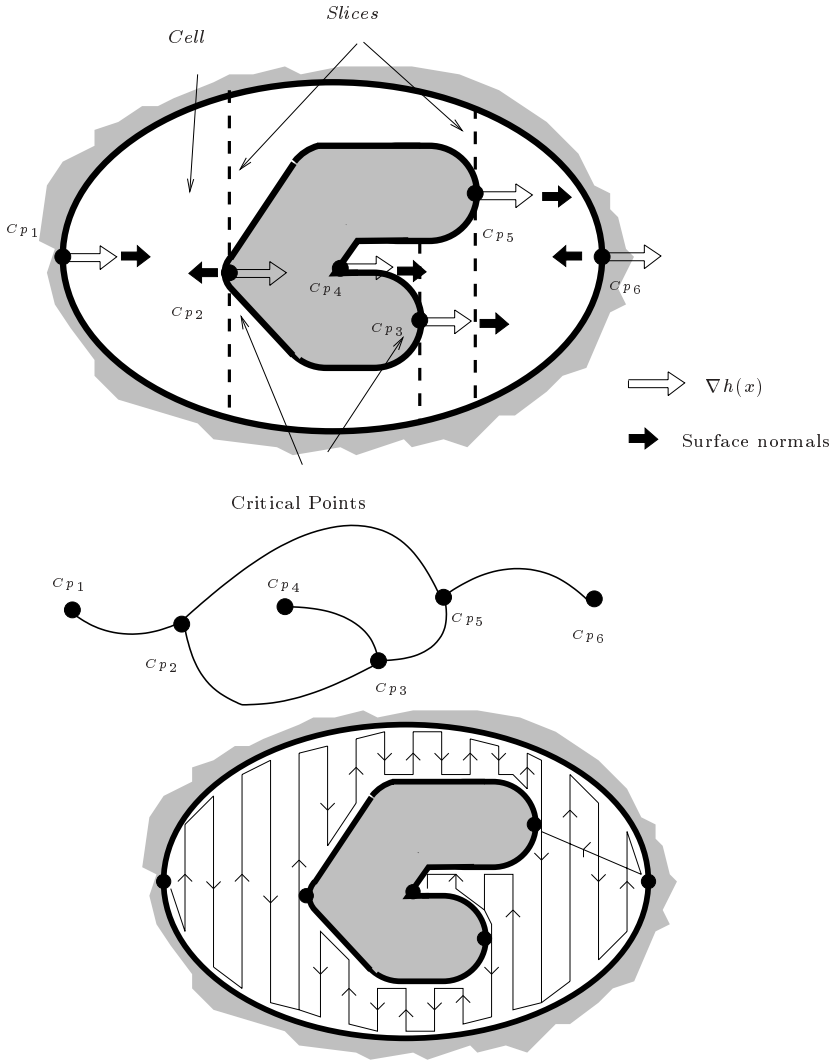


Fig. 2. An example exact cellular decomposition, its Reeb graph and the coverage path. The cell boundaries have two parts: slices that contain critical points and portions of the obstacle boundaries. At the critical points of $h|_{\partial \mathcal{C}_i}$, the gradient of h, $\nabla h(x)$ (white arrows), and the surface normals (black arrows), are parallel. The Reeb graph representation of the cellular decomposition encodes all the information we need to achieve complete coverage. The nodes of the Reeb graph correspond to the critical points and its edges represent the cells. To cover each cell the robot only needs to perform back-and-forth motions. When the robot is done with one cell, it travels back to a critical point that has an associated uncovered cell.

takes two minutes to pick up and drop the mine/UXO, not including the travel time. For complete coverage we use the cellular decomposition and back-and-forth motions described in the previous section.

First, we compare random and complete coverage in an environment without obstacles. Figure 3 shows the plot of the number of found mines/UXOs versus the mean total search time in the random search. Table 1 summarizes what we have found. With a perfect detector (100% detection rate), in order to find 100% of targets, random search takes much longer time (20 h) compared to complete search (5.8 h). Therefore, achieving complete coverage with random motions with a perfect detector is inefficient. Note that with an imperfect detector, the robot cannot, on average, find 100% of the mines/UXOs, even with a complete strategy. Thus, for the imperfect detector case, we compare the recovery of 80% of the mines, suggested by our collaborators. We have found that the difference in mean search time between random and complete strategies becomes smaller as the sensor detection rate becomes worse. However, the variability of search time for random search is

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

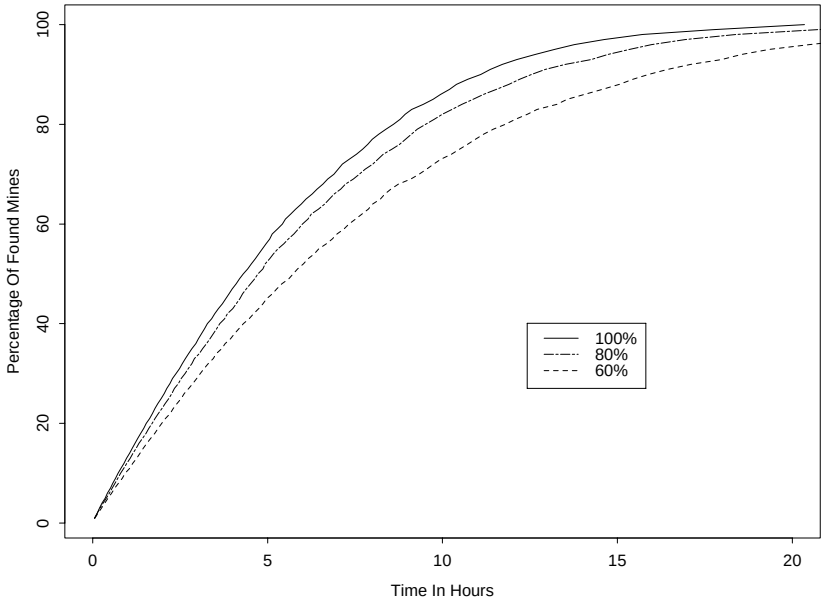


Fig. 3. Plot of the number of found mines versus total mean search time by random search using different detection rates (100%, 80% and 60%). Target area is 100 × 100 ft^2 without any obstacles.

Table 1. The Means and Standard Deviations in Search Time (in Hours) to Detect 80% of Mines for Random and Complete Coverage of a 100 × 100 ft^2 Field Without Any Obstacles

	Detection Rate	60%	80%	100%
80% of targets	Random (Std. Dev.)	11.9 (1.01)	9.55 (0.76)	8.90 (0.59)
	Complete (Std. Dev.)	8.75 (0.26)	6.38 (0.14)	5.80 (0)
	Ratio (mean)	1.36	1.5	1.53

Note that we can never guarantee 100% clearance of the mines/UXOs with an imperfect detector (in mean time).

much larger than the complete strategies. Hence, random coverage still remains inefficient. It is also worth noting that even with a perfect detector, with a randomized strategy, finding the last 20% is harder than finding the first 80%.

Next we examine the effect of placing obstacles in the minefield. We have found that the duration of random coverage increases drastically depending on the location of the obstacles. Meanwhile, coverage duration for complete search does not change dramatically and it remains proportional to the size of the target area. We present two types of obstacle configurations (Figure 4). In the first configuration, we place 25 obstacles evenly in the field. Figure 5 shows the amount of found mines versus mean of the total search time in the random search. Using complete coverage with a perfect detector the robot spends 5 h 27 min to cover the whole space, whereas random coverage takes about 30 h to recover all the mines. In the second obstacle configuration, we place 30% the mines inside the area surrounded by the obstacles. In random search, once the robot moves outside the area enclosed by the obstacles, it has a very small chance to move back to the enclosed area, because the size of the exit is small. In this case, the random search time increases dramatically to more than 50 h, whereas complete coverage takes 6 h 40 min. Figure 6 shows the amount of found mines versus the mean of total search time.

From these simulations, we conclude that efficiency of the random coverage strongly depends on the obstacle configurations. A minefield that is populated with trees is an example

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

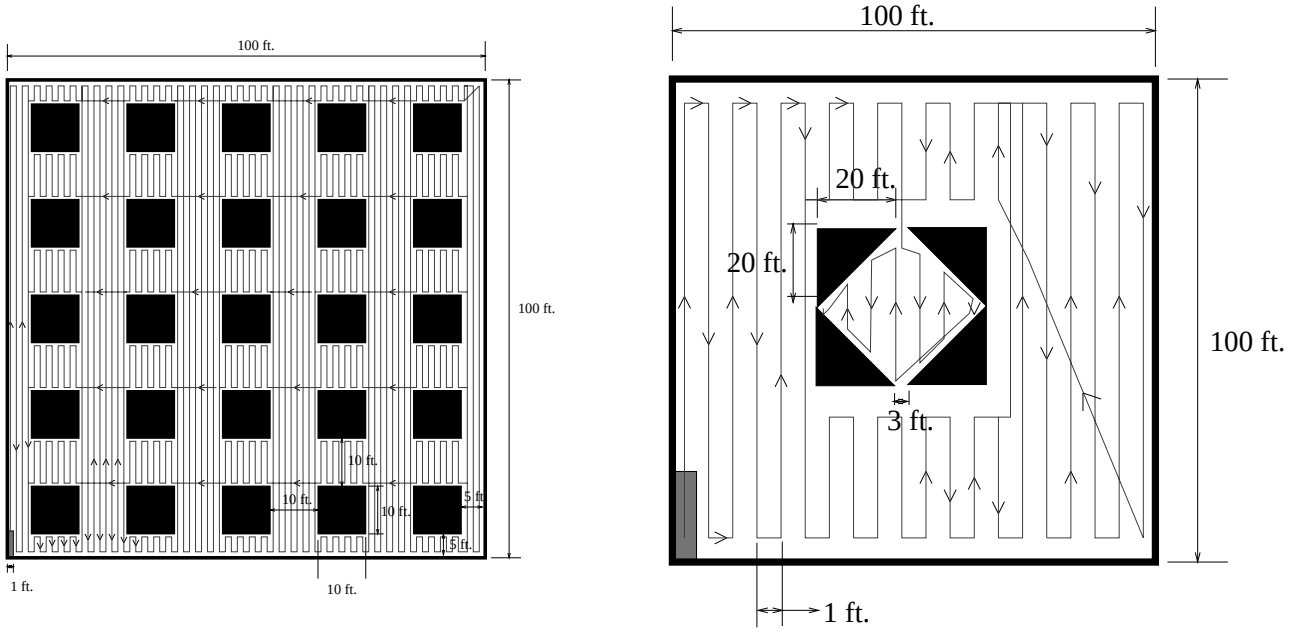


Fig. 4. Complete coverage paths in a 100×100 ft^2 environment with 25 and 4 obstacles.

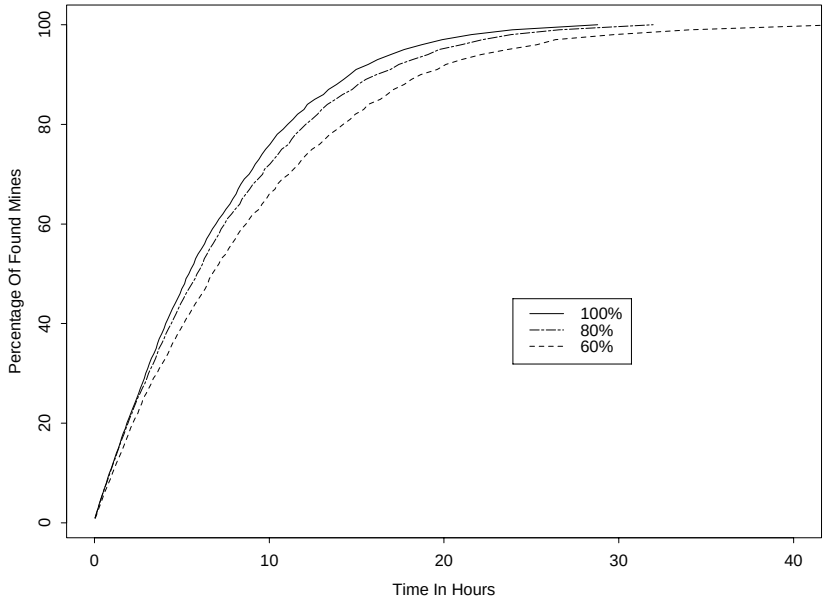


Fig. 5. Plot of the number of found mines versus mean of the total search time by random search using different rate detectors (100×100 ft^2 environment with 25 obstacles).

