

Robust Sensor-based Coverage of Unstructured Environments

Ercan U. Acar
Howie Choset
Carnegie Mellon University
Pittsburgh, PA 15213
eua,choset+@andrew.cmu.edu

Abstract

Sensor-based coverage uses sensor information to determine a path that passes a detector or some effector over all points in an unknown space. This work identifies features of a provably complete coverage algorithm to reject “bad” sensor readings in unstructured environments without performing complicated sensor-data processing. First, we briefly review our provably complete sensor-based coverage algorithm that uses an exact cellular decomposition in terms of critical points of Morse functions. Then we present features of the algorithm that are used to overcome failures due to bad sensor data. We verify our approach by performing experiments using a mobile robot that has 16 ultrasonic sensors.

1 Introduction

Sensor-based coverage has many applications from floor cleaning to de-mining. Our early work resulted in a sensor-based provably complete coverage algorithm for planar environments. Even though these spaces were unknown *a priori*, we had to form obstacle boundaries using cardboard walls, i.e., form planar extrusions, to make the planar space “friendly” to our sonar ring. In this paper, we extend our early work to the case of unstructured environments (Fig. 1) where “bad” sensor readings often occur.

Our approach to coverage uses a cell decomposition where coverage in each cell can be achieved by performing back and forth motions. Visiting each cell ensures complete coverage. We define each cell by passing a slice (a line segment) through the free space from left to right. The “left-most” and “right-most” boundaries of cells occur at slices where the connectivity of the slice in the free space changes. These connectivity changes occur at the points called *critical points*. Our prior work described how to sense these critical points in planar structured environments. Bad sensor readings cause trouble when the robot is fooled into thinking that it sensed a critical point. However a thorough analysis of our coverage algorithm enabled us to reject such bad sensor readings and verify correct ones. Previously researchers [5], [7] 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 re-analyze our algorithm to identify features that can be used to overcome failures due to sensor uncertainty.



Fig. 1. Unstructured indoor environment with sonar “unfriendly” obstacles.

2 Sensor-based Coverage

Our sensor-based coverage algorithm is built upon “a slicing method” [2] that sweeps a slice \mathcal{CS}_λ (a line segment) through out the configuration space \mathcal{CS} of a circular robot. The slice is the pre-image of a real-valued function $h(x) = x_1$, i.e. $\mathcal{CS}_\lambda = \{x \in \mathcal{CS} | h(x) = \lambda, \lambda \in \mathcal{R}\}$. We use the critical points of $h|_{\partial\mathcal{CC}}(x)$, the restriction of $h(x)$ to the boundaries of the configuration space obstacles $\partial\mathcal{CC}$ as described in the next section to decompose the space.

2.1 Critical Points

We know from basic calculus that at the critical points of a function, its first derivative vanishes and the function takes a local extremum. Also a critical point is *non-degenerate* if and only if the Hessian of the function at the critical point is non-singular. If all the critical points of a function are non-degenerate, then the function is a *Morse function* [8].

To decompose the space into cells, we use the critical points of $h|_{\partial\mathcal{CC}}(x)$ ¹. In our early work [1], we showed that at a critical point of $h|_{\partial\mathcal{CC}_i}(x)$, the gradient of $h(x)$, $\nabla h(x)$, and surface normals, $\nabla m(x)$, of the obstacles \mathcal{CC}_i are parallel to each other (Fig. 2-(a)).

¹In fact, we assume that obstacle boundaries have a structure that ensures $h|_{\partial\mathcal{CC}}$ to be Morse. Fomenko and Kunii [6] show that there always exists a Morse function. In other words, given the space it is always possible to find a real-valued function with non-degenerate critical points.

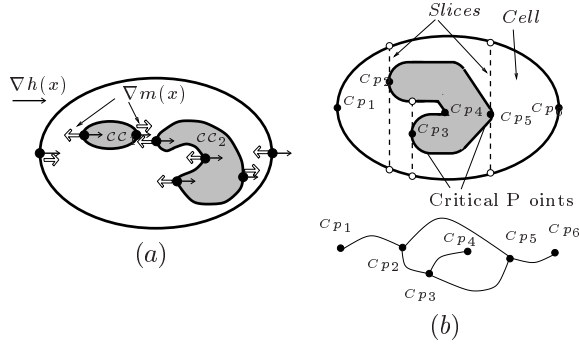


Fig. 2. (a) At the critical points of $h|_{\partial C_i}$, the gradient of h , $\nabla h(x)$, and the surface normals, $\nabla m(x)$, are parallel. (b) An example exact cellular decomposition and its Reeb graph. The cell boundaries have two parts: slices that contain critical points and portions of the obstacle boundaries.

