Ercan U. Acar
Howie Choset
Carnegie Mellon University, USA

Sensor-based Coverage of Unknown Environments: Incremental Construction of Morse Decompositions

Abstract

The goal of coverage path planning is to determine a path that passes a detector over all points in an environment. This work prescribes a provably complete coverage path planner for robots in unknown spaces. We achieve coverage using Morse decompositions which are exact cellular decompositions whose cells are defined in terms of critical points of Morse functions. Generically, two critical points define a cell. We encode the topology of the Morse decomposition using a graph that has nodes corresponding to the critical points and edges representing the cells defined by pairs of critical points. The robot simultaneously covers the space while incrementally constructing this graph. To achieve this, the robot must sense all the critical points. Therefore, we first introduce a critical point sensing method that uses range sensors. Then we present a provably complete algorithm which guarantees that the robot will encounter all the critical points, thereby constructing the full graph, i.e., achieving complete coverage. We also validate our approach by performing experiments on a mobile robot equipped with a sonar ring.

KEY WORDS—sensor-based coverage, Morse decompositions, Reeb graph

1. Introduction

Humanitarian demining, autonomous lawn mowing, floor cleaning, and snow removal are robotic applications that require a *coverage algorithm* which can guide a robot to pass a detector or tool over all points in *a priori* unknown environments. If a coverage algorithm ensures that the detector passes over every point of a space, we say it is *complete*. In this paper, we introduce a complete sensor-based coverage algorithm along with its proof of completeness. In demonstrating such a proof, we show that simple raster-scan-like motions are not sufficient for coverage, as previously believed. Moreover, our algorithm requires a "minimal" sensor system that can be used to guide the robot along the obstacle boundaries and detect the surface normals. We achieve this using sonar sensors.

Our algorithm is based on Morse decompositions which are exact cellular decompositions that are formulated using critical points of Morse functions (Acar et al. 2002). The critical point formulation forms simple cells that can be covered by performing simple motions, such as back and forth. This work uses critical points not only for decomposing the space into simple cells, but also to form an "adjacency" graph representation that has edges as the cells and nodes as the critical points to represent the topology of the Morse decomposition. This graph enables us to reduce the sensor-based coverage to an incremental graph construction procedure, i.e., the robot can simultaneously cover the space and, as it encounters critical points, construct the graph incrementally. Since each cell can be completely covered via a simple pattern, complete coverage is achieved simply by ensuring that the robot has identified and visited each cell in the environment, i.e., it has exhaustively constructed the graph.

The incremental construction procedure has two major aspects: critical point sensing and guaranteeing that the robot encounters all critical points in unknown spaces. Essentially, we extend the notion of a "critical point sensor" introduced by Rimon and Canny (1994) to robots with range sensors such as simple sonars and laser-range finders. With the critical point sensing method in hand, we develop a coverage algorithm which guarantees that the robot finds all the critical points in an unknown environment, thereby ensuring the completeness of our critical point-based approach. We verified our coverage algorithm and the critical point sensing method on a Nomad 200 Mobile robot base (Nomad 1996) that has a ring of 16 ultrasonic sensors.

2. Prior Work

The coverage path planning problem has been approached by many researchers using either systematic or random strategies. Originally, it had been believed that random strategies

The International Journal of Robotics Research
Vol. 21, No. 4, April 2002, pp. 345-366,
©2002 Sage Publications

were suitable for inexpensive robots because of their low sensor and computational power requirements (Healey et al. 1995). Random coverage strategies do not possess completeness arguments, however Gage (1995) showed that the percentage of the covered area can be increased by using either numerous robots at once or a small number of robots over long periods of time. Coverage path planning has also been approached using behavior based methods (MacKenzie and Balch 1996) to design random strategies. Recently, a lawn-mowing robot that uses random zig-zag motions for coverage was reported in *Consumer Reports* (2000). It is reported that the robot takes an excessive amount of time to cover a given area and misses five percent of the lawn.

Many systematic coverage path planners use exact cellular decompositions, either implicitly or explicitly. Perhaps the most widely-known exact cellular decomposition is the trapezoidal decomposition (Preparata and Shamos 1985) where each cell is a trapezoid or a triangle. VanderHeide and Rao (1995) use the trapezoidal decomposition to develop a sensor-based coverage algorithm. Their algorithm initially guides the robot along the enclosing boundary of the room while generating a map of the outer boundary and then decomposes the space into sub-regions in the shape of trapezoids. The robot covers each sub-region by back and forth motions.

A minor short-coming of the trapezoidal decomposition for coverage is that many small cells are formed that can seemingly be "clumped" into neighboring cells. To address this issue, the boustrophedon cellular decomposition approach was introduced (Choset and Pignon 1997). The boustrophedon decomposition is formed by considering the vertices at which a vertical line can be extended both up and down in the free space. Straight line segments formed at these vertices between the obstacle boundaries form the cell boundaries in the free space. In this work, we use the generalization of the boustrophedon decomposition beyond polygons introduced by Acar et al. (2002).

The result presented by Cao et al. (1988) is a sensor-based boustrophedon decomposition of unknown environments that are populated only by convex obstacles. They use "extrema values" which are the left-most and right-most points on a convex obstacle to decompose the space. The extrema points are in fact critical points of a Morse function and Cao et al.'s algorithm determines the locations of these extrema points by circumnavigating the obstacles. In our work, we also use critical points to decompose a space. However, our approach is not limited by the types of obstacles and is also suitable for coverage with different patterns.

A coverage algorithm that uses an approach similar to a cellular decomposition, but is methodically different, was presented first by Lumelsky et al. (1990), and later improved by Hert et al. (1996). Their algorithm guarantees that the robot will see everything in unknown environments with a finite-range vision sensor. They place straight lines parallel to each other with a constant spacing to guide the robot. They use the intersection points of these guide lines with the special parts of the boundaries of the obstacles, namely "capes" and "bays", to identify uncovered regions. They assume that all these intersection points can be sensed while the robot is following one of the adjacent guide lines and none of them can be occluded by the obstacles. This assumption limits the generality of the possible obstacle configurations and practicality of the sensor-based implementation.

In this work, we mainly focus on coverage of "non-degenerate" spaces. A coverage algorithm solely designed for degenerate spaces where all the critical points are not isolated was presented by Butler et al. (1998). Their algorithm decomposes the environment using "interesting points" into rectangular cells. Essentially, these interesting points correspond to degenerate critical points. They extend their work to develop a cooperative coverage algorithm that uses multiple robots (Butler et al. 2000). Another multi-robot coverage algorithm that decomposes an unknown environment into trapezoids and triangles using convex vertices of an unknown environment was presented by Rekleitis et al. (1997). They use two robots to improve localization. The robots move one at a time and each robot localizes itself with respect to the other one.

The coverage algorithms that use grid-based methods can be thought of as cellular decompositions with very fine cells. Such a complete coverage algorithm was presented by Zelinsky et al. (1993). They generate a coverage path, using a path transform which can be regarded as a numeric potential field. They propagate a wavefront which is a weighted sum of the distance from the given goal configuration together with a measure of the discomfort of getting too close to obstacles. They generate the coverage path following the path of steepest ascent from goal to a start configuration. One disadvantage of grid-based methods is that they can generate coverage paths with many turns. Since the robot is susceptible to dead-reckoning error at each turn, it can easily experience localization problems after traveling a short distance.

Finally, we would like to note that this paper mainly focuses on complete, but not optimal coverage of unknown environments. For reference on optimal coverage algorithms that work in discrete or known spaces see Spires and Goldsmith (1998), Wagner and Bruckstein (1995), Gabriely and Rimon (2000), and Huang (2001).

3. Sensor-based Coverage: Incremental Construction of Morse Decompositions

Our sensor-based coverage algorithm is based on a Morse decomposition of a compact configuration space $\mathcal{CS} \subset \Re^2$ of a disk robot with a finite boundary and a finite number of configuration space obstacles $\mathcal{CO}_i$. In our previous work, we formulated the Morse decompositions using a "slicing method" (Acar et al. 2002) where a slice in $\mathcal{CS}$ is the pre-image of a real-valued function $h : \Re^2 \rightarrow \Re$, i.e.,

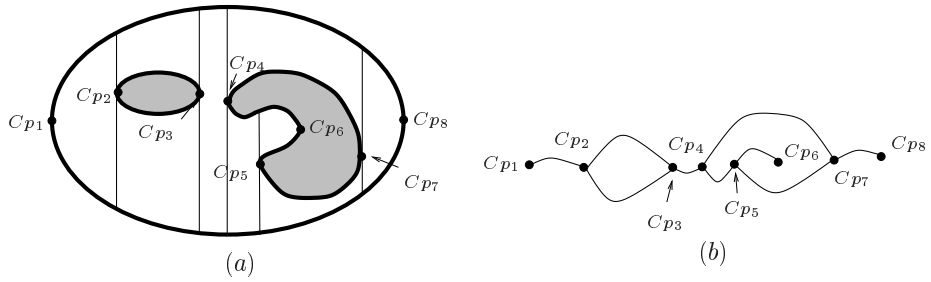


Fig. 1. (*a*) Exact cellular decomposition of the environment in terms of critical points Cp_k. (*b*) Graph representation of the environment. Nodes represent critical points and edges represent cells.