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

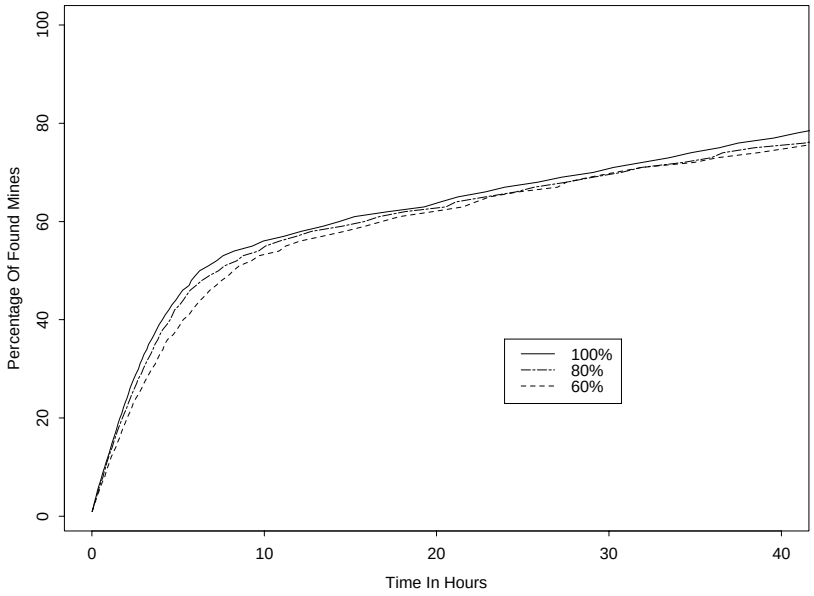


Fig. 6. Plot of the number of found mines versus mean of the total search time by random search using different detection rates (100×100 ft^2 environment with 4 obstacles).

where random coverage will perform poorly. On the other hand, complete coverage is not affected by the obstacle configuration; it will always achieve complete coverage. Even if we put the benefit of completeness aside, random coverage suffers from long coverage duration that requires more power which ironically increases the cost, both in money, weight and complexity of the final system.

3. Incremental Construction in Unstructured Environments

In general, maps of mine/UXO fields are not available. Even if we have a map, it is very likely that it will be incomplete or incorrect. Therefore, a coverage algorithm should be able to cover a priori unknown spaces using its sensors. In other words, the robot needs an algorithm that constructs the cellular decomposition incrementally. Furthermore, the algorithm should be able to handle bad sensor data due to unstructured obstacle boundaries because demining and UXO clearance robots need to work in outdoor fields. In this section, we first review our prior incremental construction algorithm (Acar and Choset 2002) that works in unstructured environments. Then, in this paper, we exploit the topology and geometry encoded in the cellular decomposition to reject bad sensor readings. Our approach differs from conventional approaches in the sense that we do not use uncertainty models or perform low-level data processing. We only use the features that are independent of the amount of the sensor uncertainty. We present our approach with supporting experimental data. Incremental construction has two main parts: critical point sensing and encountering all the critical points, and hence achieving complete coverage. We first present critical point sensing methods.

3.1. Critical Point Sensing Methods

We introduce two methods to sense the critical points: one using range sensors and the other using relative position information. First we present the one that uses range sensors. We model the range sensors using a distance function $d_i(x)$. The value $d_i(x)$ is the shortest distance between a point x and a workspace obstacle $\mathcal{C}_i$ (Figure 7). The distance function and its gradient are

$$d_i(x) = \min_{c \in \mathcal{C}_i} \|x - c\|,$$

$$\nabla d_i(x) = \frac{x - c_0}{\|x - c_0\|} \quad \text{for} \quad c_0 = \text{argmin}_{c \in \mathcal{C}_i} \|x - c\|.$$

We calculate $d_i(x)$ and $\nabla d_i(x)$ using a range sensor, such as a sonar ring, that supplies distance measurements radially distributed around the robot (Figure 7). The distances $d_i(x)$ to the obstacles are determined by calculating the local minima of the range measurements with respect to an angular parameter θ. The magnitude of the local minimum together with its direction θ are used to determine $d_i(x)$ and $\nabla d_i(x)$.

Once the distance measurements and $\nabla h(x)$ (sweep direction) are available, we relate them for critical point sensing. In

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

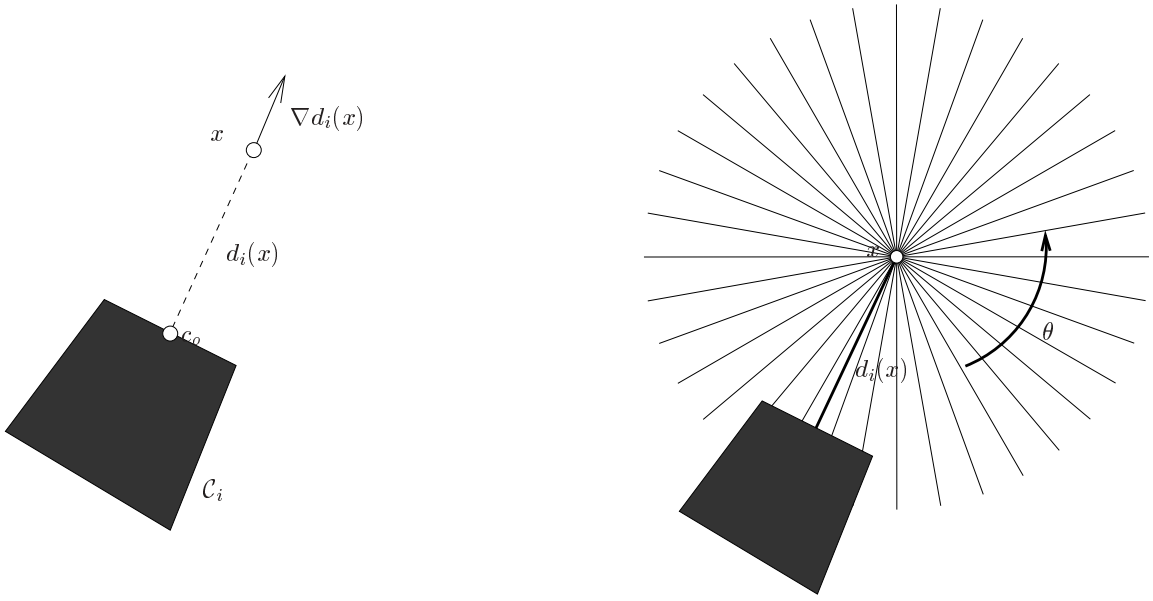


Fig. 7. The distance $d_i(x)$ is defined as the Euclidean distance between x and the closest point c_o on the obstacle $\mathcal{C}_i$. The gradient of $d_i(x)$, $\nabla d_i(x)$, is the unit vector pointing away from c_o. The local minimum of the range measurements that are radially distributed around the robot with respect to θ is used to determine $d_i(x)$ and $\nabla d_i(x)$.

our prior work (Acar and Choset 2002), we have shown that if the robot is located at $x \in \mathcal{CS}_{\lambda_2}$ and $\nabla d_i(x)$ is orthogonal to the slice, i.e., $\nabla d_i(x) \perp \mathcal{CS}_{\lambda_2}$, then there exists a critical point on the boundary of the configuration space obstacle, $\partial\mathcal{CC}_i$, at $(x - \nabla d_i(x) d_i(x)) \in \partial\mathcal{CC}_i$ (Figure 8).

Loosely speaking, the coverage algorithm in Section 3.3 has two types of motions: one where the robot moves along straight lines which we call laps, and one where the robot follows the obstacle boundaries. In large open spaces, most of the coverage occurs on the laps. In all spaces, we direct the robot to follow the boundaries of obstacles at a small safety distance. While following the boundaries of obstacles, the robot detects the critical points. We use the parallel vector result from above to explicitly sense the critical points. Therefore, when the robot encounters a critical point, i.e., $\nabla d_i(x) \perp \nabla h(x)$, the robot registers the critical point's position as the robot's center point.

The second critical point sensing method uses relative position information. We know that, at the critical points, the function $h|_{\partial\mathcal{CC}}$ takes its local extrema. Since we use the function $h|_{\partial\mathcal{CC}}(x) = x_1$, the robot can look for local extrema in the first coordinate of its position to locate the critical points while it is performing boundary following. In Figure 9 we depict two sample critical points Cp_1 and Cp_2 on the boundaries of the obstacles $\mathcal{CC}_1$ and $\mathcal{CC}_2$, respectively. Consider a path on $\partial\mathcal{CC}_1$ that passes through Cp_1 as depicted in Figure 9. Moving along the path towards Cp_1 decreases the value of $h|_{\partial\mathcal{CC}}(x)$. After passing through Cp_1, the value increases. In other words, $h|_{\partial\mathcal{CC}}(Cp_1)$ is a local minimum of $h|_{\partial\mathcal{CC}}$. Likewise, $h|_{\partial\mathcal{CC}}(Cp_2)$ is a local maximum of $h|_{\partial\mathcal{CC}}$. Since $h|_{\partial\mathcal{CC}}(Cp_1)$ and $h|_{\partial\mathcal{CC}}(Cp_2)$ are extrema of $h|_{\partial\mathcal{CC}}$, Cp_1 and Cp_2 are the critical points of $h|_{\partial\mathcal{CC}}$. Therefore, the robot can use relative position information to determine local extrema and hence the locations of the critical points.

These two methods allow us to use two different sensor systems (range and position) to locate the critical points. Using two sets of sensing systems, we can improve the robustness of the coverage operation in the sense that if one sensing system fails we can use the other one. Moreover, we can use both systems to verify each other's outcome.

3.2. Types of Critical Points and How to Sense Them

In the previous section, we have prescribed methods to sense critical points, but we have not characterized them as local minima or maxima. In this section we identify the types of critical points (IN, OUT, START, END[6]) so that this identification can be used to reject bad sensor readings for covering unstructured spaces (Section 3.3). Recall that critical points occur on the boundaries of obstacles. When the obstacle boundary is locally convex near the critical point, we

6. Borrowing terms from computational geometry (O'Rourke 1998).

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

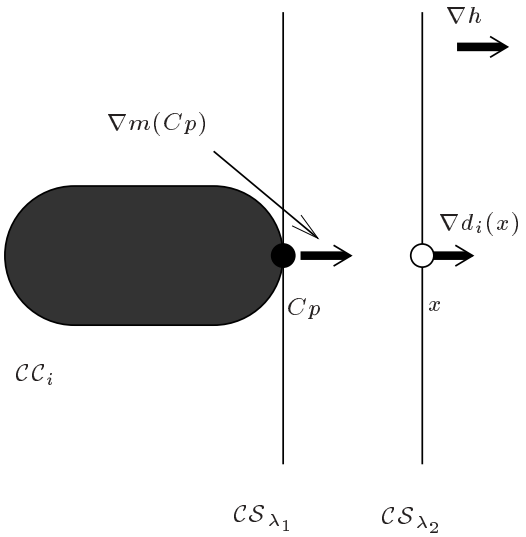


Fig. 8. At point x, $\nabla d_i(x) \perp \mathcal{CS}_{\lambda_2}$. Therefore, there exists a critical point Cp at $x - \nabla d_i(x) d_i(x)$. Note that the critical point is located on the boundary of the configuration space obstacle.

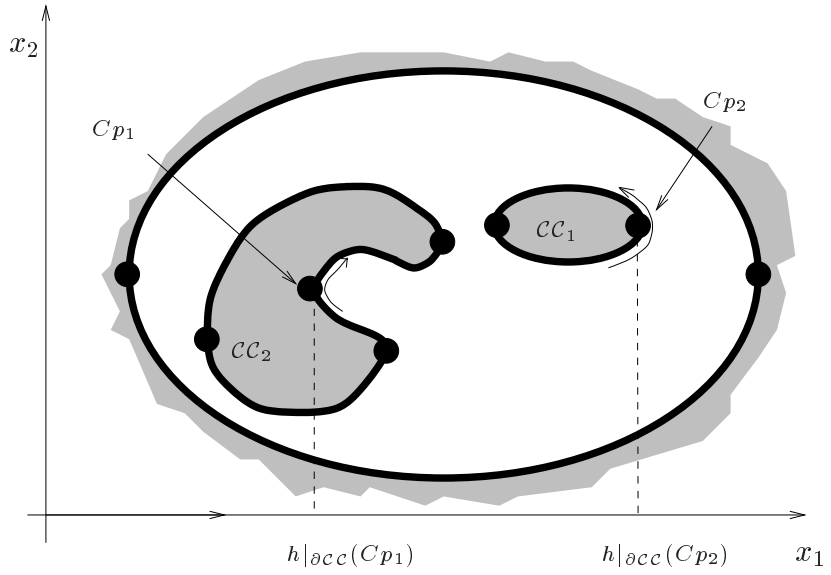


Fig. 9. The restriction of the slice function $h(x) = x_1$ to the obstacle boundaries $h|_{\partial \mathcal{CC}}$ takes a local minimum at Cp_1 and a local maximum at Cp_2. Since $h|_{\partial \mathcal{CC}}$ takes its extrema at Cp_1 and Cp_2, they are the critical points.