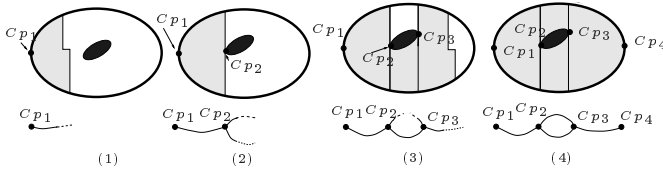


Fig. 3. Incremental construction of the graph while the robot is covering the space. Gray area depicts the covered region.

2.2 Cells and Graph Representation of the Cellular Decomposition

We know that as we sweep the slice in the free space, its connectivity changes at the critical points [3]. We use these connectivity changes to form cells by identifying slices that contain critical points and portions of obstacle boundaries between the critical points. In Fig. 2-(b) we show the cellular decomposition of an example space and its Reeb graph [6] representation that describes the topology of the cellular decomposition. The Reeb graph represents the critical points as nodes and cells as edges. In this work, we assume that “left” or “right” most cell boundaries in the interior of the free space is defined by one critical point each. Therefore generically, we can characterize each cell by two critical points and represent it with an edge between two nodes. Note that dealing with non-generic configurations is an implementation problem.

2.3 Incremental Construction of the Cellular Decomposition

To achieve coverage in an unknown space, the robot simultaneously covers the space and incrementally constructs the Reeb graph representation. In Figure 3-(1), the robot starts to cover the space at the critical point Cp_1 and it instantiates an edge with only one node. When the robot is done covering the cell between Cp_1 and Cp_2 , it joins their corresponding nodes with an edge (Figure 3-(2)). Now the robot has two new uncovered cells. It chooses the lower cell to cover. When

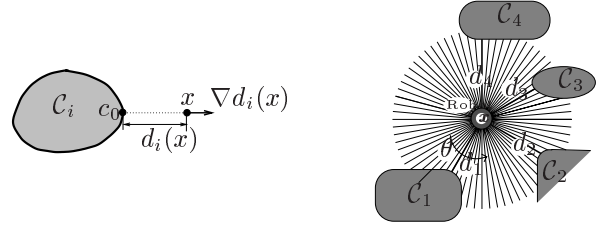


Fig. 4. The distance between x and the workspace obstacle C_i is the distance to the closest point $c_0 \in \partial C_i$. The gradient of the distance function $\nabla d_i(x)$ is the unit vector pointing away from $c_0 \in \partial C_i$. The distances to obstacles, $d_i(x)$, are the local minima of all the distance measurements with respect to angular parameter θ . The corresponding angle θ_i is used to calculate $\nabla d_i(x)$.

the robot reaches Cp_3 , nodes of Cp_2 and Cp_3 become connected with an edge and the lower cell is completed (Figure 3-(3)). At Cp_3 , the robot decides to cover the cell to the right of Cp_3 . When the robot senses Cp_4 , it goes back to Cp_3 and starts to cover the upper cell. When it comes back to Cp_2 , it determines that all the edges of all the nodes (critical points) have been explored (Figure 3-(4)). Thus the robot concludes that it has completely covered the space. This incremental construction method serves a basis for the sensor-based coverage algorithm.

2.4 Critical Point Sensing Methods

In unknown environments, the incremental construction approach that we described requires methods to sense critical points. In this section we present two methods. The first method uses range sensors, and the second method uses relative position data. We use these methods concurrently to eliminate both false-negatives and positives.

2.4.1 Critical Point Sensing Using Range Sensors

We use a distance function $d_i(x)$ to model range sensors. The value $d_i(x)$ is the shortest distance between a point x and a workspace obstacle C_i . The distance function and its gradient are (Fig. 4)

$$d_i(x) = \min_{c \in C_i} \|x - c\| \quad \text{and}$$

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

We calculate $d_i(x)$ and $\nabla d_i(x)$ by using a range sensor, such as a sonar ring, that supplies distance measurements radially distributed around the robot. The distances $d_i(x)$ to obstacles are determined by calculating the local minima of the range measurements with respect to an angular parameter θ (Fig. 4). 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

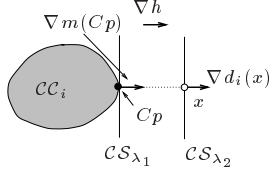


Fig. 5. At point x , $\nabla d_i(x) \perp \mathcal{CS}_{\lambda_2}$. Therefore there exists a critical point Cp at $x - \nabla d_i(x)d_i(x)$. Note that the critical point is located in the boundary of the configuration space obstacle.