$\mathcal{CS}_\lambda = \{x \in \mathcal{CS} \mid h(x) = \lambda,\ \lambda \in \Re\}$. We showed that the connectivity of $\mathcal{FCS}_\lambda$[1] changes (as $\mathcal{FCS}_\lambda$ is swept through the space by varying λ) at the critical points of $h|_{\partial\mathcal{CC}}$, the restriction of $h(x)$ to the obstacle boundaries $\partial\mathcal{CC}$ (Acar et al. 2002). We used the connectivity changes at the critical points to form the Morse decompositions that has simple cells which can be covered by back and forth motions.

In this work, we use a graph representation, termed the Reeb graph, to encode the Morse Decomposition (Reeb 1946; Fomenko and Kunii 1997). The Reeb graph represents the critical points as nodes and cells as edges (Figure 1). Generically, we can characterize a cell by two critical points and present it with an edge between two nodes that correspond to the critical points. Then "left" or "right" most cell boundary is defined by one critical point each.[2]

The Morse decomposition together with the Reeb graph lends itself to incremental construction while the robot is performing coverage. We depict this construction procedure in Figure 2. The robot essentially looks for critical points while it is covering the space. When the robot encounters critical points, it updates the Reeb graph. When all the edges of all the nodes are explored, the robot concludes that it has completely covered the space. The incremental construction method requires a critical point sensing method that uses range sensors and a coverage algorithm that guarantees that all the critical points will be found.

3.1. Critical Point Sensing

In this section, we show that omni-directional range sensors, modeled by a distance function $d_i(x)$, can be used to calculate the surface normals $\nabla m(x)$ of obstacles and hence to sense the critical points. In our previous work, we proved the following corollary which states that at the critical points the sweep direction and the surface normal of an obstacle are parallel (Acar et al. 2002).

1. The portion of the slice that is contained in the free configuration space $\mathcal{FCS}$, i.e., $\mathcal{FCS}_\lambda = \mathcal{CS}_\lambda \bigcap \mathcal{FCS}$.

2. Note that dealing with two or more critical points on the same slice is an implementation detail.

COROLLARY 1. The point $x \in \partial\mathcal{CC}_i$ is a critical point of $h|_{\partial\mathcal{CC}_i}$ if and only if $\nabla h(x)$ is parallel to $\nabla m(x)$.

By definition, the following corollary, which we will use in Section 3.1, immediately follows.

COROLLARY 2. The slice is tangent to an obstacle boundary at critical points.

The value $d_i(x)$ is the shortest distance between a point x and a convex object $\mathcal{C}_i$[3]. The distance function and its gradient (Figure 3) are formally given as

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

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

The distances $d_i(x)$ and the corresponding gradients $\nabla d_i(x)$ are determined by calculating the local minima of the range measurements with respect to an angular parameter θ. The robot uses these distance measurements made in the workspace to perform coverage. However, since we have formulated the cellular decomposition in the configuration space, first we supply a correspondence between the configuration space and workspace distance measurements. Since we consider a disk-like robot which, in the configuration space, is represented by a point at the center of a $2r$ diameter circular disk, the minimum distance between the disk and an obstacle in the workspace is the same as the distance between the robot's configuration and an obstacle in the configuration space. Likewise, the corresponding gradients are parallel. Therefore we can use the distance function in the configuration space without any added complexity.

Once the distance measurements and the gradient of the slice function are available, we relate them for critical point sensing by using the Lemma 1. The lemma states that the slice $\mathcal{CS}_{\lambda_1}$ contains a critical point if and only if $\nabla d_i(x)$ is orthogonal to the slice $\mathcal{CS}_{\lambda_2}$ where the robot is located at $x \in$

3. We form concave objects as the union of convex objects.

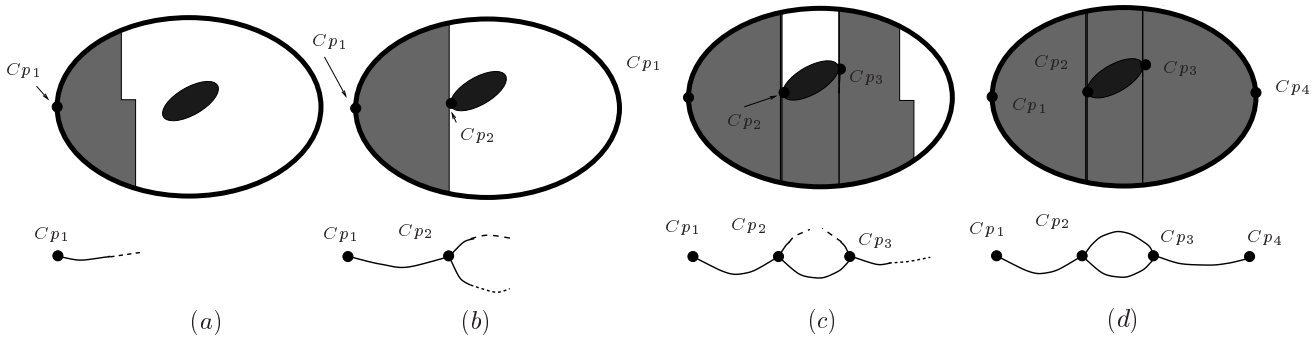


Fig. 2. Incremental construction of the graph while the robot is covering the space. The gray area depicts the covered region. (a) The robot starts to cover the space at the critical point Cp_1 and instantiates an edge in the graph. (b) When the robot is done covering the cell between Cp_1 and Cp_2, it joins the nodes in the graph that correspond to Cp_1 and Cp_2 with an edge, and starts two new edges. (c) The robot covers the cells below the obstacle and to the right of Cp_3. (d) While covering the cell above the obstacle, the robot encounters Cp_2 again. Since all the critical points have explored edges, i.e., covered cells, the robot concludes that it has completely covered the space.

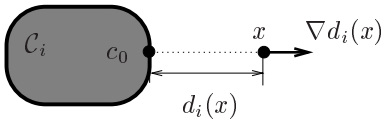


Fig. 3. The distance between x and the workspace obstacle $\mathcal{C}_i$ is the distance to the closest point c_0 on $\partial\mathcal{C}_i$. The gradient of the distance function $\nabla d_i(x)$ is the unit vector pointing away from the nearest point $c_0 \in \partial\mathcal{C}_i$.

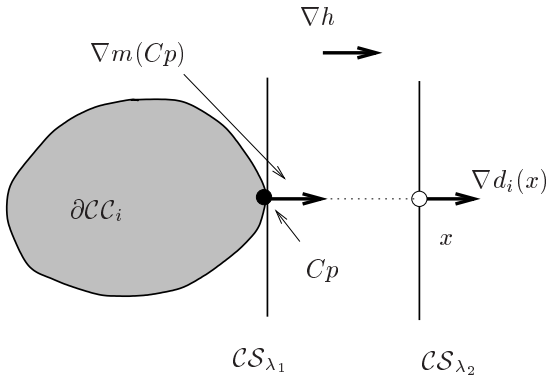


Fig. 4. At point x, $\nabla d_i(x) \perp \mathcal{CS}_{\lambda_2}$. Therefore, there exists a critical point $Cp = x - \nabla d_i(x) d_i(x)$. Note that the critical point is located on the boundary of the configuration space obstacle.

$\mathcal{CS}_{\lambda_2}$ (Figure 4). Note $d_i(x) = |\lambda_2 - \lambda_1|$. Assume that the closest configuration space obstacle $\mathcal{CC}_i$ is locally convex and smooth.

LEMMA 1. Let x be in $\mathcal{CS}_{\lambda_2}$. For all i, $\nabla d_i(x)$ is orthogonal to $\mathcal{CS}_{\lambda_2}$ if and only if there exists a critical point at $Cp = (x - \nabla d_i(x) d_i(x)) \in (\mathcal{CS}_{\lambda_1} \cap \partial\mathcal{CC}_i)$.