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

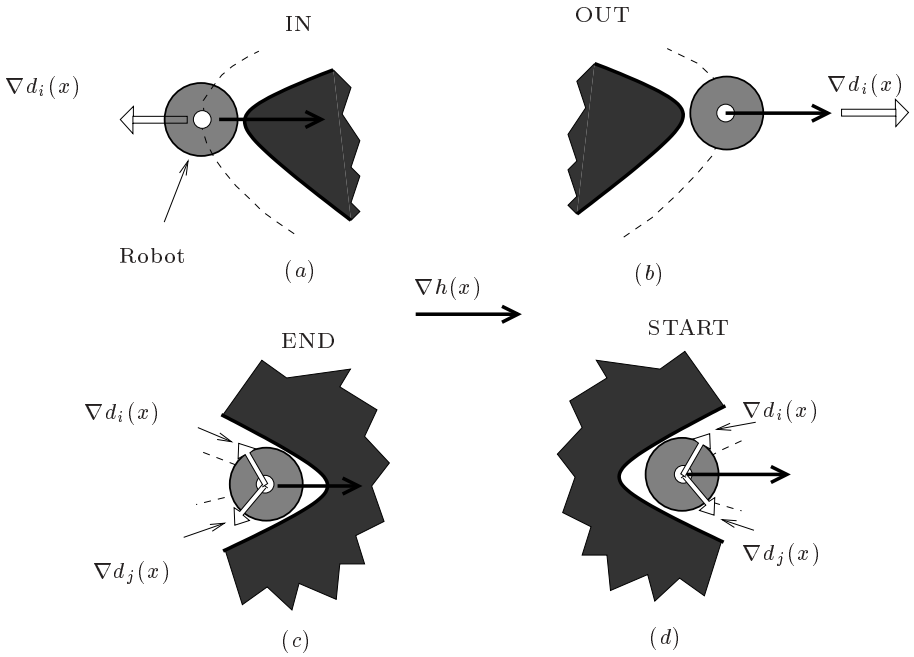


Fig. 10. The robot can use the direction of $\nabla d_i(x)$ with respect to $\nabla h(x)$ to identify four different types of critical points: (a) $-\nabla h(x) = \nabla d_i(x)$, IN; (b) $\nabla h(x) = \nabla d_i(x)$, OUT; (c) $-\nabla h(x) \in CO\{\nabla d_i(x), \nabla d_j(x)\}$, END; (d) $\nabla h(x) \in CO\{\nabla d_i(x), \nabla d_j(x)\}$, START.

have an IN or OUT critical point where $\nabla d_i(x) = -\nabla h(x)$ at IN critical points and $\nabla d_i(x) = \nabla h(x)$ at OUT critical points (Figures 10(a) and (b)). When the obstacle boundary is locally concave, we have a START or an END critical point. Depending on the curvature of the obstacle boundary, the robot needs to use position or range data to sense the START and END critical points. First we describe the case when the curvature of the robot's periphery is greater than the obstacle's. In this case, the robot uses the distance measurements where $d_i(x)$ is non-smooth. In other words, while the robot is following the boundary of an obstacle it constantly measures the distance and its gradient, and when a jump in the gradient of $d_i(x)$ occurs, the robot locates the non-smooth point. At an END critical point $-\nabla h(x) \in CO\{\nabla d_i(x), \nabla d_j(x)\}$ where $CO\{\nabla d_i(x), \nabla d_j(x)\}$ is the convex hull of $\nabla d_i(x)$ and $\nabla d_j(x)$ (Figure 10(c)). The convex hull is the set of vectors from point x to any point on $(\nabla d_i(x) - \nabla d_j(x))$. At a START critical point $\nabla h(x) \in CO\{\nabla d_i(x), \nabla d_j(x)\}$ (Figure 10(d)).

When the boundary's curvature is smaller than the robot's periphery, i.e., $d_i(x)$ is smooth, the robot cannot use the distance measurements to distinguish START and END critical points from IN and OUT critical points (Figure 11). However, using relative position information, the robot can determine whether it has just passed a local minimum or maximum while it is following the boundary of an obstacle (Figure 12). If the robot senses a local minimum, the critical point can be either an IN or a START critical point. If the robot senses a local maximum, the critical point can be either an OUT or END critical point. The robot can distinguish IN from START or OUT from END by observing the relative history of the position information. If the boundary of the obstacle is locally convex (concave) and the robot senses a local minimum, then the critical point's type is IN (START). If the boundary of the obstacle is locally convex (concave) and the robot senses a local maximum, then the critical point's type is OUT (END).

3.3. Encountering All Critical Points and Identifying Features

Now that we know how to sense and determine the types of critical points, in this section we overview our prior provably complete algorithm that allows the robot to simultaneously cover a space and look for the critical points, and identify features for bad sensor rejection. We first present the main part of the coverage algorithm, *cycle algorithm*, which ensures that the robot encounters the critical point that indicates the completion of a cell. Then we identify the algorithm's geometrical and topological features to reject bad sensor readings. We present experimental results that support our findings.

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

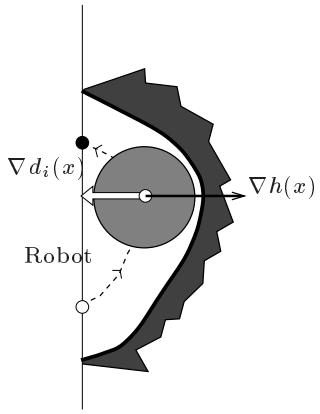


Fig. 11. The curvature of the boundary is smaller than $1/r$ where r is the radius of the robot. In this case the robot can use the distance measurements to sense the critical point but not to determine the type of the critical point. In this situation, the robot uses relative position data to determine the type of the critical point.

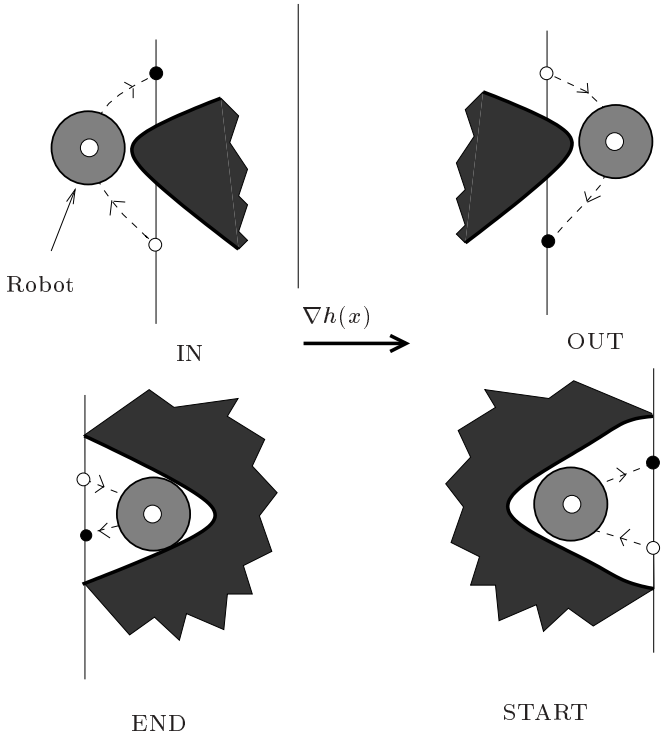


Fig. 12. The robot uses relative position data to determine the existence of local extrema and hence to sense the critical points. IN and START critical points occur when there is a local minimum. OUT and END critical points occur when there is a local maximum.

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

3.3.1. Cycle Algorithm

The robot achieves complete coverage by covering each individual cell. The robot covers a cell by performing lapping and boundary following motions while looking for the critical point that indicates the complete coverage of the current cell. Once the robot encounters this critical point, coverage via back-and-forth motions of the current cell is completed. The robot can now move to an uncovered cell. Therefore, we can reduce the problem of encountering all the critical points in an unknown space to encountering the critical point that indicates the completion of the cell being covered. The cycle algorithm achieves this task incrementally; for the completeness proof of the algorithm, see Acar and Choset (2002). The coverage algorithm basically runs the cycle algorithm repetitively and, as the robot encounters critical points, it updates the graph representation of the cellular decomposition.

We assume that the lap width, i.e., lateral distance between two consecutive laps, is equal to the robot's diameter (detector size). Without loss of generality, the robot starts the cycle algorithm at S_i in slice $\mathcal{CS}_{\lambda_i}$ and lies in the ceiling[7] of the cell being covered. Any lapping path (motion along a straight line) followed in a direction from ceiling towards floor is referred to as forward lapping. We also refer to sweep direction as forward.

From S_i the robot looks for critical points via three phases: forward, reverse and closing. The robot first executes the ***forward phase:*** the robot follows a forward lapping path starting at S_i along $\mathcal{CS}_{\lambda_i}$ towards the floor. Then it follows the boundary in the forward direction. The robot terminates forward boundary following and the forward phase if it laterally moves one lap width (Figure 13(a1)) or encounters a critical point (OUT or END) in the floor (Figure 13(b1)). In Figures 13(a), (b) and (c), the robot follows the dashed path between points S_i and 2 during the forward phase. At the end of the forward phase, the robot executes the ***reverse phase:*** the reverse phase is an interleaved sequence of reverse laps, (towards ceiling), and reverse boundary following paths. The robot may encounter zero (Figure 13(a2)) or one or more critical point(s) while it is performing reverse boundary following motion(s) (Figures 13(b2) and (c2)). A reverse boundary following path initially starts in the reverse direction. Each reverse boundary following operation terminates when the robot either (1) senses an IN critical point[8] (e.g. Figure 13(b2)), or (2) returns back to the slice that contains the starting point S_i (e.g. Figure 13(a2)), and thus completes the reverse phase. In Figures 13(a2), (b2), (b3), (c2) and (c3), the robot follows the solid path between points 2 and 3 during the reverse phase. The robot may reach the starting point S_i at the end of the reverse phase, but, if not, it executes the ***closing phase:*** the robot follows lapping paths along the slice $\mathcal{CS}_{\lambda_i}$ inter-mixed with boundary following paths. Closing boundary following paths initially start in the forward direction, and terminate when the robot returns to $\mathcal{CS}_{\lambda_i}$. The robot encounters at least one critical point during the closing phase. In Figures 13(b3) and (c3), the robot follows the dotted path between points 3 and S_i during the closing phase.

7. We borrowed the terms ceiling and floor from computational geometry literature (O'Rourke 1998). Ceiling refers to the upper boundary of a cell and floor refers to the lower boundary.

8. Note that if the robot senses OUT or START critical points, it continues to follow the boundary.

EXAMPLE 1. An example path generated by the repeated execution of the cycle algorithm for coverage in a hallway is shown in Figure 14. The robot starts to lap at point S_1, encounters an object and performs boundary following. Then it performs *reverse lapping* and at the end of *reverse boundary following* returns back to S_1. The robot completes the *cycle*. Now, the robot must perform the next cycle. So first, it "undoes" the prior reverse boundary following step. Let S_2 be the beginning of the next cycle, but we locate it at the starting point of the last lapping motion. The robot does not need to drive to S_2, because it has already traveled along the previous lap. So it completes the next cycle starting with boundary following, pretending as if it has started at S_2.

3.3.2. Types of Critical Points Encountered in Each Phase

During the extensive testing of the coverage algorithm using a Scout mobile robot (Nomad 1996), which has 16 sonar sensors in unstructured environments, we have observed many failures because of the bad sensor data. To prevent some of these failures, we determine the types of critical points that can occur in each phase of the cycle algorithm.

In the forward phase, the robot can only sense OUT or END types of critical points (Figure 15(a)). In the reverse phase, there are three possibilities: (1) only IN, or (2) only START and OUT, or (3) IN and START and OUT types of critical points can be sensed (Figure 15(b)). Finally in the closing phase, the robot can only sense OUT and/or START types of critical point(s) (Figure 15(c)). See Table 2 for a summary.