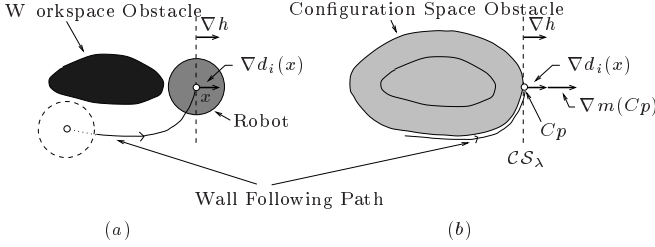


Fig. 6. (a) The robot senses the critical point Cp while it is following the boundary of the obstacle using the workspace distance measurements. (b) At point x , $\nabla d_i(x) \perp \mathcal{CS}_{\lambda}$, $\nabla m(Cp) \parallel \nabla h(Cp)$ and $\nabla m(Cp) \parallel \nabla d_i(x)$. The critical point Cp lies in the boundary of the configuration space obstacle. The position of the center of the robot is, in fact, the critical point.

sensing. In our prior work [1], we showed that if the robot is located at $x \in \mathcal{CS}_{\lambda_2}$ and $\nabla d_i(x)$ is orthogonal to the slice, then there exists a critical point on the boundary of the configuration space obstacle, $\partial \mathcal{CC}_i$, at $(x - \nabla d_i(x)d_i(x)) \in \partial \mathcal{CC}_i$ (Fig. 5).

In this work, we restrict the robot to sense the critical points while it is following the boundaries of the obstacles. Then if the robot senses a critical point (Fig. 6-a), it is located in configuration space at a position that corresponds to the center of the robot (Fig. 6-b).

2.4.2 Critical Point Sensing Using Relative Position Information

We know that at 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 wall following. In Fig. 7 we

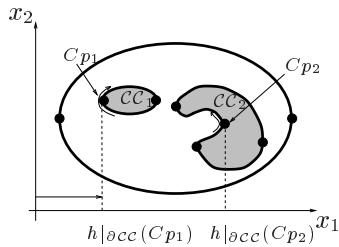


Fig. 7. The restriction of the slice function $h(x) = x_1$ to the obstacle boundaries $h|_{\partial \mathcal{CC}}$ takes a local minimum at Cp_1 and a local maximum at Cp_2 . Since $h|_{\partial \mathcal{CC}}$ takes its extrema at Cp_1 and Cp_2 , they are the critical points.

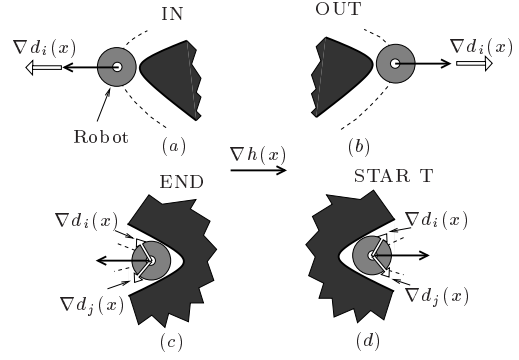


Fig. 8. The robot can use the direction of $\nabla d_i(x)$ with respect to $\nabla h(x)$ to identify four different types of critical points. (a) $-\nabla h(x) = \nabla d_i(x)$, IN (b) $\nabla h(x) = \nabla d_i(x)$, OUT (c) $-\nabla h(x) \in CO\{\nabla d_i(x), \nabla d_j(x)\}$, END (d) $\nabla h(x) \in CO\{\nabla d_i(x), \nabla d_j(x)\}$, START.

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 Fig. 7. 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 $h|_{\partial \mathcal{CC}}$.

2.5 Types of Critical Points and How to Sense Them

Borrowing terms from computational geometry [10], we classify the critical points as IN, OUT, START, END and use this characterization later in Sec. 3 to reject bad sensor readings. Recall that critical points occur on the boundaries of obstacles. When the obstacle is locally convex near the critical point, we 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 (Fig. 8-(a, b)). When the obstacle is locally concave, we have a START or an END critical point. We can sense these types of critical points when curvature of the robot's periphery is greater than the obstacle's. Note that at such critical points, the gradient of $d(x)$ is non-smooth. Then at an END critical point $-\nabla h(x) \in CO\{\nabla d_i(x), \nabla d_j(x)\}$ (Fig. 8-(c)). At a START critical point $\nabla h(x) \in CO\{\nabla d_i(x), \nabla d_j(x)\}$ (Fig. 8-(d)) where $CO\{\nabla d_i(x), \nabla d_j(x)\}$ is the convex hull of $\nabla d_i(x)$ and $\nabla d_j(x)$. When the boundary's curvature is smaller than the robot's periphery, i.e. the gradient of $d(x)$ is smooth, we cannot distinguish the START and END critical points with IN and OUT critical points using range data (Fig. 9). However, by observing the relative history of the dead-reckoning data and using the features of the algorithm, presented in