Proof. ($\Rightarrow$) Assume that $\nabla d_i(x)$ is orthogonal to $\mathcal{CS}_{\lambda_2}$. Consider an object boundary $\partial\mathcal{CC}_i$ with surface normals denoted as $\nabla m(x)$. Let $Cp \in \partial\mathcal{CC}_i$ be the argmin$(d_i(x))$, *i.e.* $Cp = x - \nabla d_i(x) d_i(x)$. Let Γ be a line that is tangent to $\partial\mathcal{CC}_i$ at Cp (Figure 4). By the properties of $d_i(x)$, $\nabla d_i(x)$ is orthogonal to $\partial\mathcal{CC}_i$ and hence to Γ (Clarke 1990). Since both Γ and $\mathcal{CS}_{\lambda_2}$ are orthogonal to $\nabla d_i(x)$, they are parallel to each other. Therefore Γ is the slice $\mathcal{CS}_{\lambda_1}$. Furthermore, we know that $\nabla d_i(x) \parallel \nabla m(x)$, then $\nabla m(x) \perp \mathcal{CS}_{\lambda_1}$. Therefore Cp is a critical point.

($\Leftarrow$) Assume that $Cp \in \partial\mathcal{CC}_i$ is a critical point. Consider the slice passing through Cp. Since Cp is a critical point, $\mathcal{CS}_{\lambda_1}$ should be tangent to $\partial\mathcal{CC}_i$ at Cp by Corollary 2. Let x be a point satisfying $Cp = x - \nabla d_i(x) d_i(x)$. Let $\mathcal{CS}_{\lambda_2}$ be the slice parallel to $\mathcal{CS}_{\lambda_1}$ and passing through x. Since the minimum distance between two parallel lines is the length of the line perpendicular to them and since $\partial\mathcal{CC}_i$ is locally convex, $|Cp - x|$ is the minimum distance between Cp and the slice $\mathcal{CS}_{\lambda_2}$. Hence, $x \in \mathcal{CS}_{\lambda_2}$ is the closest point to $\partial\mathcal{CC}_i$ and distance function $d_i(x)$ has an extremum. When the projection of $\nabla d_i(x)$ onto $\mathcal{CS}_{\lambda_2}$ is equal to zero, $d_i(x)$ has an extremum (Clarke 1990), then $\nabla d_i(x)$ is orthogonal to $\mathcal{CS}_{\lambda_2}$. □

Note that if the boundary is non-smooth, we use the generalized gradients.

The coverage strategy that we choose mainly guides the robot along slices and boundaries of the obstacles. We call

the motion along the slices as lapping and the motion along the obstacle boundaries as wall following. The result of the Lemma 1 directly applies to both cases. If the robot senses a critical point while it is lapping, the critical point is located in the boundary of the configuration space obstacle, but not on the slice being followed. If the robot senses a critical point while it is wall following, the critical point is located at a position that corresponds to the center of the robot (Figure 5).

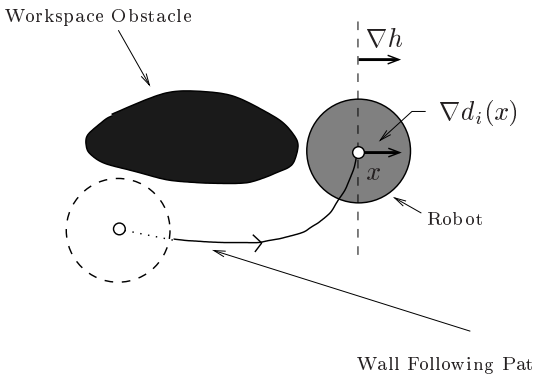


Fig. 5. The robot senses the critical point Cp while it is following the boundary of the obstacle at a safety distance, it locates Cp at x on the boundary of the configuration space obstacle.

3.2. Encountering All Critical Points

Now that we have shown how to sense the critical points using range measurements, we supply an algorithm that guides the robot along a coverage path, and guarantees that it will encounter all the critical points in an unknown space. In unknown spaces, the robot incrementally constructs the Reeb graph while it is performing coverage. Since the Reeb graph is connected, exhaustive construction ensures complete coverage because the robot will have visited every critical point and their related cells. Since covering each cell is trivial, the main challenge to incremental construction of the Reeb graph is encountering the critical point that indicates completion of the cell being covered, termed the current cell. We term this critical point the *closing critical point* of the current cell. Note that for each new cell, the robot looks for the new cell's closing critical point.

3.2.1. Problems with Conventional Coverage

Conventional coverage algorithms used by many researchers (Cao et al. 1988; Hert et al. 1996; Hofner and Schmidt 1995; Lumelsky et al. 1990) can miss the closing critical point or have a problem equivalent to missing the closing critical point because they perform the bulk of their coverage using a raster-scan-like type of motion: lap to an obstacle, follow the obstacle boundary for a lateral distance equal to inter-lap spacing, and repeat. This raster-scan approach alternates wall following between the "ceiling"[4] and the "floor" of the cell, as shown in Figure 6. However, since the robot does not follow the boundary of the ceiling or floor, it cannot sense the critical points in the ceiling or floor using the critical point sensing method described in Section 3.1, even with an unlimited sensor range. In the next section we introduce an algorithm that solves this problem.

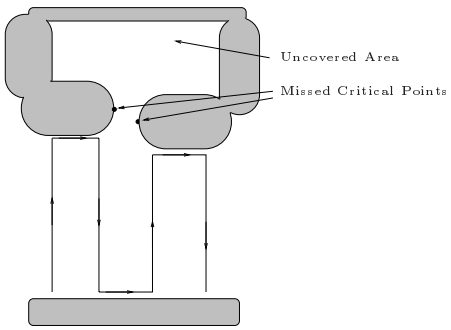


Fig. 6. Critical points in the ceiling are missed with conventional coverage algorithms.

3.2.2. Cycle Algorithm

The cycle algorithm is the most important sub-algorithm of our coverage path planner because it guides the robot to look for the closing critical point in three phases. Our algorithm not only improves the raster-scan-like motions to handle more general classes of spaces but also achieves complete coverage even with a limited range sensor that can be used to guide the robot along the obstacle boundaries.

We assume that the lap-width, the lateral distance between two consecutive laps, is equal to the robot's diameter. While the robot is executing the cycle algorithm, it consecutively follows lapping and wall following paths. A wall following path, $w : [0, 1] \rightarrow \partial\mathcal{FCS}$, is implicitly defined as $\dot{w}(t) = [\nabla d_i(w(t))]^{\perp}$ where $w(t) \in \partial\mathcal{CC}_i$, and a lapping path is $l : [0, 1] \rightarrow \mathcal{FCS}_{\lambda}$ such that $l(0) \in \partial\mathcal{FCS}$ and $l(1) \in \partial\mathcal{FCS}$. We assume without loss of generality that the robot starts the cycle algorithm at $S_i \in \partial\mathcal{FCS}^j_{\lambda_i}$ in the ceiling of the current cell where $\mathcal{FCS}^j_{\lambda_i}$ is an interval. The sweep direction $\nabla h(x)$ is referred to as forward and λ increases in this direction. Any lapping path followed in a direction from ceiling to floor is referred to as forward lapping. From S_i the robot looks for critical points via the following phases (Figure 7(a)).

4. 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.