A snapshot of an experiment that uses Table 2 to reject bad sensor data is given in Figure 16. The robot senses an IN critical point (convex local minimum) using the distance measurements, even though there is no critical point, while it is executing the forward phase. Since there cannot exist an IN critical point along the forward-phase path (because of the design of the cycle algorithm; see Acar and Choset (2002) for further details), the robot ignores the sensor data that indicates an IN critical point and continues to follow the boundary until it senses an END critical point.

Table 2. Types of Critical Points for Each Phase

	START	OUT	IN	END
Forward		X		X
Reverse	X	X	X	
Closing	X	X		

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

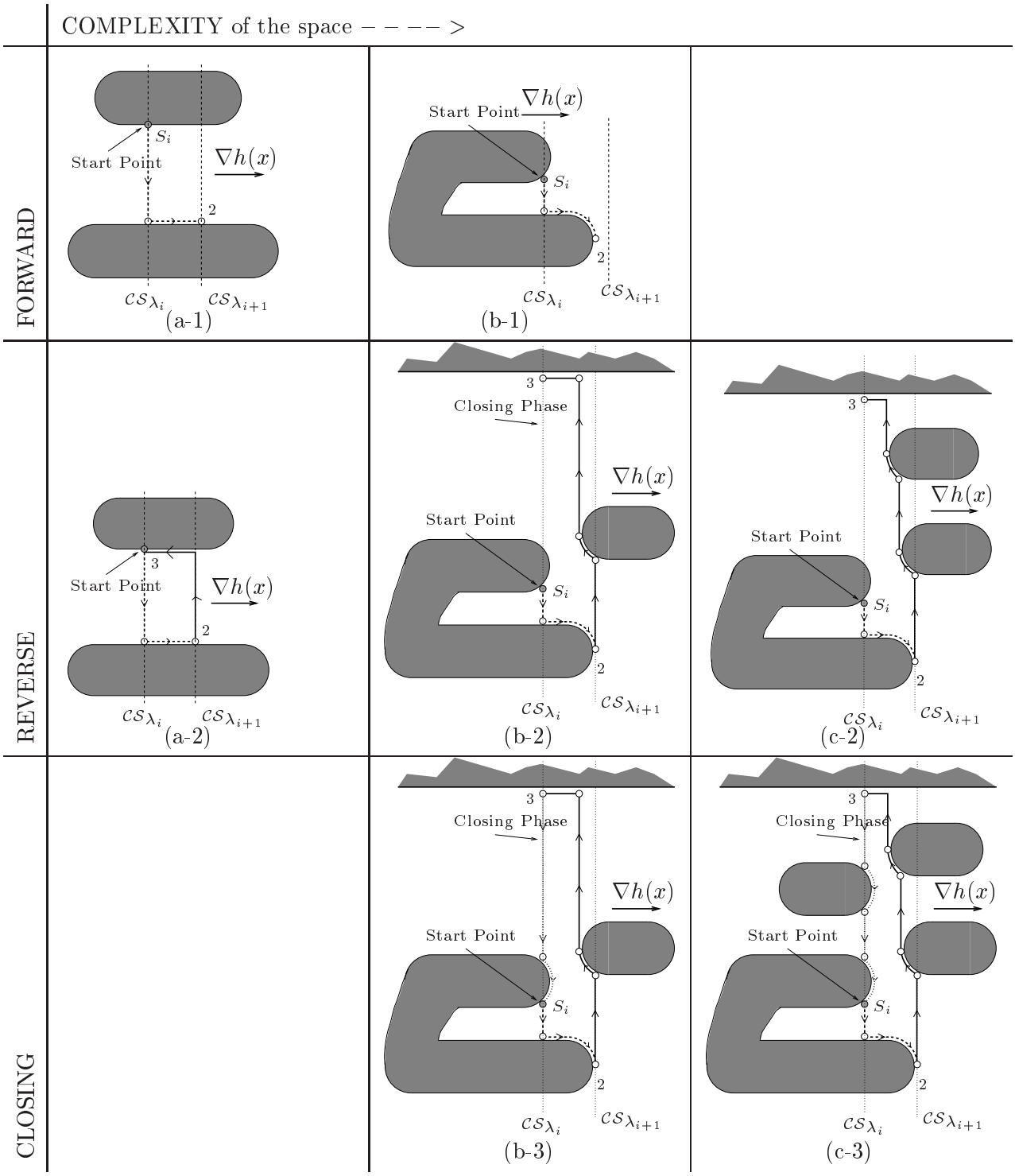


Fig. 13. The robot follows the dashed path during the forward phase, the solid path during the reverse phase and the dotted path during the closing phase. The robot terminates the forward phase when (a1) it reaches the next slice after traveling laterally one robot width or (b1) senses a critical point before reaching the next slice. The robot terminates the reverse phase when it reaches the slice in which it has started the cycle algorithm. During the reverse phase the robot may encounter no critical points (a2) one or more critical point(s) (b2, c2). If the robot needs to execute the closing phase it will encounter one (b3) or more critical points (c3).

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

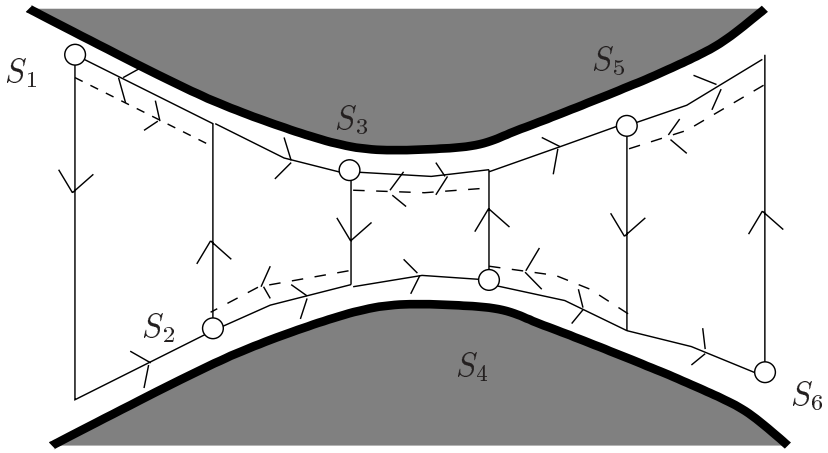


Fig. 14. Path generated by the cycle algorithm in a simple space. S_i refers to starting point of each cycle.

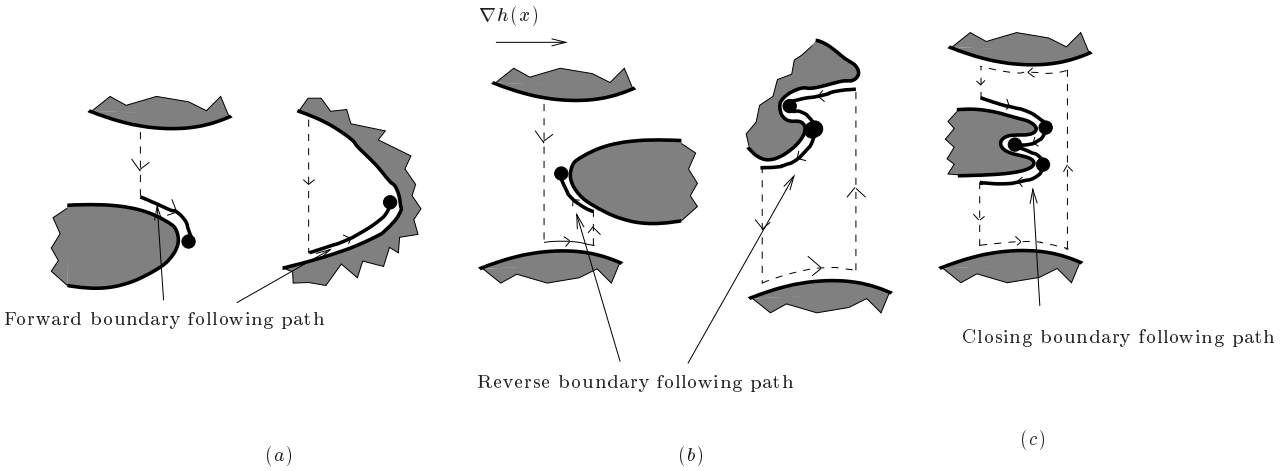


Fig. 15. (a) Only OUT or END types of critical point can exist along a forward-phase path. (b) Only IN and/or START and OUT types of critical points can exist along a reverse-phase path. (c) Only OUT and/or START type of critical point(s) can exist along a closing-phase path.

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

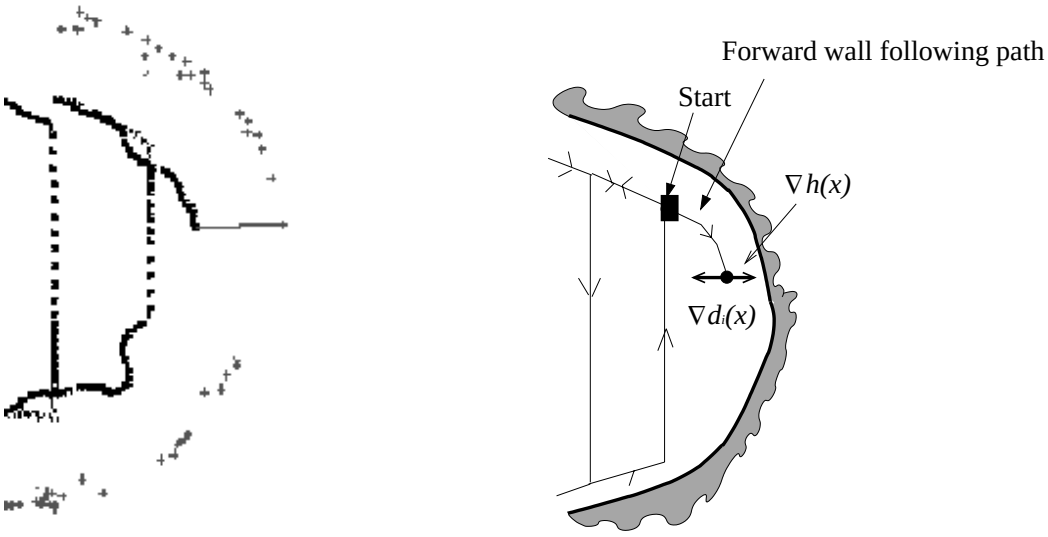


Fig. 16. Experimental data and sketch of the path followed by the robot. Gray dots are the sonar returns and black dots are the path followed by the robot. During forward boundary following, the robot using the noisy sonar measurements calculates $\nabla d_i(x)$ which at some point becomes equal to $-\nabla h(x)$. Since at an IN critical point, $\nabla d_i(x)$ is equal to $-\nabla h(x)$, the robot locates an IN critical point even though there is none. According to Table 2, during forward boundary following the robot cannot sense an IN critical point. Therefore, the robot rejects the sensor data and continues to perform forward boundary following until it senses the END critical point.

3.3.3. Special Note on the Closing Phase

As we have described, the characterization of critical points can be used to prevent some failures due to bad sensor data. However, the procedure outlined in the previous section does not prevent all failures. In this section we use relative position information as it relates the starting and end points of phases of the cycle algorithm to further reject bad sensor data.

As explained in Section 3.3.1, at the end of the reverse phase if the robot does not reach the starting point of the cycle algorithm, the robot executes the closing phase. Under ideal sensing conditions, through a rigorous formulation, we also know that the end point of the closing phase can never be "below" the starting point of the cycle algorithm (Figures 13(c2) and (c3)). However, as seen in Figure 17, towards the end of the closing boundary following motion, the robot receives bad sonar reflections and ends up at a lower position along the slice. Thus, the robot thinks that it has not reached the starting point of the cycle. The robot continues to execute the closing phase without having the opportunity to satisfy the termination condition (reaching the start point), i.e., the robot will run forever. We prevent this failure by augmenting the termination condition. Here we look at the position of the robot along the slice, and if it is below the starting point of the cycle, the robot concludes that it has reached the starting point of the cycle path even in the presence of the sensor noise. Note that during this experiment we observed this failure not because of the accumulation of dead reckoning error but because of bad sensor data. Also note that we cannot solve this problem indefinitely by adjusting the error thresholds. However, the geometric feature of the algorithm that we have used, the relative positions of the phases, is independent of the error thresholds.