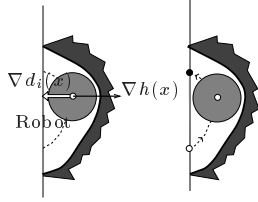


Fig. 9. The curvature of the boundary is smaller than $1/r$ where r is the radius of the robot. In this case the robot uses relative position data and features of the algorithm to sense and determine the type of the critical point.

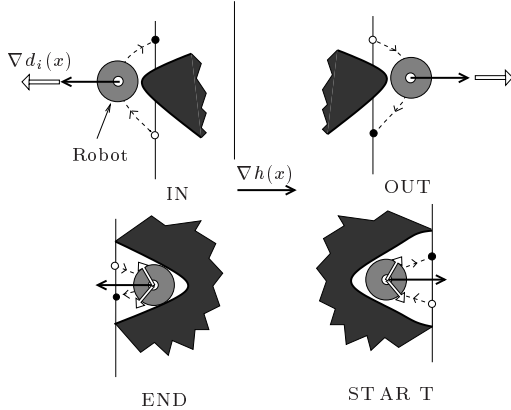


Fig. 10. The robot uses relative position data to determine the existence of local extrema and hence to sense the critical points. IN and START critical points occur when there is a local minimum. OUT and END critical points occur when there is a local maximum.

Sec. 3, we can make this judgment.

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 (Fig. 10). 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. Even though the robot cannot distinguish between the types of critical points between IN or START (similarly OUT or END), in Sec. 3 we show how to deal with this ambiguity.

3 Encountering all Critical Points and Rejecting Bad Sensor Data: Cycle Algorithm

Now that we know how to sense and determine the types of critical points, in this section we present a provably complete algorithm that allows the robot to simultaneously cover the space and look for the critical point that indicates the completion of a cell (for the completeness proof of the algorithm see [1]). Our coverage algorithm basically runs the cycle algorithm, described below, repetitively and as the robot encounters

the critical points, 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. Without loss of generality, the robot starts the cycle algorithm at S_i in slice \mathcal{CS}_{λ_i} and lies in the ceiling² of the cell being covered. Any lapping path (motion along a straight line) followed in a direction from ceiling towards floor is referred as forward lapping. We also refer to sweep direction as forward. From S_i the robot looks for critical points via the following phases (Fig. 11-(a)).

1. **Forward phase:** The robot follows a forward lapping path starting at S_i along \mathcal{CS}_{λ_i} towards the floor. Then it follows the wall in the forward direction. The robot terminates forward wall following and the forward phase if it laterally moves one lap width or encounters a critical point in the floor.
2. **Reverse phase:** The reverse phase is an interleaved sequence of reverse laps, (towards ceiling), and reverse wall following paths. A reverse wall following path initially starts in the reverse direction. Each reverse wall following operation terminates when the robot either
 - (a) senses a critical point Cp_k with $\nabla d_i(Cp_k) = -\nabla h(Cp_k)$ ³, or
 - (b) returns back to the slice that contains the start point S_i , and thus completes the reverse phase.
3. **Closing phase:** The robot may reach the start point S_i at the end of the reverse phase, but if not, it executes the closing phase as follows. The robot follows lapping paths along the slice \mathcal{CS}_{λ_i} inter-mixed with wall following paths. Closing wall following paths initially start in the forward direction, and terminate when the robot returns to \mathcal{CS}_{λ_i} .

EXAMPLE 3.1 An example path generated by the repeated execution of the cycle algorithm for coverage in a hallway is shown in Fig. 11-(b). The robot starts to lap at point S_1 , encounters an object and performs wall following. Then it performs *reverse lapping* and at the end of *reverse wall 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 wall following step. Let S_2 be the beginning of the next cycle, but we locate it at the start point of the last lapping motion. The robot does not need to drive to S_2 , because it has already traveled along previous

²We borrowed the terms ceiling and floor from computational geometry literature [10]. Ceiling refers to the upper boundary of a cell and floor refers to the lower boundary.

³Note that if $\nabla d_i(Cp_k) = \nabla h(Cp_k)$, robot continues to follow the boundary.