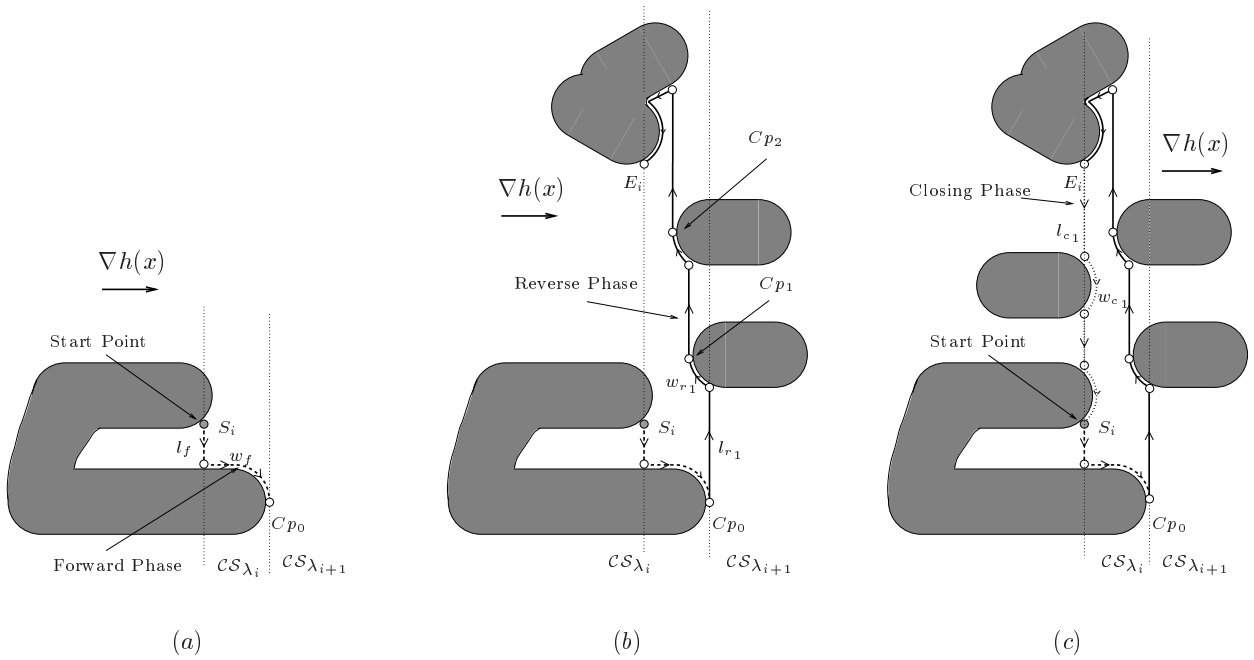


Fig. 7. (a) The robot follows the dashed path between points S_i and Cp_0 while it is executing the forward phase. (b) While the robot is performing the reverse phase, it follows the solid path between points Cp_0 and E_i. (c) During the closing phase, the robot follows the dotted path between points E_i and S_i.

1. ***Forward phase:*** The robot follows a forward lapping path l_f starting at $S_i = l_f(0)$ along $\mathcal{FCS}^j_{\lambda_i}$ toward the floor. Then it follows the wall along the path, w_f, in the forward direction, i.e., $w_f : [0, 1] \rightarrow \partial\mathcal{CC}_i$ such that $\langle \dot{w}_f(t), \nabla h(w_f(t)) \rangle > 0, \ \forall t \in [0, 1]$.

   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. Note that $w_f(1) \in \mathcal{FCS}_{\lambda_{i+1}}$ and $|\lambda_{i+1} - \lambda_i| \leq 2r$. In Figure 7$(a)$, the robot follows the dashed path $l_f \bullet w_f$[5] between points S_i and Cp_0 in the forward phase.

2. ***Reverse phase:*** The reverse phase is an interleaved sequence of M reverse laps, l_{r_n} (toward the ceiling), and M reverse wall following paths w_{r_n} such that $l_{r_n}(1) = w_{r_n}(0), w_{r_n}(1) = l_{r_{n+1}}(0)$ and $w_{r_M}(1) \in \mathcal{FCS}_{\lambda_i}$. A reverse wall following path w_{r_n} initially starts in the reverse direction $-\nabla h(w_{r_n}(0))$, i.e., $\langle \dot{w}_{r_n}(0), \nabla h(w_{r_n}(0)) \rangle < 0$.

   Each reverse wall following operation terminates when the robot either

   (a) senses a critical point Cp_k with $\nabla d_i(Cp_k) = -\nabla h(Cp_k)$,[6] or

   (b) returns back to the slice that contains the start point, i.e., $w_{r_M}(1) \in \mathcal{FCS}_{\lambda_i}$, and thus completes the reverse phase.

   The path followed in the reverse phase generally takes the form $l_{r_1} \bullet w_{r_1} \bullet \cdots \bullet l_{r_n} \bullet w_{r_n} \bullet \cdots \bullet l_{r_M} \bullet w_{r_M}$, where $M \geq 1$. In Figure 7(b), the solid path between points Cp_0 and E_i with $M = 3$ is followed by the robot in 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 P lapping paths l_{c_n}, along the slice $\mathcal{CS}_{\lambda_i}$, inter-mixed with P wall following paths w_{c_n} such that $l_{c_1}(0) = w_{r_M}(1)$, $l_{c_n}(1) = w_{c_n}(0)$, $w_{c_n}(1) = l_{c_{n+1}}(0)$, and $w_{c_P}(1) = S_i$. Closing wall following paths initially start in the forward direction, i.e., $\langle \dot{w}_{c_n}(0), \nabla h(w_{c_n}(0)) \rangle > 0$, and terminate when the robot returns to $\mathcal{FCS}_{\lambda_i}$. Note that the end and start points of each wall following motion lie in different intervals that are subsets of the same slice, i.e., if $w_{c_n}(0) \in \partial\mathcal{FCS}^{q+1}_{\lambda_i}$ then $w_{c_n}(1) \in \partial\mathcal{FCS}^{q}_{\lambda_i}$.

   The path followed in the closing phase generally takes the form $l_{c_1} \bullet w_{c_1} \bullet \cdots \bullet l_{c_n} \bullet w_{c_n} \bullet \cdots \bullet l_{c_P} \bullet w_{c_P}$, where $P \geq 0$. In Figure 7(c), the robot follows the dotted path between points E_i and S_i with $P = 2$ in the closing phase.

5. $f \bullet g$ is the concatenation of paths f and g (Latombe 1991).
6. Note that if $\nabla d_i(Cp_k) = \nabla h(Cp_k)$, the robot continues to follow the boundary.

The resulting cycle path is the product of the consecutive wall following and lapping paths, $\Pi_{cycle} = l_f \bullet w_f \bullet l_{r_1} \bullet w_{r_1} \bullet \cdots \bullet l_{r_M} \bullet w_{r_M} \bullet l_{c_1} \bullet w_{c_1} \bullet \cdots \bullet l_{c_P} \bullet w_{c_P}$, where $M \geq 1$ and $P \geq 0$.

We just described the algorithm using a complicated obstacle configuration. A simple path generated by the repeated execution of the cycle algorithm for coverage in a hallway is shown in Figure 8. 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. S_2 marks 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 travelled along the previous lap. So it completes the next cycle starting with wall following as if it had started at S_2.

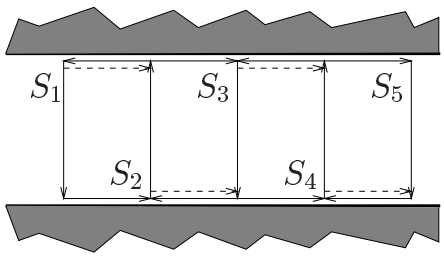


Fig. 8. S_i refers to start point of each cycle.

3.2.3. Completeness of the Cycle Algorithm

In this section, we show that, for any possible generic configuration of the obstacles, the cycle algorithm guides the robot along a coverage path that guarantees the robot will encounter the closing critical point of a cell, if it exists between consecutive laps. We show the completeness of the cycle algorithm in three steps. First, we prove that the cycle path is a simple closed path. In other words, by executing the cycle algorithm the robot always comes back to the point where it has started. Then we show that critical points can only exist along the cycle path but not in the region enclosed by the cycle path. Finally, we prove that the closing critical point of the cell being covered is always the last critical point encountered along the cycle path.

In our previous work, we introduced several slice functions to generate a variety of coverage patterns (Acar et al. 2002). In this paper, we only consider planar coverage with patterns that are generated by slice functions where $\left|\left|\nabla h(x)\right|\right|$ is constant for all x. We define the maximum distance between two slices $\mathcal{CS}_{\lambda_i}$ and $\mathcal{CS}_{\lambda_{i+1}}$, i.e., the lap-width, as

$$LW(\mathcal{CS}_{\lambda_i}, \mathcal{CS}_{\lambda_{i+1}}) = \sup_{\forall x \in \mathcal{CS}_{\lambda_i}} \left(\inf_{\forall y \in \mathcal{CS}_{\lambda_{i+1}}} |x - y| \right).$$

We choose an inter-lap spacing of $2r$ where r is the radius of the detector (essentially the robot) $LW(\mathcal{CS}_{\lambda_i}, \mathcal{CS}_{\lambda_{i+1}}) \leq 2r$. The following lemmas depend on this assumption. Without loss of generality for the remainder of this paper, we consider the slice function $h(x) = x_1$ for which the slice $\mathcal{CS}_\lambda$ is a straight line.

In Appendix A, we prove through Proposition A1 that the cycle path Π_{cycle} is a simple closed path. Since Π_{cycle} is simple and closed, by the Jordan curve theorem (Courant and Robbins 1996), we know that it partitions the free configuration space $\mathcal{FCS}$ into two disjoint sets, the inside $\mathcal{CE}$ and outside (Figure 9).