3.3.4. Non-Generic Configurations Commonly Encountered

We also use the relative positions of the start and end points of the phases to deal with "non-generic" configurations. Many motion planning algorithms make a "generic position" assumption on the relative placements of obstacles, which does not change the "output" of the algorithm if the obstacles are slightly moved. For example, a slice along which the robot drives cannot be tangent to an obstacle. This is generic in the sense that a slight perturbation of the obstacle will not make the slice become tangent to the obstacle; i.e., if the slice is not tangent to the obstacle, with a slight movement of the obstacle, the slice is still not tangent. We refer to the obstacle configurations as non-generic where a slice that is not tangent to an obstacle becomes tangent (or vice versa) by a slight movement of the slice. In real-life applications we cannot make non-generic configuration assumptions because of sensor noise. Furthermore, in an unstructured space the chances of encountering non-generic configurations increase due to bad sensor readings.

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

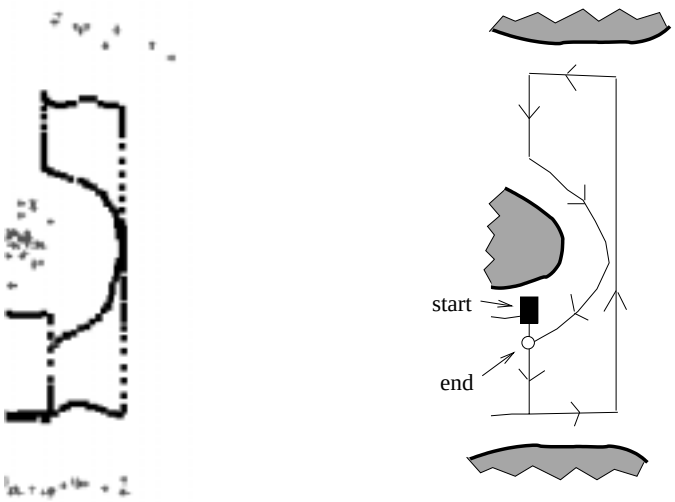


Fig. 17. The end point of the closing phase should always lie above the starting point of the cycle along the slice. The robot ends up below the starting point at the end of the closing phase because of the sensor noise. The robot rejects the sensor data and concludes that it has completed the cycle.

In this section we analyze two non-generic configurations that are caused by the small lap-width and bad sensor data. With small lap-widths (determined by the detector range), the robot has to perform laps close to each other. This increases the chances of encountering configurations where the robot gets very close to a critical point while it is lapping. As depicted in a snapshot of an experiment (Figure 18), the robot just passes by a critical point, i.e., there is enough space for the robot to pass by the obstacle while it is lapping in the forward direction. Now the robot has to sense the "missed" critical point while it is performing reverse boundary following motion. However, because of the sensor noise, the robot reaches the forward lapping path and terminates the reverse boundary following motion (because it has traveled laterally one lap-width) without sensing the critical point. This will lead to incomplete coverage.

To deal with this problem, we determine the possible locations of the end point of the reverse phase. If there is no sensor noise, the end point should be always "above" the starting point of the cycle. In the situation we have described, this is not the case (Figure 18(a)). Therefore, at the end of the reverse phase, the robot notes that it is located "below" the starting point. Since this is only possible because of the bad sensor data, the robot continues to perform the reverse boundary following motion until it senses the critical point (Figure 18(b)). When the robot senses the critical point, it terminates the reverse phase and the cycle path (no need to execute the closing phase). Then the robot chooses an uncovered cell (if any left) and starts to cover it.

Another non-generic configuration commonly encountered is shown in Figure 19. In this case, the critical point lies very close to the end point of a boundary following motion. The robot first executes a forward boundary following motion. This motion ends right before the robot senses the END critical point, which is very close to the end point of the forward boundary following path. After performing reverse lapping motion, the robot starts to perform a reverse boundary following motion and, when it is done, the robot starts to "undo" the reverse boundary following motion by following the boundary in the forward direction. If there is no sensor noise, normally we expect the robot to reach the slice-1 (Figure 19). However, because of bad sensor data, the robot cannot reach the slice-1 and continue to follow the boundary. Since the robot continues to perform undo reverse boundary following motion, it misses the END critical point which causes the robot to perform boundary following motion indefinitely. We have solved this problem by making the robot look for the END critical point while it is undoing the reverse boundary following motion using range and position data.

3.4. Experiments in Unstructured Indoor Environments

In the previous sections, we have shown snapshots of experimental data collected while the robot was performing coverage in unstructured rooms. In this section, we show full successful coverage experiments. We use the distance measurements made by sonars to find the closest point on the closest object to the robot. In other words, we calculate the

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

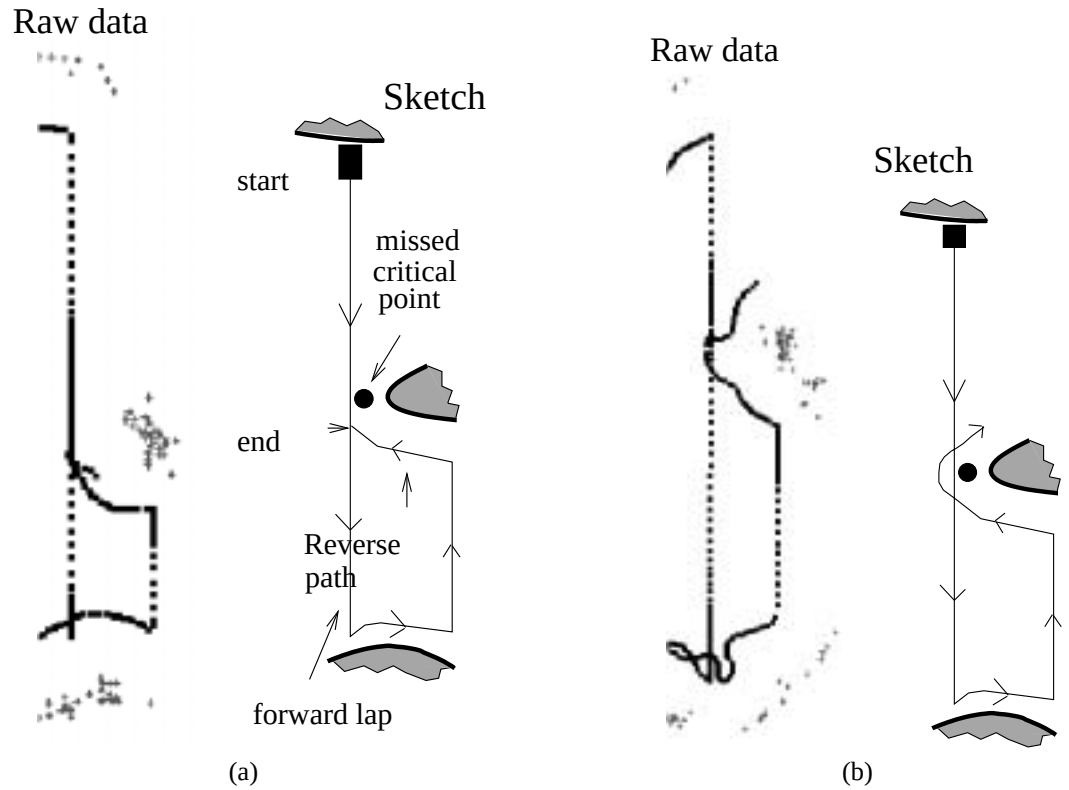


Fig. 18. A commonly encountered non-generic configuration. (a) The robot just passes by a critical point while it is lapping. The robot has to sense the critical point during reverse boundary following motion. However, the robot reaches the slice before sensing the critical point. (b) We solve this problem by making the robot continue to perform reverse boundary following motion until the robot senses a critical point.

global minimum of the distance measurements and its direction. The global position and orientation of the robot (x, y, θ) are determined via dead reckoning using the wheel encoders.

We have used an inter-lap spacing that is equal to the robot's diameter (0.40 m). Since the test environments were not known a priori, we picked an arbitrary slicing direction for each one. Note that, to achieve complete coverage, we only need to store the locations of the critical points and the graph representation, but not the sensed locations of the obstacle boundaries and the path followed by the robot.

Figure 20 shows different stages of a coverage experiment in a 2.5×3.1 m^2 room with a stool (0.40 m diameter and 0.75 m height) in the middle. The dotted black lines represent the path traced by the center point of the robot. The vertical lines are the lapping portions of the path and the jagged-curved lines represent boundary following. Note how the boundary following path resembles the configuration space obstacle for the mobile robot. This makes sense because we are taking the center point of the circular robot as a reference point and we are finding the critical points in the configuration space using work space distance measurements. Unlike in prior work (Acar and Choset 2000), the robot determined the locations of the critical points more robustly and precisely using both range and relative position data as described in Section 3.1. In Figure 20(a), notice how the robot followed the boundary of the obstacle in the vicinity of the critical point 2 to sense the local minimum and hence the critical point.

In Figure 21, we show the coverage path in a more complicated 4×4.6 m^2 room with a table in the middle. In this experiment, we observed the failures due to bad sonar data and recoveries from them. Around critical point A, the robot encounters a non-generic configuration (Figure 22(a)). Since the end point of the reverse boundary following path is below the start point of the cycle path, as we described in Section 3.3.4, the robot continues to follow the boundary until it senses the critical point. Around point B, the robot receives sonar data, which indicate that the surface normal of the obstacle and the sweep direction are parallel while it is performing reverse boundary following (Figure 22(b)). This is the condition for an IN critical point. However, the robot does not sense a

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

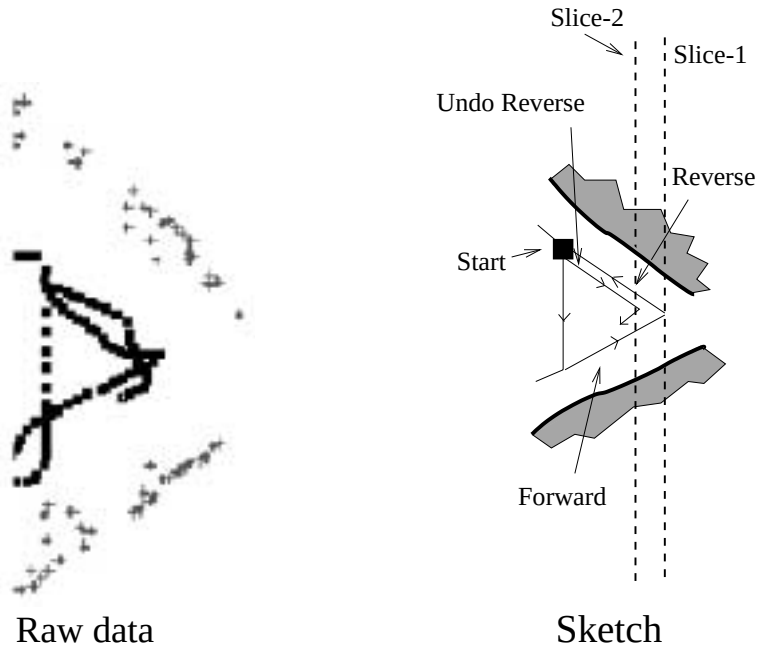


Fig. 19. Non-generic position. The critical point is located on the lapping path. The robot could not reach the slice at the end of the undo reverse boundary following motion.

local minimum to verify the existence of an IN critical point. Therefore, the robot rejects the bad sonar data and finishes the reverse boundary following motion.

We performed an experiment in a 2×2 m^2 living room of an apartment, which has a coffee table in the middle (Figure 23). We observed two failures and recoveries from them. Around point *A* (Figure 24(a)), the robot senses an IN critical point using range data during the forward following path. However, we know that along a forward boundary following path, there cannot exist an IN critical point (Section 3.3.2). Therefore, the robot rejects the sonar data and continues to follow the boundary of the obstacle until it senses the END critical point. Around point *B* (Figure 24(b)), the robot senses an IN critical point using range data along the reverse following path. However, the robot cannot verify the existence of an IN critical point by sensing a local minimum. Therefore, it rejects the IN critical point and continues to perform reverse boundary following motion. In this experiment, we observed the effect of the dead-reckoning error. The robot's perception of the left corner of the coffee table was rotated and shifted. Even though in this experiment the dead-reckoning error did not cause a failure, in larger spaces we need to develop localization methods for coverage. In outdoor environments, global positioning systems with dead reckoning can be used to obtain the position information of the robot.