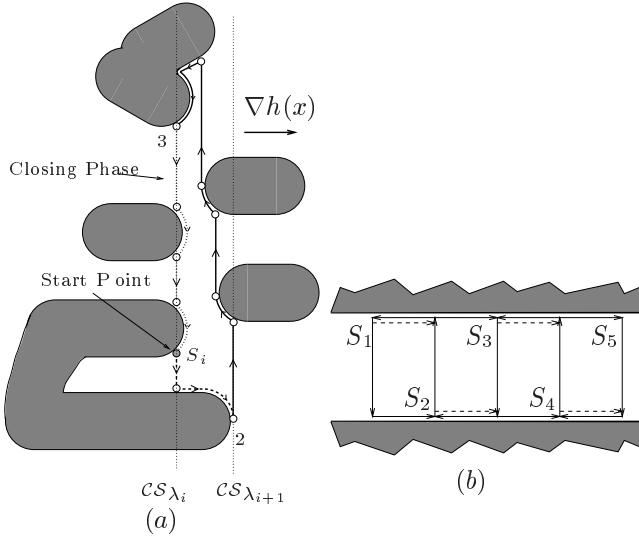


Fig. 11. (a) Phases of the cycle algorithm. In the forward phase, the robot follows the dashed path between S_i and point 2. In the reverse phase, the robot follows the solid path between points 2 and 3. Finally, in the closing phase the robot follows the dotted path between points 3 and S_i . (b) Path generated by the cycle algorithm in a simple space. S_i refers to start point of each cycle.

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

TABLE 1

TYPES OF CRITICAL POINTS FOR EACH PHASE

lap. So it completes the next cycle starting with wall following, pretending as if it has started at S_2 .

3.1 Types of Critical Points Encountered in Each Phase

Along each path followed in each phase of the cycle algorithm, there can exist only certain types of critical points (due to space restrictions we do not give the related proofs) as summarized in Table 1. In the forward phase, the robot can only sense OUT or END type of critical point (Fig. 12-(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 (Fig. 12-(b)). Finally in the closing phase, the robot can only sense OUT and/or START type of critical point (Fig. 12-(c)).

In the experiments we performed using a Nomad Scout mobile robot [9] that has 16 sonar sensors, we frequently encountered bad sensor readings. We use Table 1 to reject such bad sensor data. For example, in Fig. 13, the robot senses an IN critical point while it is executing the forward phase right before it senses an END critical point. Since there cannot exist an IN critical

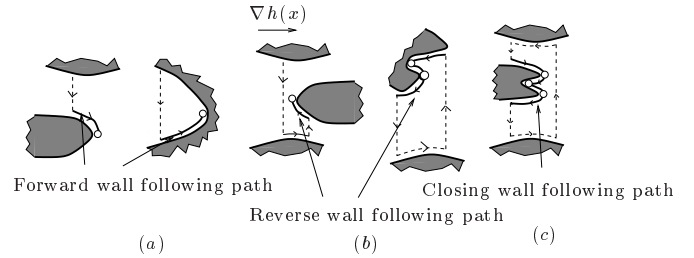


Fig. 12. (a) Only OUT or END type of critical point t can exist along a forward-phase path (b) Only IN and/or START and OUT types of critical points can exist along a reverse-phase path (c) Only OUT and/or START type of critical point can exist along a closing-phase path.

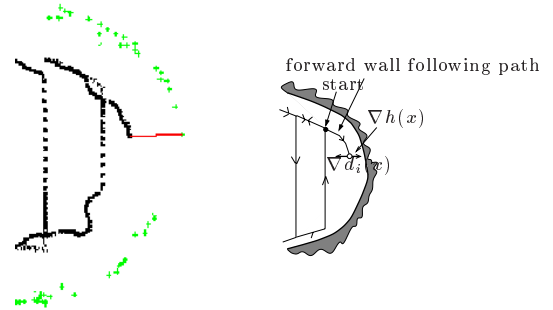


Fig. 13. Experimental data and sketch of the path followed by the robot. Gray dots are the sonar returns and black dots are the path followed by the robot. An IN critical point is sensed during forward wall following $\nabla d_i(x) = -\nabla h(x)$ due to sensor noise. However according to Table 1, during forward wall following, the robot cannot sense an IN critical point. Therefore the robot continues to perform forward wall following until it senses the END critical point.

point along the forward-phase path (Table 1), the robot ignores the sensor data that indicates an IN critical point and continues to follow the boundary until it senses the END critical point.