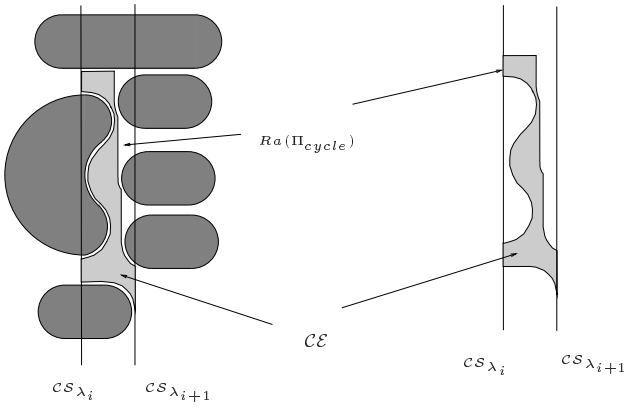


Fig. 9. The cycle path, Π_{cycle}, partitions $\mathcal{FCS}$ in two disjoint sets, the inside and the outside. $\mathcal{CE}$ denotes the inside, and it is bounded by the slices $\mathcal{CS}_{\lambda_i}$, $\mathcal{CS}_{\lambda_{i+1}}$ and the obstacles.

LEMMA 2. No obstacles can be fully contained in the interior of $\mathcal{CE}$.

Proof. Consider the first two laps l_f and l_{r_1} of the cycle path where the range of l_f is a subset of $\mathcal{CS}_{\lambda_i}$, i.e., $Ra(l_f) \subset \mathcal{CS}_{\lambda_i}$ and similarly $Ra(l_{r_1}) \subset \mathcal{CS}_{\lambda_{i+1}}$. The set $\mathcal{CE}$ lies entirely between slices $\mathcal{CS}_{\lambda_i}$ and $\mathcal{CS}_{\lambda_{i+1}}$, i.e., $\mathcal{CE} \subset \{h^{-1}(\lambda) \in \mathcal{FCS} | \lambda_i \leq \lambda \leq \lambda_{i+1}\}$. Note that $LW(\mathcal{CS}_{\lambda_i}, \mathcal{CS}_{\lambda_{i+1}}) = \lambda_{i+1} - \lambda_i \leq 2r$ (Figure 9). Hence a $2r$ diameter ball centered at $x \in \mathcal{CE}$, can not be a subset of interior of $\mathcal{CE}$, i.e., $\beta_{2r}(x) \not\subset int(\mathcal{CE})$. However, we know that for all i, there exists $x \in \mathcal{CC}_i$ such that $\beta_{2r}(x) \subset \mathcal{CC}_i$. Therefore $\mathcal{CC}_i$ can not be a subset of $\mathcal{CE}$. □

The following corollaries which immediately follow from the previous lemma state that the critical points can only lie in the range of Π_{cycle}, $Ra(\Pi_{cycle})$, but not in the interior of $\mathcal{CE}$.

COROLLARY 3. No critical point exists in the interior of $\mathcal{CE}$.

COROLLARY 4. Critical points can only lie on $\partial\mathcal{CE} = Ra(\Pi_{cycle})$.

When the robot is covering the space, most of the time, it executes the cycle algorithm without encountering any critical points. Corollaries 3 and 4 ensure that if the robot senses no critical points along the cycle path then there do not exist any other critical points inside $\mathcal{CE}$.

Now, we consider the scenarios where the robot finds a critical point (or multiple critical points) while it is executing the phases of the cycle algorithm. We introduce a lemma and its proof for each phase and then combine the results to state that the last critical point encountered along the cycle path is the closing critical point of the cell being covered. First, we consider the case when the robot encounters a critical point only in the forward phase.

LEMMA 3. If the robot senses a critical point Cp_0 only in the forward phase, then the critical point sensed is the closing critical point and it is located in the floor.

Proof. Assume that the robot starts the cycle algorithm in $\mathcal{FCS}^j_{\lambda_i}$, i.e., $\mathcal{FCS}^j_{\lambda_i}$ is path connected to the cell being covered. By Corollary 3, $\forall\lambda,\ \lambda_i \le \lambda < h(Cp_0)$, $\mathcal{FCS}^j_\lambda$ does not contain a critical point. Since Cp_0 is the only critical point along the cycle path, $\forall\lambda,\ \lambda_i \le \lambda < h(Cp_0)$, $\partial\mathcal{FCS}^j_\lambda$ does not contain a critical point either. Therefore $\{\mathcal{FCS}^j_\lambda \in \mathcal{FCS} | \lambda_i \le \lambda < h(Cp_0)\}$ is path connected to the cell. Since cells are connected components of $\mathcal{FCS} \setminus I$, Cp_0 is the closing critical point (Figure 10(a)). □

It is possible that the robot encounters critical points both in the forward and reverse phases. Then the following lemma states that the last critical point encountered along the cycle path is the closing critical point of the cell being covered. Note that the robot cannot sense more than one critical point during the forward phase.

LEMMA 4. If the robot senses zero or one critical point in the forward phase and one or more critical points during the reverse phase but no critical points in the closing phase, then the last critical point sensed during the reverse phase is the closing critical point.

Proof. Let $\{Cp_k\}$ be the set of critical points sensed during the forward and reverse phases, ordered in the order of appearance. Let $w_f(1) = Cp_0$. By Lemma A2 (Appendix A), we know that the end point of each reverse wall following motion w_{r_n} for $n < M$ is a critical point with $\nabla h(w_{r_n}(1)) = -\nabla d_i(w_{r_n}(1))$. Note that $w_{r_M}(1)$ cannot be a critical point. Let $w_{r_n}(1) = Cp_n$. Since $h(w_{r_1}(0)) = h(w_f(1))$, $h(w_f(1)) = \lambda_{i+1}$ and, by Lemma A3,

$$\lambda_{i+1} = h(Cp_0) > h(Cp_1) > h(Cp_2) > \cdots > h(Cp_{M-1}). \quad (1)$$

At this point, we proceed in two steps. First, we consider the case where no critical point is encountered during the final reverse wall following motion w_{r_M}. Then we consider the case where there exists a critical point in w_{r_M}.

When there is no critical point along w_{r_M}, from eq (1), it follows that Cp_{M-1} is the closing critical point. In other words, we can "sweep" the interval $\mathcal{FCS}^j_{\lambda_i}$ up until Cp_{M-1} without changing its connectivity, i.e., $\forall\lambda,\ \lambda_i \le \lambda < h(Cp_{M-1})$, $\partial\mathcal{FCS}^j_\lambda$ and $\mathcal{FCS}^j_\lambda$ do not contain a critical point (Figure 10(b)). Note that the robot may not sense a critical point during the forward phase. However, the result of the lemma does not change.

Now, we consider the second case where there are critical points along w_{r_M} (Figure 10(c)). Note that by Lemma A2, the robot can only encounter critical points Cp_k with $\nabla h(Cp_k) = \nabla d_i(Cp_k)$ along w_{r_M}, so that it does not terminate reverse wall following. We know that the starting point of the cycle path lies in the ceiling of the cell being covered and it is a simple closed path (Proposition A1). Therefore the robot definitely follows the ceiling while it is following the final reverse wall following path w_{r_M}. This enables us to "sweep" the interval $\mathcal{FCS}^j_\lambda$ without changing its connectivity up until the last critical point $Cp^* \in \{Cp_k\}$ sensed along w_{r_M}, i.e., $\forall\lambda \quad \lambda_i \le \lambda < h(Cp^*)$, $\partial\mathcal{FCS}^j_\lambda$ does not contain a critical point. We also know, by Corollary 3, that $\forall\lambda,\ \lambda_i \le \lambda < h(Cp^*)$, $\mathcal{FCS}^j_\lambda$ does not contain a critical point either. Therefore the subset of $\mathcal{CE} \setminus \bigcup_k \mathcal{FCS}^j_{h(Cp_k)}$ where $\mathcal{FCS}^j_{h(Cp_k)}$ is an interval that contains a critical point Cp_k in its boundary, $\{\mathcal{FCS}^j_\lambda \in \mathcal{FCS} | \lambda_i \le \lambda < h(Cp^*)\}$, is path connected to the cell. Since cells are connected components of $\mathcal{FCS} \setminus I$, Cp^* is the closing critical point. □

The robot may or may not execute the closing phase at the end of the reverse phase. If it executes, then the last critical point sensed is the closing critical point.

LEMMA 5. If the robot senses one or more critical points during the closing phase, then the last critical point sensed during the closing phase is the closing critical point irrespective of how many critical points the robot senses during the reverse and the forward phases.