4. Probabilistic Demining

Exhaustive coverage is the best strategy when the robot has unlimited time (less than what is required by a random strategy) and no a priori information. However, in many situations, time or power limitations may not permit covering a target environment completely. Moreover there may exist a priori information about the minefield. Probabilistic planning exploits a priori information about the distribution of the mines to prevent exhaustive coverage. A probabilistic search in an unknown environment raises two major questions: (1) how to construct a mine distribution map efficiently; (2) given a mine distribution map, how to generate the optimal search path to guide a robot to locate the mines. Gelenbe and Cao (1998) studied the second question. In this section, we give an overview of our efforts to solve the first question.

When mines are deployed by ground vehicles or humans, minefields have a regular pattern because of the military doctrine, tactical efficiency and inherent limitations in the mine-laying process (Lake and Keenan 1995). Typical characteristics of regular patterns are collinearity and equal spacing. Extracting characteristics of a mine-laying pattern helps to quickly build a distribution map.

As a start-up problem, we focus on a particular grid pattern with every other row shifted with respect to the neighboring

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

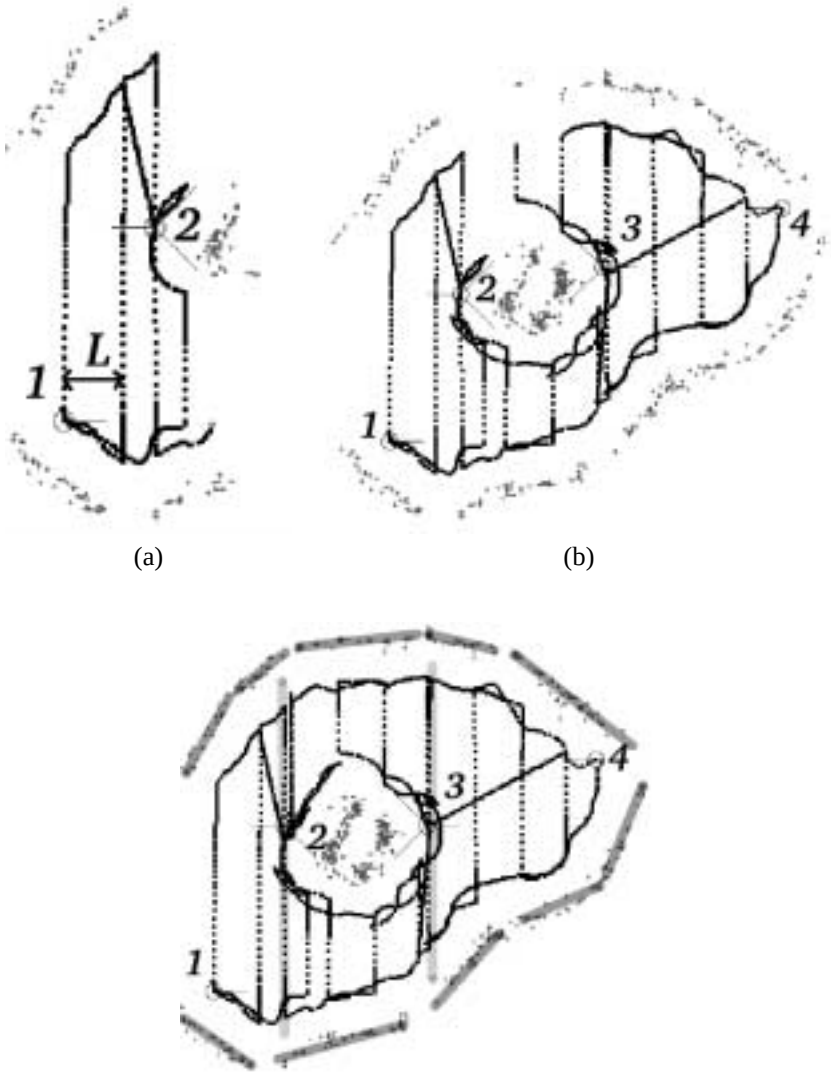


Fig. 20. Three stages of the coverage in an unknown environment. The robot incrementally constructs the graph representation by sensing the critical points 1, 2, 3, 4, 2 (in the order of appearance) while covering the space. In the final stage (c), since all the cells that are associated with the critical points are covered, the robot concludes that it has completely covered the space. For the sake of discussion, we have outlined the boundaries of the obstacles and cells in (c). $L = 0.40$ m.

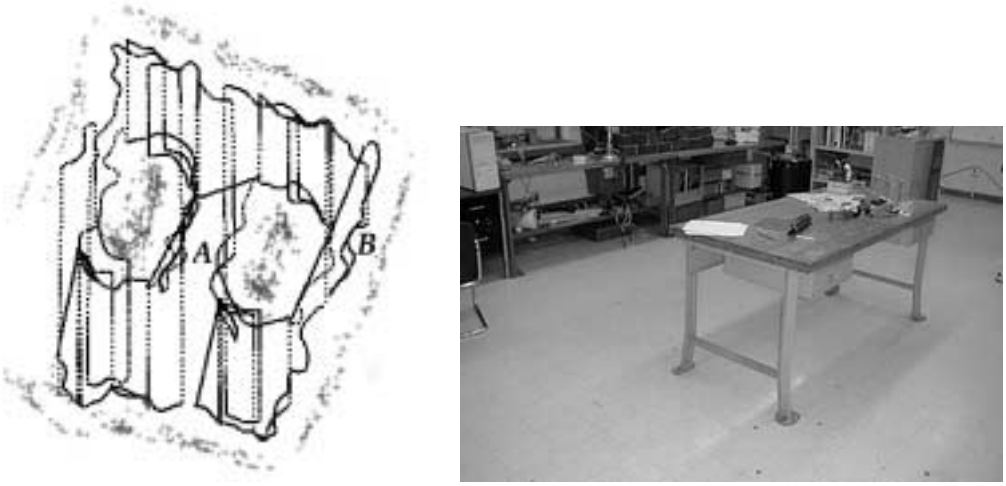
rows as shown in Figure 25. The pattern is characterized by a six-parameter vector $\mathbf{\Delta} = (C_1, C_2, \nu, \theta, X_{00}, Y_{00})$ where C_1 is the inter-row spacing, C_2 is the inter-column spacing, ν is the amount of row shift and (θ, X_{00}, Y_{00}) determines the orientation and position of the minefield with respect to a fixed frame. We develop a probabilistic method to build a distribution map by extracting the "true" parameters of this pattern by covering a small area of the minefield.

The following are the constraints that we consider.

- The method should be able to deal with uncertain information. The mine detector can produce false negative or false positive errors. Moreover, the model of the minefield pattern is generally inaccurate. Possible reasons for deviations of the model from the real world are deployment errors, mine explosions. Therefore, the actual minefield can look quite different than the intended minefield (Figure 25).
- The method should be computationally efficient. Also, it should not require complete coverage of the entire minefield.

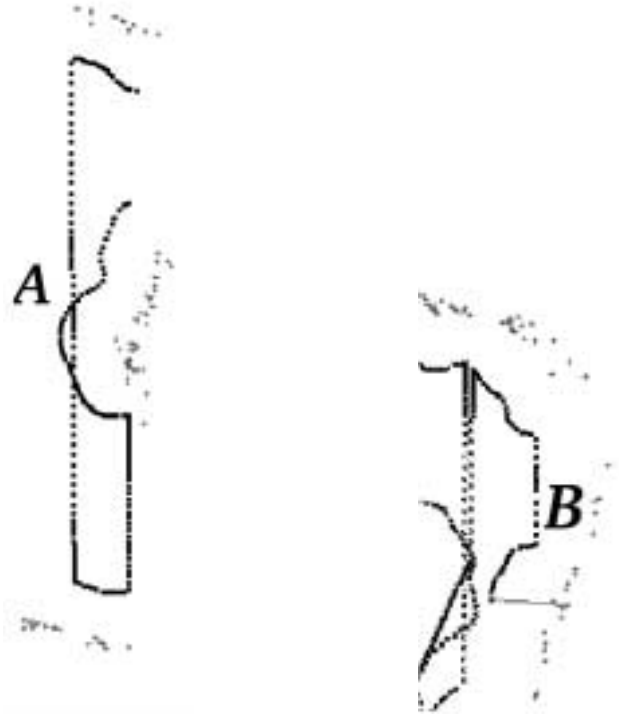
In the following, we will concentrate on describing our method for the pattern parameter estimation. The robot first covers a small portion A of the minefield. The observed infor-

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.



(a) (b)

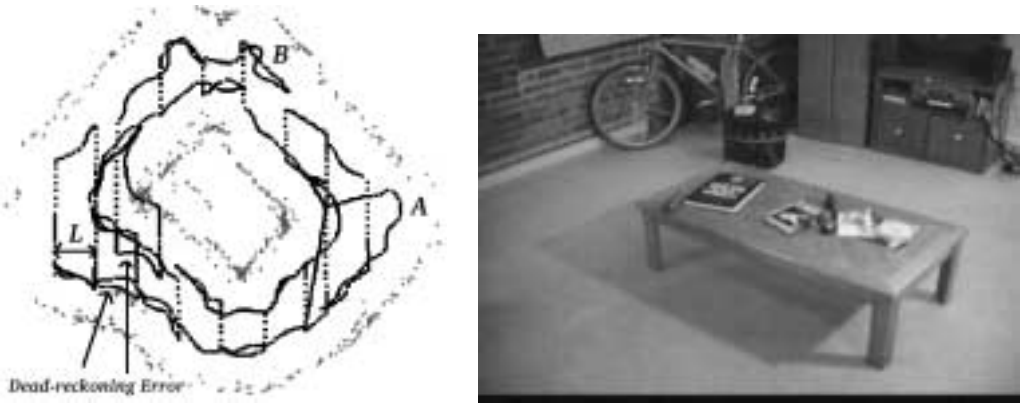
Fig. 21. Coverage path followed by the robot and the sonar returns in a 4×4.6 m^2 room with a table in the middle. The robot successfully covers the room. Note that the robot also covers the area under the table.



(a) (b)

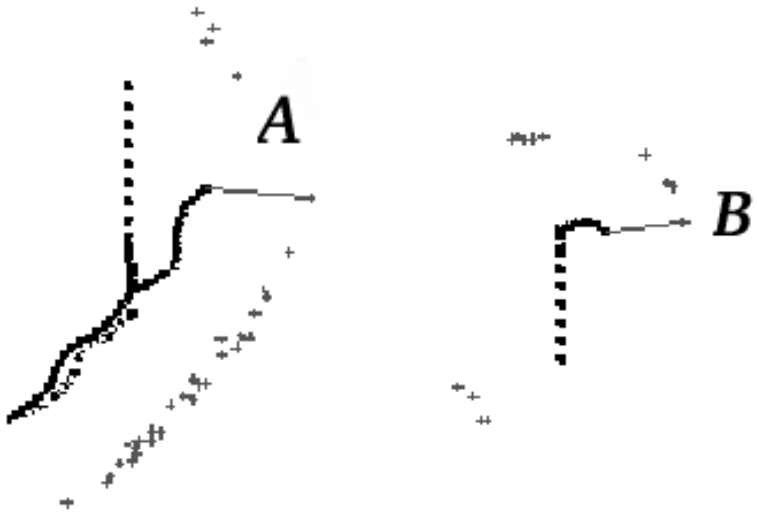
Fig. 22. Blown up shots of the data shown in Figure 21. (a) A non-generic configuration. The robot continues to follow the boundary until it senses the critical point. (b) The robot cannot verify the IN critical point indicated by the sonar data using the relative position data. Therefore, it rejects the sonar data.

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.



(a) (b)

Fig. 23. Coverage path followed by the robot and the sonar returns in a 2×2 m^2 room with a coffee table in the middle. The robot successfully covers the room. The dead-reckoning error is observable.