3.2 Special Note on Closing Phase

The closing phase of the cycle algorithm is executed when the robot cannot reach the starting point of the cycle path at the end of the reverse phase. In the closing phase the robot is ensured to reach starting point of the cycle by performing wall following and lapping motions. Moreover, the end point of the closing phase should lie above the starting point of the cycle along the slice in which the robot has started the cycle. However, as seen in Fig. 14, the robot may end up at a lower position along the slice at the end of the closing phase because of the sensor noise. In this case, since the distance between the start and end points is bigger than the error threshold, the robot could have thought that it had not reached the starting point of the cycle and continued to execute the closing phase. However, we know that the robot cannot end up at a lower position, if that is the case, then it must be because of bad sensor

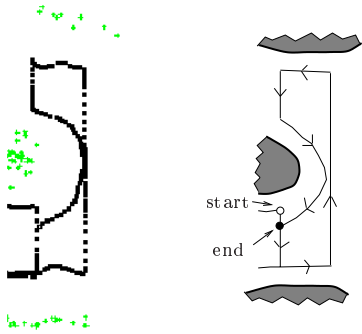


Fig. 14. The end point of the closing phase should always lie above the starting point t of the cycle along the slice.

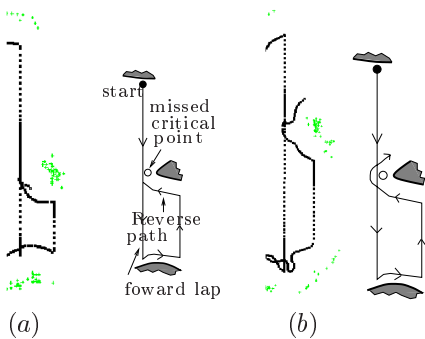


Fig. 15. A commonly encountered non-generic configuration. (a) The robot just passes by a critical point t while it is lapping. The robot has to sense the critical point during reverse wall following motion. However, it reaches the slice before sensing it. (b) We solve this problem by making the robot continue to perform reverse wall following motion until the robot senses a critical point t .

data. Therefore the robot concludes that it has reached the starting point of the cycle path and continues to execute the coverage algorithm.

4 Non-Generic Configurations Commonly Encountered

The lap-width is determined by the detector's range. Therefore for small detector ranges, the robot has to perform laps close to each other. This increases the chances of encountering non-generic configurations of the obstacles. We identify two commonly encountered non-generic configurations. The first one is depicted in Fig. 15-(a, b). The robot just passes by a critical point while it is lapping in the forward direction. Now it has to sense the "missed" critical point while it is performing reverse wall following. However because of the sensor noise, the robot reaches the forward lapping path (travels laterally one lap-width) and terminates the reverse wall following motion without sensing the critical point.

To deal with this problem, we note the possible locations of the end point of the reverse phase. It should always be above the starting point of the cycle path. In

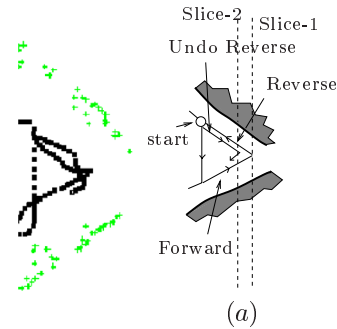


Fig. 16. Non-generic position. The critical point is located on lapping path. The robot could not reach the slice at the end of the undo reverse wall following motion.

the situation we described, this is not the case. Hence the robot concludes that there must be a critical point very close to the previous forward lapping path. Therefore the robot continues to perform the reverse wall following motion until it senses the critical point. 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 Fig. 16. The robot starts with a forward wall following path. At the end of the forward wall following motion, the robot reaches the corner and executes a reverse lap with zero path length. The robot starts to perform a reverse wall following path and when it is done, the robot starts to "undo" the reverse wall following path by following the wall in the forward direction. Normally we expect the robot to reach the slice-1. However because of sensor noise, the robot cannot reach the slice-1 and continue to follow the wall. Since the robot continues to perform undo reverse wall following motion, it misses the end critical point. We solved this problem by making the robot look for the END critical point while it is undoing the reverse wall following motion using range and position data.