Proof. Let $\{Cp_k\}$ be the set of critical points along the cycle path, ordered in the order of appearance. Consider the interval $\mathcal{FCS}^j_{h(Cp_k)}$ that contains a critical point Cp_k in its boundary. The subset of $\mathcal{CE} \setminus \bigcup_k \mathcal{FCS}^j_{h(Cp_k)}$ that forms a connected component with the cell being covered contains the closing critical point in its boundary. This subset of $\mathcal{CE}$ has boundaries of $Ra(l_f)$, $\mathcal{FCS}^j_{h(Cp^*)}$, and subsets of $Ra(w_f)$ and $Ra(w_{c_P})$. Note that Cp^* is the closing critical point (Figure 11).

We know that the starting point of the cycle path lies in the ceiling of the cell and it is a simple closed path (Proposition A1). Therefore the robot definitely follows the ceiling while it is following the final closing wall following path w_{c_P}. This enables us to "sweep" the interval $\mathcal{FCS}^j_\lambda$ without changing its connectivity up until the last critical point $Cp^* \in \{Cp_k\}$ sensed along w_{c_P}, i.e., $\forall\lambda,\ \lambda_i \le \lambda < h(Cp^*)$, $\partial\mathcal{FCS}^j_\lambda$ does not contain a critical point. We also know that, by Corollary 3, $\forall\lambda,\ \lambda_i \le \lambda < h(Cp^*)$, $\mathcal{FCS}^j_\lambda$ does not contain a critical point either. Therefore the subset of $\mathcal{CE} \setminus \bigcup_k \mathcal{FCS}^j_{h(Cp_k)}$, $\{\mathcal{FCS}^j_\lambda \in \mathcal{FCS} | \lambda_i \le \lambda < h(Cp^*)\}$, is path connected to the

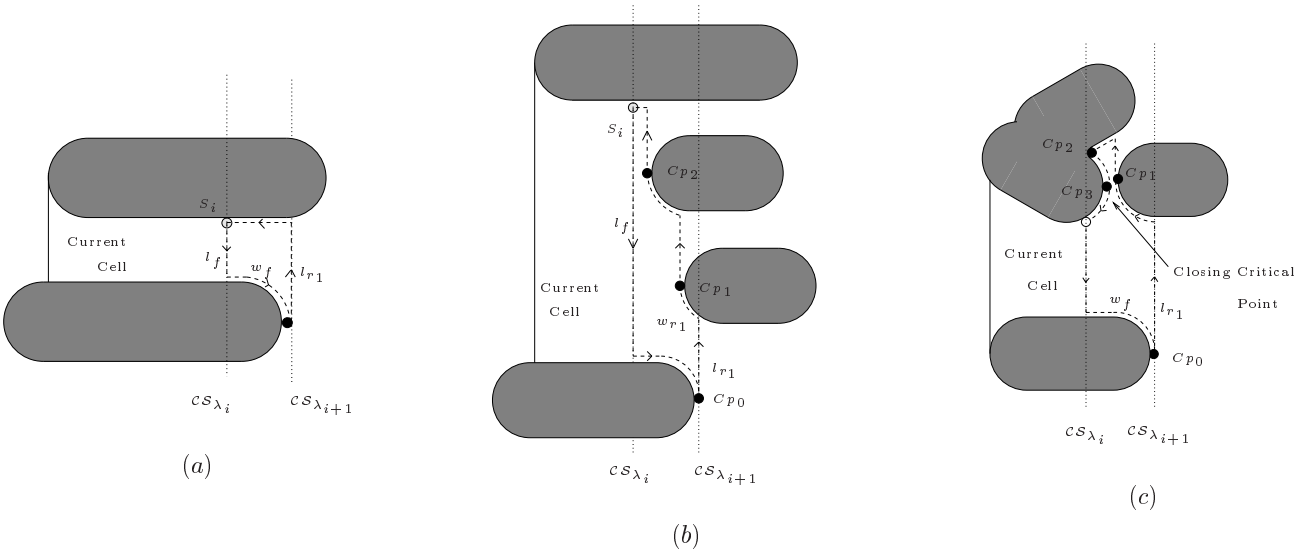


Fig. 10. The closing critical point is sensed during the forward phase in (a), reverse phase in (b), and along the final reverse wall following path w_{r_M} in (c).

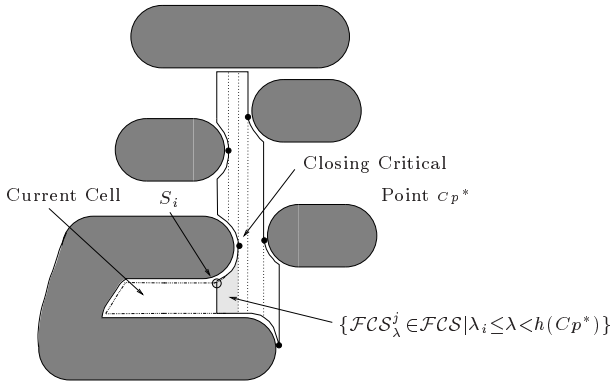


Fig. 11. The intervals that contain a critical point partitions $\mathcal{CE}$ into connected subsets. The closing critical point Cp^* lies in the ceiling and the interval which contains it lies in the boundary of the cell.

cell. Since cells are connected components of $\mathcal{FCS} \setminus I$, Cp^* is the closing critical point. □

Finally, we combine the results of the previous three lemmas to state the following proposition.

PROPOSITION 1. The last critical point encountered along the cycle path is the closing critical point.

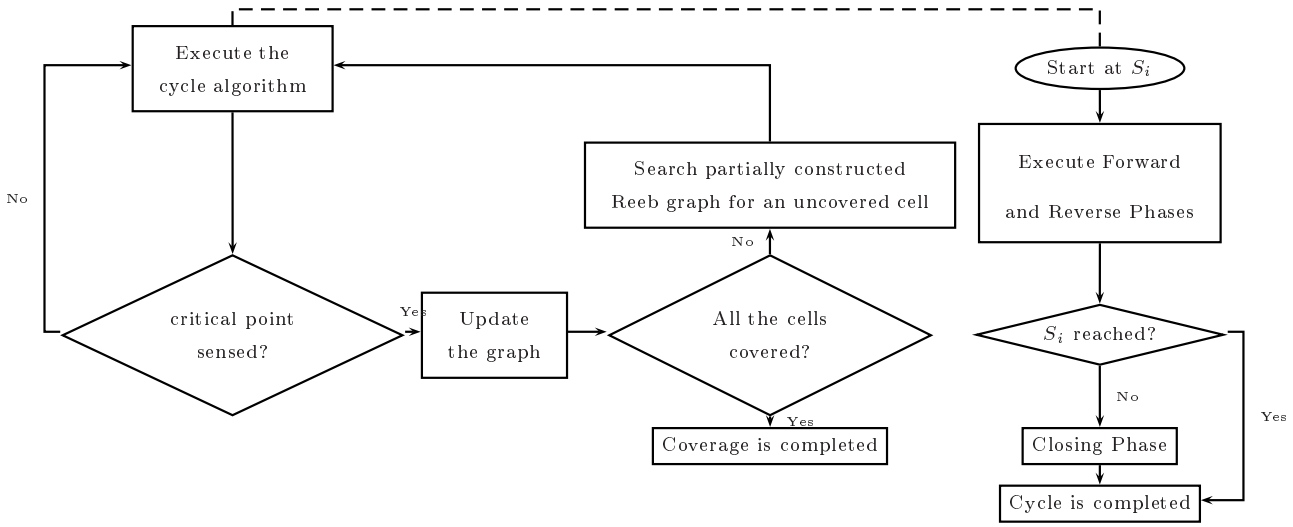
Proof. The proof immediately follows from Lemmas 3, 4, and 5. □

3.2.4. Coverage Algorithm

The cycle algorithm constitutes a major part of the coverage algorithm. Most of the time, the robot executes the cycle algorithm repeatedly without encountering critical points. When the robot encounters critical points along a cycle path, it uses the last encountered critical point Cp_F (Proposition 1) as the closing critical point and updates the Reeb graph (Flow Chart 1). Note that if the robot encounters multiple critical points along a cycle path, cells identified by the encountered critical points are formed in the area enclosed by the cycle path to avoid multiple coverage of the same area (Figure 11).

If the critical point Cp_F has not been discovered before, it can have either zero or two associated uncovered cells. If there are uncovered cells associated with Cp_F, the robot picks one of the cells and covers the cell until it finds the closing critical point of the cell. If there are no uncovered cells associated with Cp_F, the planner performs a depth-first search on the Reeb graph to determine an already discovered critical point with an uncovered cell, if any are left.