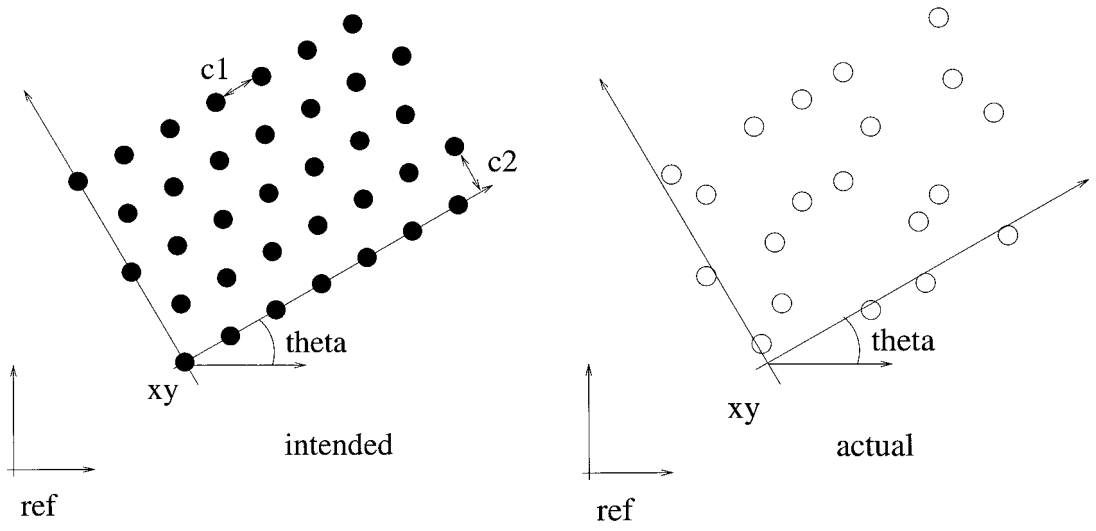


(a) (b)

Fig. 24. Blown up shots of the data shown in Figure 23. (a) Bad sonar data indicate an IN critical point along a forward-phase path. However, there cannot exist an IN critical point in the forward phase, therefore the robot rejects the data. (b) Bad sonar data indicate an IN critical point, but the robot cannot verify it using the relative position data. Therefore, the robot rejects the data.

mation is a set of detected mines at positions $\mathbf{y} = (\mathbf{y}_1, ..., \mathbf{y}_k)$, where $\mathbf{y}_i \in A$, $i = 1, ..., k$. Then, a Bayesian approach is used to solve the pattern parameter estimation problem using the information collected in the covered region A. The Bayesian approach calculates the density of posterior distribution $f_{\mathbf{\Delta}|\mathbf{Y}}(\boldsymbol{\delta}|\mathbf{y}) = \frac{f_{\mathbf{Y}|\mathbf{\Delta}}(\mathbf{y}|\boldsymbol{\delta}) f_{\mathbf{\Delta}}(\boldsymbol{\delta})}{f_{\mathbf{Y}}(\mathbf{y})}$, after observing the locations of some mines at $\mathbf{y}$ inside the covered region A, where we define $\mathbf{\Delta}$ and $\mathbf{Y}$ as random vectors and $\boldsymbol{\delta}$ and $\mathbf{y}$ as their realizations. Meanwhile, the observed mines $\mathbf{y}$ depend on the true minefield pattern parameters $\boldsymbol{\delta}$ through a known conditional probability density $f_{\mathbf{Y}|\mathbf{\Delta}}(\mathbf{y}|\boldsymbol{\delta})$, which is called the likelihood function. Loosely speaking, the likelihood function quantifies the match between the observed mines $\mathbf{y}$ and intended minefield characterized by $\boldsymbol{\delta}$ when certain detector and deployment errors apply. The density of the prior distribution is given by $f_{\Delta}(\boldsymbol{\delta})$. The prior distribution is specified based on the prior intelligence. In the equation given above, $f_{\mathbf{Y}}(\mathbf{y}) = \int f_{\mathbf{Y}|\mathbf{\Delta}}(\mathbf{y}|\boldsymbol{\delta}) f_{\mathbf{\Delta}}(\boldsymbol{\delta}) d\boldsymbol{\delta}$ is the density of the marginal distribution of $\mathbf{Y}$. The posterior distribution is often impossible to compute in closed form, and, even if it were possible, the density $f_{\mathbf{\Delta}|\mathbf{Y}}$ is typically im-

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.



(a) (b)

Fig. 25. The intended and actual minefields. The detector error, deployment error or mine explosion can cause the discrepancy between the intended and actual minefields. The pattern is characterized by six parameters, $\mathbf{\Delta} = (C_1, C_2, \nu, \theta, X_{00}, Y_{00})$, where C_1 is the inter-row spacing, C_2 is the column spacing, ν is the amount of row shift and (θ, X_{00}, Y_{00}) determines the orientation and position of the minefield with respect to the reference frame.

possible to recognize as anything familiar. Instead of trying to calculate the density, we will use a Markov chain Monte Carlo (MCMC) algorithm (Tierney 1994) to create a sample from the posterior distribution of the parameters.

4.1. Random Error Models

The likelihood function can be specified based on specific detector and deployment error models. Here we describe the random error models. See Zhang et al. (2001) for a detailed description of the likelihood function derivation.

The random errors in locations of detected mines are introduced during the mine-laying (deployment) process; the actual laid position is different from the position in which the mine is intended to be laid. In our initial research, a Gaussian white noise model is used to model the deployment error. The detected mine position, (x, y), is bivariate normally distributed with mean (μ_x, μ_y) and fixed variance-covariance matrix $\sigma^2 I(1, 1)$. The density function is

$$f((x, y)|(\mu_x, \mu_y)) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-\mu_x)^2+(y-\mu_y)^2}{2\sigma^2}}, \qquad (1)$$

where (μ_x, μ_y) is the intended mine position, which is the function of the index (I, J) and the pattern parameters $\mathbf{\Delta}$. The chance that the detected mine is located in any region decreases as the distance between the intended and detected locations increases. With a 99.9% chance, each mine will fall within a circle centered at the intended mine position with radius 3.2σ.

Typical mine detectors produce false negatives; the detector can be near a mine, but not sense it. False positives are false alarms; the detector says there is a mine where there is none. Although false positives may delay demining, false negatives are catastrophic. For the sake of simplicity, the initial research assumes that the robot does not produce false positives, but false negatives. We use a simple detection model where the probability of detecting a mine given that it is covered region is

$$P(\text{detected}|\text{mine} \in \text{covered region}) = p_d.$$

Another possible reason that an intended mine is not detected is that the mine was accidentally not deployed or the mine exploded before the demining. Therefore, we modeled the probability of detection and mine absence together in p_d.

4.2. Mode Partitioning Algorithm

It is practically impossible to apply any general MCMC algorithm directly. The algorithm is very inefficient because of the specialty of the likelihood function. First, the likelihood function has many local maxima. Secondly, the likelihood function could peak very quickly around the local maxima and the function is relatively flat when it is away from the maxima. This means that the MCMC algorithm could either easily be trapped in the maxima or accept proposed parameters with near zero probability.

We introduce a mode partitioning algorithm to solve these two problems. This algorithm divides the parameter space

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

into subspaces. In each subspace, only one local maximum exists, and the subspace only includes the area where the likelihood value is not too close to zero. Then, we can direct the MCMC algorithm from one subspace to another one. Next, we describe the details of the mode partitioning algorithm.

We use (I, J) to index the mines on the grid, where (I, J) represents the mine in the Ith column and Jth row with respect to the $(0, 0)$ mine. Let the label of the first found mine be $(0, 0)$. Then, the possible (I, J) labels for the second found mine, the mine found closest to the first mine, can be calculated in the following way. We assume that the intended locations of $(0, 0)$ and (I, J) mines can lie anywhere within circles of a distance 3.2σ centered at the found mine locations. Also, we assume that the mines could not be closer together than some fixed distance:

$$C_1 > C_{1min} \qquad C_2 > C_{2min}.$$

The possible minimum distance n between the intended locations of $(0, 0)$ and (I, J) mines for a given ν can be calculated as:

$$n = \begin{cases} \sqrt{(ic_{1min})^2 + (jc_{2min})^2} & \text{if} \quad j \text{ is even.} \\ \sqrt{((i+\nu)c_{1min})^2 + (jc_{2min})^2} & \text{if} \quad j \text{ is odd.} \end{cases}$$

The valid (I, J) labels for the second mine should satisfy the following equation for some ν

$$n < l + 2r,$$

where l is the distance between the first and second found mines. Based on the intended mine locations and the minimum distance restrictions, the boundary of the parameter subspace when the second mine is labeled as a valid (I, J) can be calculated sequentially. The following is an outline of our approach.

- Given a valid (I, J) pair, find a bounded interval $[\nu_{min}((I, J)), \nu_{max}((I, J))]$, of valid ν values.
- Given (I, J) and ν, find a bounded interval $[\theta_{min}((I, J), \nu), \theta_{max}((I, J), \nu)]$ of valid θ values.
- Given (I, J), ν and θ, we find a region of valid $(x_{(0,0)}, y_{(0,0)})$ pairs.
- Finally, given (I, J), ν, θ and $(x_{(0,0)}, y_{(0,0)})$, we calculate bounds for c_1 and c_2 separately.

4.3. Simulation Results

In Figures 26 and 27, we show snapshots of simulations using our MCMC algorithm. We first generate an intended minefield with known parameters. Then we perturb randomly the intended minefield using our random error models to lay out the actual detected mines. We assume that the robot covers a small portion of the minefield and detects some number of mines (ten mines in Figure 26 and five mines in Figure 27). The mode partitioning algorithm directs the MCMC algorithm to sample the posterior density of the six minefield parameters within each parameter subspace and between the different subspaces. Our simulation experiments show that our algorithm can decode the "true" minefield efficiently in most of the situations. However, when the amount of detected mines are small (about five), the decoding algorithm may lead to the wrong intended minefield.

5. Conclusions

Mine/UXO clearance is an important problem where robotic systems can bring efficiency and safety to the process. In general, a priori information about a mine/UXO field is either not available at all or it is very limited. Furthermore, mine/UXO fields are unstructured spaces where bad sensor readings can easily occur. In this paper, we have focused on the path planning aspects of the mine/UXO clearance problem. We have presented two approaches, complete and probabilistic. Complete coverage was based on exact cellular decomposition in terms of critical points. The robot executing the complete coverage algorithm incrementally constructs the cellular decomposition while it is covering the space with back-and-forth motions. Our algorithm is proven to be complete and does not require any a priori information about the mine/UXO field. Our contribution here is that we show that complete coverage outperforms random coverage which has been considered state of the art for mine/UXO clearance. Even though the general belief is that complete strategies require high-cost robot systems, our complete sensor-based algorithm requires a sensor suit that can guide the robot along the boundaries of the obstacles. In the future we are planning to implement our algorithm on robots that have infrared rings, tactile sensors, etc., so that the cost of the robot can be reduced.

Robotic mine/UXO clearance requires a coverage algorithm that works in unstructured environments since mine/UXO fields are populated with range sensor unfriendly obstacles. In this paper, we have introduced methods and identified features of our algorithm that can be used to overcome failures due to bad sensor data. Currently, our approach is limited by the available features and the requirement that the observations made for the first time are correct. In the future, systematic frameworks that deal with these limitations, and extend the ideas that we use to reject bad sensor readings to other path planning algorithms, should be developed. Also, to decrease the sensor uncertainty, data processing algorithms, such as the representation of obstacle boundaries as straight lines for boundary smoothing, should be explored. We would like to point out that these algorithms will be limited by the angular resolution of the ranging system. Therefore, algorithms that increase the angular resolution of the range data while the robot is following the boundaries of the obstacles should be