5 Incremental Construction with a Mobile Robot in Unstructured Environments

In the previous sections, we showed snapshots of experimental data collected while the robot was performing coverage in unstructured rooms that has obstacle boundaries as shown in Fig. 1. In this section, we show full successful coverage experiments. We processed the distance measurements made by the 16 sonars of the Nomad Scout mobile robot using a method [4] that improves the angular resolution of the distance measurements. Note that this sonar processing method does

not eliminate bad sonar data. We use the processed data to find the closest point on the closest object to the robot. In other words, we calculate the global minimum of the processed 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 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 17 shows different stages of a coverage experiment in a $2.5 \times 3.1[m]$ room with a stool in the middle and a time-exposure photograph to show the area swept by a light stick that is as wide as the robot. 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 walls following the path. Note how the wall 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 workspace distance measurements. Unlike prior work [1], the robot determined the locations of the critical points more robustly and precisely using both range and relative position data as described in Sec. 2.4. In Fig. 17-(1), 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 Fig. 18, we show the coverage path in a more complicated $4 \times 4.6[m]$ 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 (Fig. 19-(a)). Since the end point of the reverse wall following path is below the start point of the cycle path, as we described in Sec. 4, the robot continues to follow the boundary until it senses the critical point. Around point *B*, the robot receives sonar data that indicates that the surface normal of the obstacle and the sweep direction are parallel while it is performing reverse wall following (Fig. 19-(b)). This is the condition for an IN critical point. However, the robot does not sense a local minimum to verify the existence of an IN critical point. Therefore the robot rejects the bad sonar data and finishes the reverse wall following motion.

We performed an experiment in a $2 \times 2[m]$ living

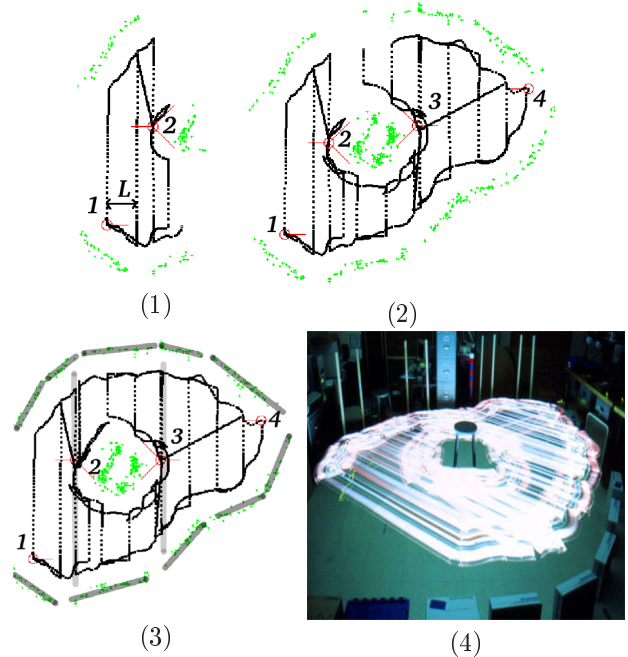


Fig. 17. Three stages of the coverage in an unknown environment and a time-exposure photograph that shows the covered area. The robot incrementally constructs the graph representation by sensing the critical points 1, 2, 3, 4, 2 (in the order of appearance) while covering the space. In the final stage (3), since all the critical points have explored edges, the robot concludes that it has completely covered the space. For the sake of discussion, we outlined the boundaries of the obstacles and cells in (3). $L = 0.40[m]$.

room of an apartment that has a coffee table in the middle (Fig. 20). We observed two failures and recoveries from them. Around point *A* (Fig. 21-(a)), the robot senses an IN critical point using range data during forward wall following path. However, we know that along a forward wall following path, there cannot exist an IN critical point (Sec. 3.1). 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* (Fig. 21-(b)), the robot senses an IN critical point using range data along the reverse wall 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 wall following motion. In this experiment, we observed the effect of the dead-reckoning error. Robot’s perception of the left corner of the coffee table was rotated and shifted. Even though in this experiment, dead-reckoning error did not cause a problem, in larger spaces we need to develop localization methods for coverage.

6 Conclusions

Most coverage tasks require a complete coverage algorithm that works in unstructured environments. Our

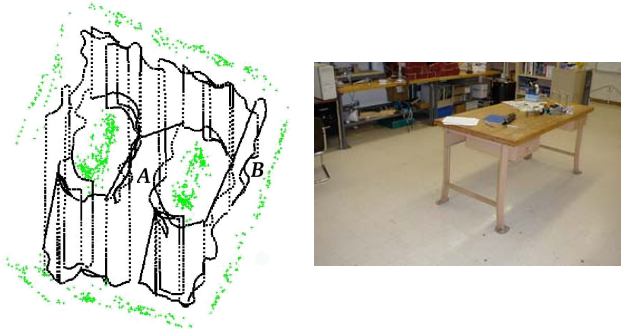


Fig. 18. Coverage path followed by the robot and the sonar returns in $4 \times 4.6[m]$ room with a table in the middle. The robot successfully covers the room. Note that the robot covers the area under the table too.