For backtracking to an already discovered critical point, the planner plans a path that passes through the covered cells and their associated critical points. We could determine a path with the Bug2 (Lumelsky and Stepanov 1987) algorithm, but using the structure particular to our method, we can set a "better" path. We modified the Bug2 algorithm where the Bug2 path is broken into a number of pieces, one piece for each covered cell. Each path piece is then "retracted" to the boundary of obstacles (either the ceiling or floor of a covered cell) and the pieces are connected via path segments along slices



Flow Chart 1. Overview flow chart of the coverage algorithm.

that contain critical points. The advantage of this method is that most of the traveling time the robot can servo off of the obstacle boundaries as opposed to following the straight line that connects the start and goal positions. See Figure 12 for a depiction of an example coverage run.

4. Complexity of the Algorithm

We formulate the complexity of the algorithm in terms of number of critical points and obstacles, length of the perimeter of the obstacles and the dimensions $W \times Z$ of a bounding rectangle that fully contains the space. The bounding rectangle is oriented such that one side of the rectangle (with length W) is parallel to the slice direction. First, we establish a relationship between the number of cells, critical points and obstacles that are encoded in the Reeb graph. Note that obstacles (including the outer boundary) are represented with faces[7] in the graph. Graph theory uses *Euler's formula* to relate the number of nodes v, edges e and faces f of a planar connected graph by $v - e + f = 2$ (Bondy and Murty 2000). The nodes of the Reeb graph correspond to critical points, its edges represent the cells, and its faces depict the obstacles. Moreover the Reeb graph is connected and planar. Therefore we can use Euler's formula with one modification. Since the outer boundary of the space is, in general, not termed an obstacle, we subtract one from the number of faces to get the number of obstacles. Let N_{cp} be the number of critical points, N_{ce} be the number of cells and N_{ob} be the number of obstacles (Figure 13), then $N_{ce} = N_{cp} + N_{ob} - 1$. This formula tells us that the number of cells increases *linearly* as the robot discovers new critical points.

Next, we calculate an upper bound on the total coverage path length. To simplify the calculation, we analyze lapping, wall following, and backtracking motions separately. Since the space is fully contained within a bounding rectangle, the length of each lapping path can be at the most Z. There must be at least $W/(2r)$, rounding to the closest integer, lapping paths where $2r$ is the inter-lap spacing. However, when the robot starts to cover an uncovered cell, it performs an extra lap starting from the critical point that indicates the uncovered cell (Figure 14). Therefore, there are N_{ce} number of extra lapping paths. Then the total number of lapping paths is $\frac{W}{2r} + N_{ce}$. Since the length of each lapping path is bounded above by Z, the total path length of lapping motions is bounded above by $\frac{WZ}{2r} + ZN_{ce}$. Note that this value is proportional to the area of the enclosing rectangle $W \times Z$ and the number of cells.

Now we analyze the length of wall following paths. Let P_{cell} be the length of the floor and ceiling of a cell. The coverage algorithm guarantees that the robot follows the entire floor and ceiling of a cell along the obstacle boundaries. Therefore, the length of wall following paths in a cell should be at least P_{cell}. However, the robot, for each cycle, performs an undo-reverse wall following motion. Hence, the lower bound is $P_{cell} + P_{cell}/2$. In the worst case, the upper bound becomes $2P_{cell}$ (Figure 15). Then, the total length of the wall following paths should be less than $2\sum_{i=1}^{N_{ce}} P_{cell_i}$ or $2P_{total}$ where P_{total} is the length of the perimeter of the obstacles and outer boundary.

After discovering the closing critical point of a cell, the robot backtracks to the closing critical point by wall following and, if necessary, lapping (Figure 16). In the worst case, the length of this backtracking path is $P_{cell} + Z$. When we consider

7. A plane graph partitions the space into connected regions. Closures of these regions are called faces (Bondy and Murty 2000).

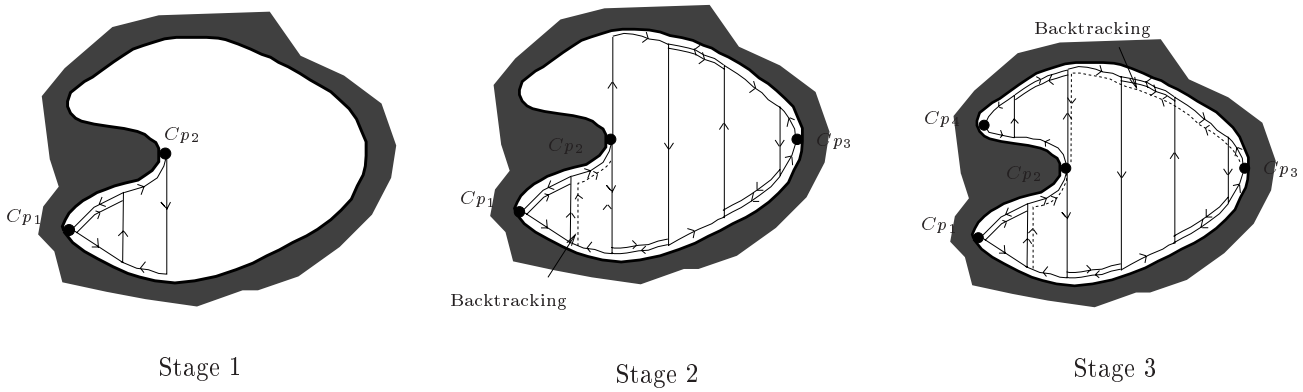


Fig. 12. The robot starts to cover the space from Cp_1 and it first encounters Cp_2. At the end of the cycle, after discovering Cp_2, the robot backtracks to Cp_2 by wall following and lapping. Then the robot starts to cover the cell located at the right of Cp_2. When the robot encounters Cp_3, it finishes covering the cell and starts to follow the ceiling of the cell until the robot reaches the slice that contains Cp_2. Then the robot laps along the slice to reach Cp_2. Finally the robot covers the uncovered cell at the left of Cp_2.

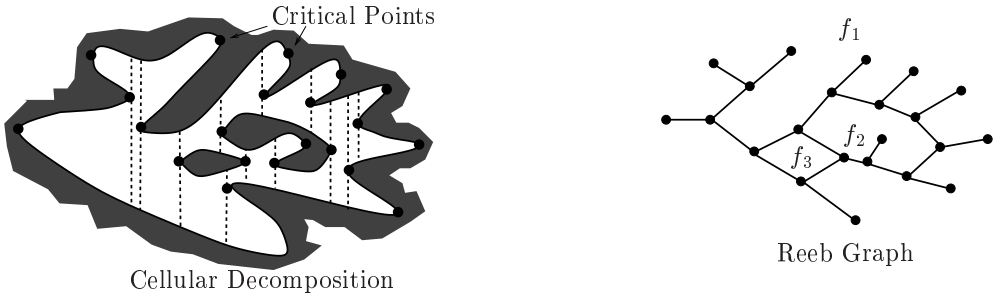


Fig. 13. In this example decomposition, there are twenty one critical points (nodes), $N_{cp} = 21$ and two obstacles (faces f_2, f_3 in the graph, f_1 is the outer boundary) $N_{ob} = 2$. Using the modified Euler's formula $N_{ce} = N_{cp} + N_{ob} - 1$, there must be twenty two cells (edges), $N_{ce} = 22$.

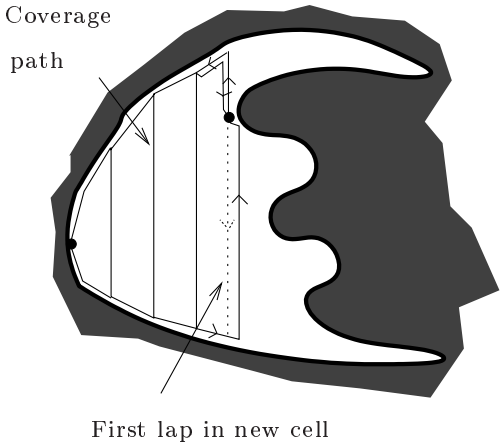


Fig. 14. When the robot starts to cover a new cell, it performs an extra lap starting from the critical point that indicates the new cell.

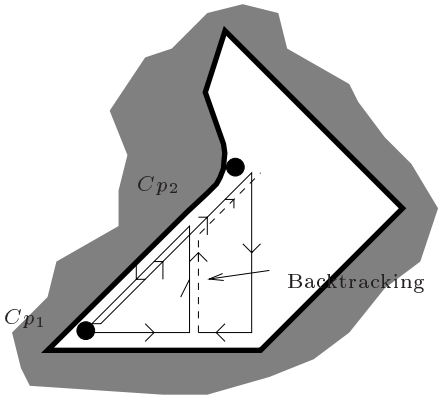


Fig. 16. The robot starts to cover the space from Cp_1. After sensing Cp_2, at the end of the cycle the robot backtracks to Cp_2 by lapping and wall following. Therefore the length of backtracking path is bounded above by $P_{cell} + Z$ for each cell.