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

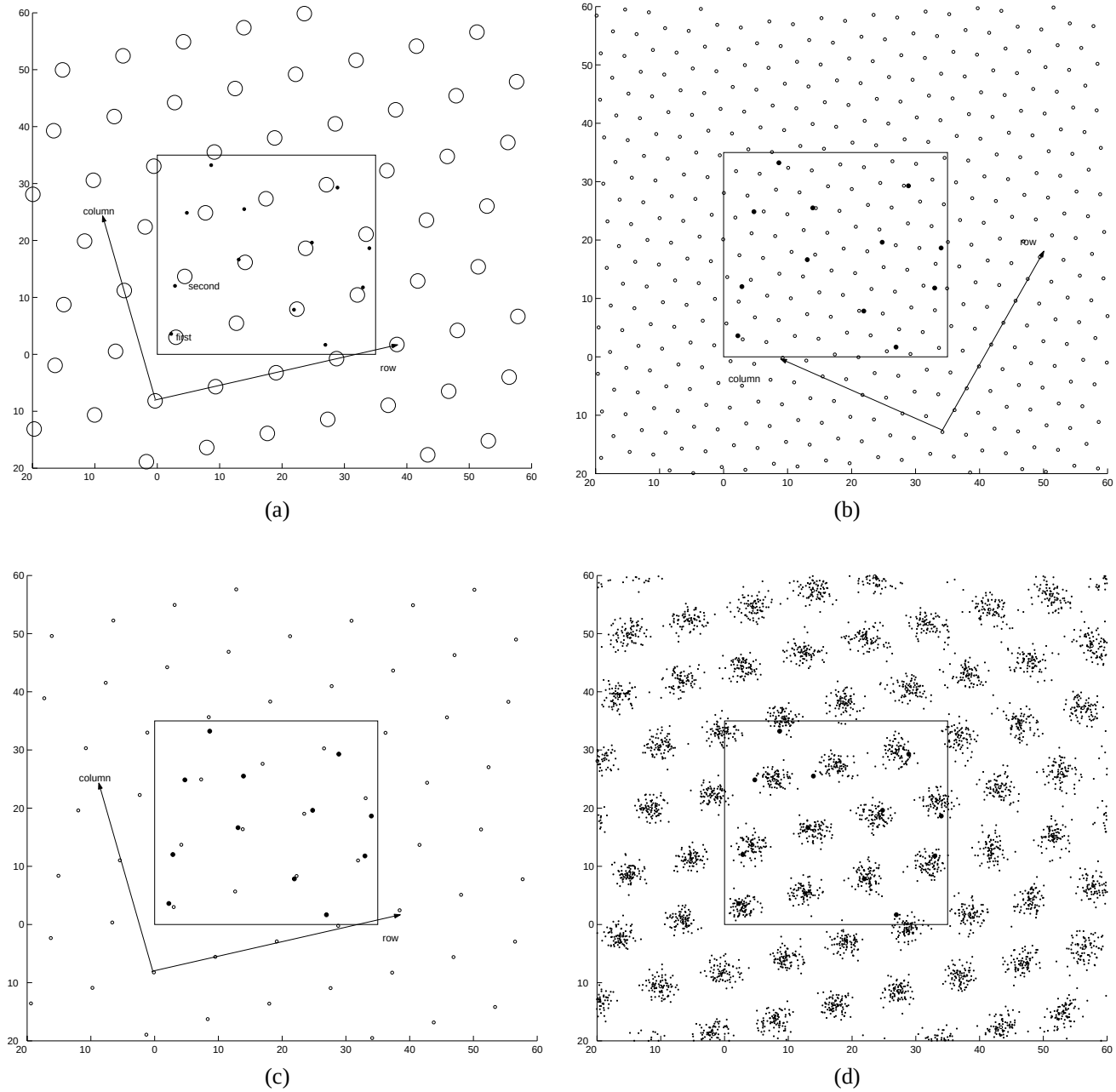


Fig. 26. Simulation snapshots of the algorithm. (1) The intended minefield, depicted by the circles, has rows and columns tilted to the sides of the rectangle. Mines are denoted as black dots. Using the locations of the two mines first discovered in the simulated minefield, the mode partitioning algorithm calculates the possible parameter subspaces. In each subspace, a general MCMC algorithm simulates the samples of the posterior density of the parameters based on the locations of the ten mines (black dots) detected in the area enclosed by the rectangle. (2), (3) Small circles depict the tested minefields as the algorithm reaches the local maximum of the posterior density. (4) The algorithm predicts the locations where the mines may exist. The predicted locations of mines (area covered by small dots) cover the "true" intended minefield (small circles).

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

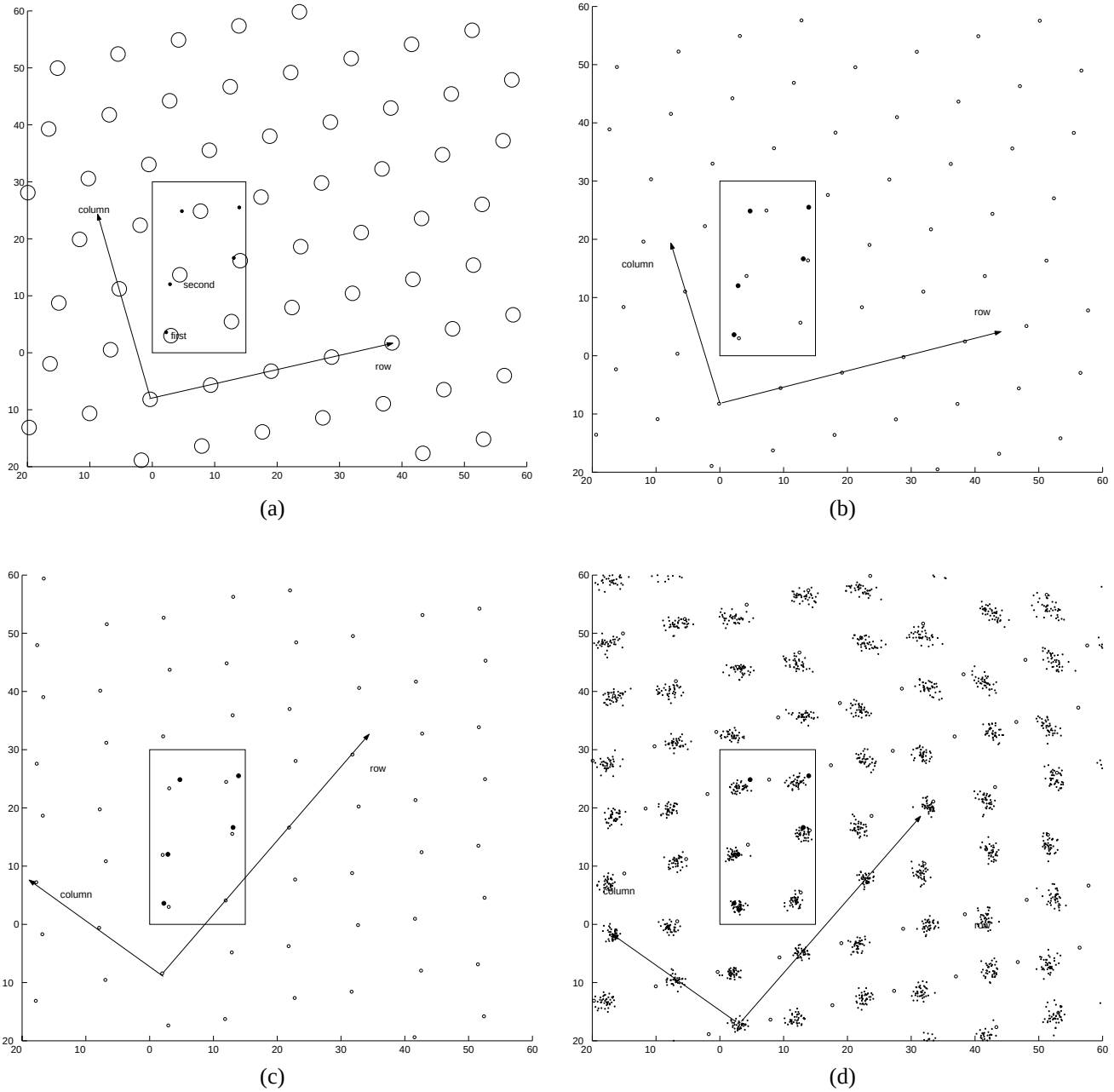


Fig. 27. (1) The intended minefield and detected mine locations in the covered region are the same as in the previous example. But, in this simulation run, the MCMC algorithm is based on the locations of the five mines detected in a smaller rectangle area than that of the previous example. The "wrong" intended minefield (3) matches the five detected mines better than the "true" minefield (2). (4) The predicted locations of the mines do not cover the "true" intended minefield.

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.

investigated (Choset, Nagatani, and Lazar 1999). Even though we have overcome the problems, to a certain extent, due to sensor noise, we still observe the effect of the dead-reckoning error. As part of our future work, we plan to develop localization algorithms using topological features of the space as natural landmarks. We are planning to use critical points as the features for localization.

For demining scenarios where time is limited and some information about the minefield is known a priori, we have developed a probabilistic method that extracts the minefield parameters. Once these parameters are determined, the minefield layout is fixed. Then it is possible to guide the robot opportunistically to decrease demining time.

Acknowledgments

The authors gratefully acknowledge the support of the Office of Naval Research Young Investigator Program Grant #N00014-99-1-0478, Tom Swean, and the Naval Explosive Ordnance Division for support of this work.

References

Acar, E. U., and Choset, H. 2000. Critical point sensing in unknown environments. In *Proceedings of IEEE ICRA'00, International Conference on Robotics and Automation*, San Francisco, CA, pp. 3803–3810.

Acar, E. U., and Choset, H. 2002. Sensor-based coverage of unknown environments: Incremental construction of Morse decompositions, *International Journal of Robotics Research* 21:345–366.

Acar, E. U., Choset, H., Rizzi, A. A., Atkar, P., and Hull, D. 2002. Morse decompositions for coverage tasks, *International Journal of Robotics Research* 21:331–344.

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

Cao, Z. L., Huang, Y., and Hall, E. 1988. Region filling operations with random obstacle avoidance for mobile robots, *Journal of Robotic Systems* 87–102 (February).

Chazelle, B. 1984. Convex partition of polyhedra: A lower bound and worst-case optimal algorithm, *SIAM Journal on Computing* 13(3):488–507.

Choset, H., Nagatani, K., and Lazar, N. 1999. The arc-transversal median algorithm: an approach to increasing ultrasonic sensor accuracy. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Detroit, MI.

DeBolt, C. K., Aviles, J. D., DeLeon, J. D., Nguyen, T. B., and Nguyen, T. N. 2000. Development of a simple, autonomous system of small robots to clear unexploded submunition ordnance. *Unmanned Ground Vehicle Technology II, Proceedings of SPIE* 4024:253–262.

Erdmann, M. 1986. Using backprojections for fine motion planning with uncertainty, *International Journal of Robotics Research* 5(1):19–45.

Gage, D. 1995. Many-robot MCM search systems. In *Proceedings of the Autonomous Vehicles in Mine Countermeasures Symposium*, April, Monterey, CA, pp. 9-55, 9-63.

Gelenbe, E., and Cao, Y. 1998. Autonomous search for mines, *European Journal of Operational Research* 108:319–333.

Healey, A. J., McMillan, S., Jenkins, D., and McGhee, R. B. 1995. BUGS: Basic UXO gathering system. In *Proceedings of the Autonomous Vehicles in Mine Countermeasures Symposium*, April, Monterey, CA, pp. 8-32, 8-33.

Hert, S., Tiwari, S., and Lumelsky, V. 1996. A terrain-covering algorithm for an AUV, *Autonomous Robots* 3:91–119.

Lake, D. E., and Keenan, D. M. 1995. Identifying minefields in clutter via collinearity and regularity detection. *Detection and Remediation Technologies for Mines and Minelike Targets*, A. C. Dubey, I. Cindrich, J. M. Ralston, and K. Rigano, eds., *Proc. SPIE*, 2496:519–530.

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

Lozano-Perez, T., Mason, M. T., and Taylor, R. H. 1984. Automatic synthesis of fine-motion strategies for robots, *International Journal of Robotics Research* 3(1):3–24.

Lumelsky, V. J., Mukhopadhyay, S., and Sun, K. 1990. Dynamic path planning in sensor-based terrain acquisition, *IEEE Transactions on Robotics and Automation* 6(4):462–472.

Milnor, J. 1963. *Morse Theory*, Princeton University Press, Princeton, NJ.

Nomad. 1996. *Nomad 200 User's Manual*, Nomadic Technologies, Inc., Mountain View, CA.

O'Rourke, J. 1998. *Computational Geometry in C*, Cambridge University Press, Cambridge, UK.

Reeb, G. 1946. Sur les points singuliers d'une forme de Pfaff completement integrable ou d'une fonction numerique, *Comptes Rendus Acad. Sciences Paris* 222:847–849.

Tierney, L. 1994. Markov chains for exploring posterior distributions (Disc: P1728-1762), *The Annals of Statistics* 22:1701–1728.

Trevelyan, J. 1998. Robots: A premature solution for the land mine problem. In *Proceedings of the Eighth International Symposium on Robotics Research*, eds. Y. Shirai and S. Hirose, pp. 382–390.

Zelinsky, A., Jarvis, R. A., Byrne, J. C., and Yuta, S. 1993. planning paths of complete coverage of an unstructured environment by a mobile robot. In *Proceedings of International Conference on Advanced Robotics*, November, Tokyo, Japan, pp. 533–538.

Zhang, Y., Schervish, M., Acar, E. U., and Choset, H. 2001. Probabilistic methods for robotic landmine search. In *Proceedings of IEEE IROS'01, International Conference on Intelligent Robots and Systems*, Maui, Hawaii, USA, pp. 1525–1532.

Downloaded from http://ijr.sagepub.com at CARNEGIE MELLON UNIV LIBRARY on August 10, 2008
© 2003 SAGE Publications. All rights reserved. Not for commercial use or unauthorized distribution.