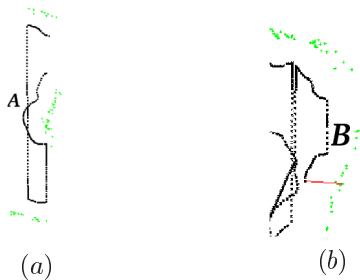


Fig. 19. Blown up shots of the data shown in Fig. 18. (a) A non-generic configuration. The robot continues to follow the boundary until it senses the critical point. (b) The robot cannot verify the IN critical point indicated by the sonar data using the relative position data. Therefore it rejects the sonar data.

earlier work resulted in a provably complete coverage algorithm that works in unknown environments that has cardboard or flat walls. In this paper we introduced methods and identified features of our algorithm that can be used to overcome failures due to sensor noise. Our approach does not require any complicated sensor-data processing algorithms. However in the future we are planning to develop such processing algorithms to further improve the performance of our implementation. We verified our approach by performing experiments with a mobile robot that has a sonar ring. Even though we overcame the problems due to sensor noise, we still observe the effect of the dead-reckoning error. As a part of the future work, we plan to develop localization algorithms using topological features of the space as natural landmarks. Finally we would like to develop a systematic framework to extend the ideas that we use to reject bad sensor readings to other planning algorithms.

Acknowledgements

The authors gratefully acknowledge the support of the Office of Naval Research Young Investigator Program Grant #N00014-99-1-0478, Tom Swain, and the

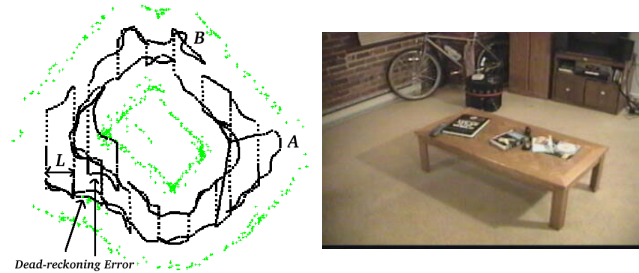


Fig. 20. Coverage path followed by the robot and the sonar returns in $2 \times 2[m]$ room with a coffee table in the middle. The robot successfully covers the room. Dead-reckoning error is observable.

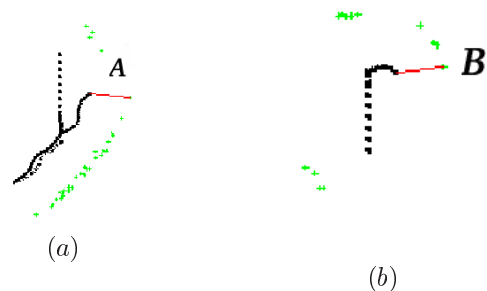


Fig. 21. Blown up shots of the data shown in Fig. 20. (a) Bad sonar data indicates an IN critical point along a forward-phase path. However there cannot exist an IN critical point in the forward-phase, therefore the robot rejects the data. (b) Bad sonar data indicates an IN critical point, but the robot cannot verify it using the relative position data. Therefore the robot rejects the data.

Naval Explosive Ordnance Division for support of this work.

References

- [1] E. Acar and H. Choset. Critical point sensing in unknown environments. In *Proc. of IEEE ICRA'00*, San Francisco, CA, 2000.
- [2] J.F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1988.
- [3] H. Choset, E. Acar, A. Rizzi, and J. Luntz. Exact cellular decompositions in terms of critical points of morse functions. In *Proc. of IEEE ICRA'00*, San Francisco, CA, 2000.
- [4] H. Choset, K. Nagatani, and N. Lazar. The Arc-Transversal Median Algorithm: an Approach to Increasing Ultrasonic Sensor Accuracy. In *Proc. IEEE Int. Conf. on Robotics and Automation*, Detroit, 1999.
- [5] M. Erdmann. Using backprojections for fine motion planning with uncertainty. *International Journal of Robotics Research*, 5(1):19-45, 1986.
- [6] A. T. Fomenko and T. L. Kunii. *Topological Modeling for Visualization*. Springer-Verlag, Tokyo, 1997.
- [7] T. Lozano-Perez, M.T. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1):3-24, 1984.
- [8] J. Milnor. *Morse Theory*. Princeton University Press, Princeton, New Jersey, 1963.
- [9] Nomadic Technologies. *Nomad Scout User's Manual*. Nomadic Technologies, Inc., Mountain View, CA, 1996.
- [10] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1998.