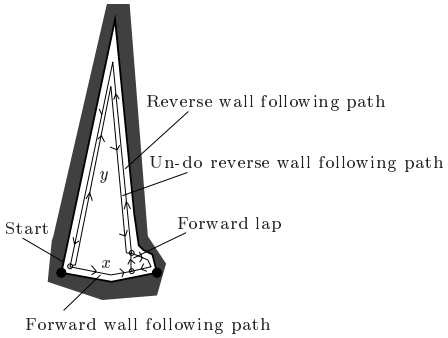


Fig. 15. The total perimeter of the cell is equal to $x + y$ where y is the length of the floor and x is the length of the ceiling and $x >> y$. The total path length traveled along the boundary of the cell is bounded above by $2x + y$. In the worst case, as x gets much larger than y, this value is equal to $2(x + y)$.

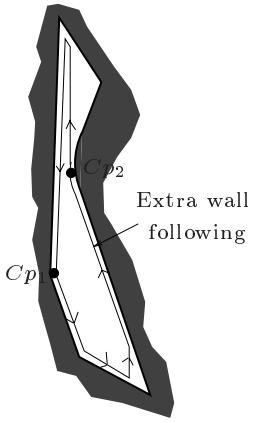


Fig. 17. The robot starts to cover the space from Cp_1. Along the first cycle path, it discovers the critical point Cp_2. However, during this cycle, the robot has covered a small portion of the cell that results in an extra wall following motion. In the worst case, the length of this path is bounded by P_{cell}.

all the backtracking paths, the upper bound becomes $P_{total} + ZN_{ce}$.

The robot starts to cover an uncovered cell from one of its defining critical points. While discovering this critical point by performing the cycle algorithm, the robot covers a small portion of the uncovered cell. The extra wall following path followed by the robot to discover the critical point is bounded above by P_{cell} (Figure 17). Hence, the total extra wall following path length is bounded above by P_{total}.

When the robot finishes covering a cell, it performs a depth-first search on the Reeb graph to choose an uncovered cell, if any are left (Figure 18). The robot reaches the uncovered cell by traversing the covered cells. To traverse a covered cell, the robot performs wall following and lapping motions, as we have explained in Section 3.2.4. In the worst case, within each covered cell the total path length traveled is bounded above by the perimeter of the cell P_{cell} and two maximum length lapping paths $2Z$, i.e., $P_{cell} + 2Z$ (Figure 19). Since we perform a depth-first search on the graph, each cell is traversed once at most (Cormen et al. 1990). Therefore, the

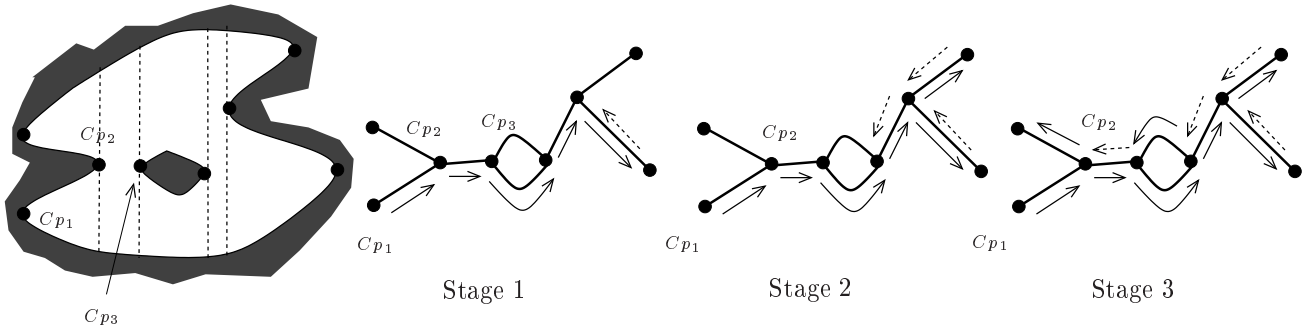


Fig. 18. The robot starts to cover the space from Cp_1. Whenever the robot finishes covering a cell, a depth-first search is performed on the graph to choose a new cell to cover. On the graph, solid arrows depict the coverage directions and dashed arrows correspond to backtracking directions. The depth-first search on the graph requires a maximum of N_{ce} backtracking.

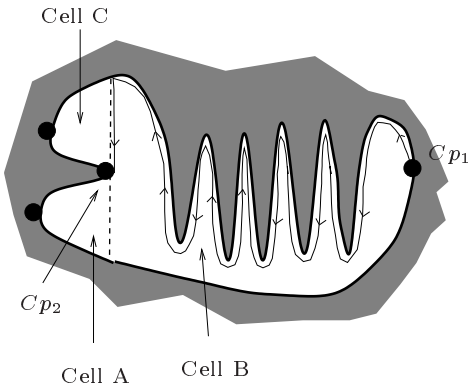


Fig. 19. After finishing covering cells A and B, the robot needs to travel from Cp_1 to Cp_2 to start to cover cell C. The robot simply follows the boundary of the obstacle either along the ceiling or floor of the cell. In the worst case, the wall following path length is bounded above by the length of the perimeter of the obstacles that form the boundary of the cell.

backtracking path length is bounded by $\sum_{i=1}^{N_{ce}} P_{cell_i} + 2N_{ce}Z$ or $P_{total} + 2N_{ce}Z$.

Combining the above upper bounds, the length of the coverage path should be less than $\frac{WZ}{2r} + 4ZN_{ce} + 5P_{total}$ or using the modified Euler's formula $\frac{WZ}{2r} + 4Z(N_{cp} + N_{ob}) + 5P_{total} - 4Z$. For an average case where the cell boundaries are not complicated, i.e., the ceiling and the floor of the cells are approximately equal in length ($x \approx y$), this bound becomes $\frac{WZ}{2r} + 4Z(N_{cp} + N_{ob}) + 2.5P_{total} - 4Z$. Therefore, the total coverage path length linearly is bounded by the area of the space, number of critical points, the length of the perimeter of the obstacles, and the outer boundary.

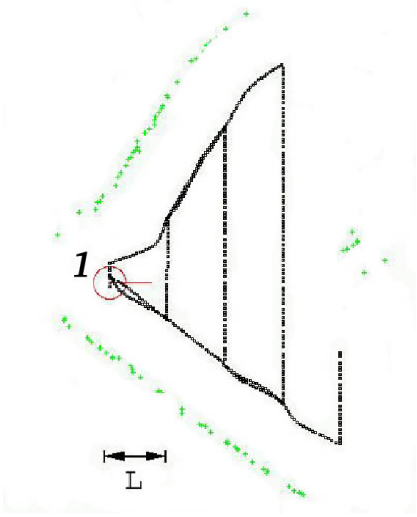
These upper bounds can be improved by keep tracking of the start and end points of the wall following motions and using these points to avoid the extra lapping motions. For example in Figure 14, the robot can start to cover the new cell by first performing wall following motion rather than lapping. Also, we can make the robot sense critical points remotely. In Figure 17, the robot could have sensed Cp_2 remotely and would not need to perform extra wall following motions. However, this would require long range sensors.

5. Experiments

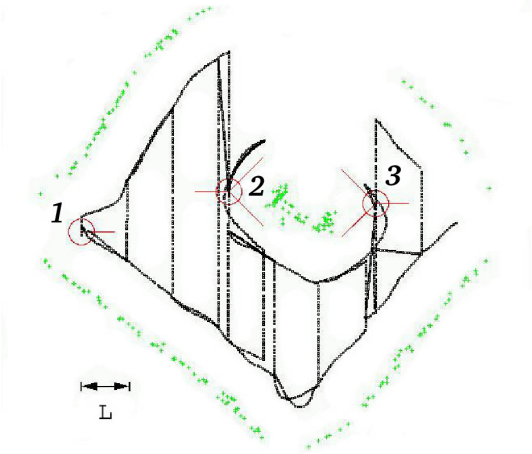
We demonstrate our critical point sensing method and the coverage algorithm by performing experiments with a Nomad 200 mobile robot (Nomad 1996) equipped with 16 ultrasonic sensors. The sonar measurements are processed using a method developed by Choset et al. (1999) that improves the angular resolution of the distance measurements. We use the processed sonar 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 performed coverage experiments in three different unknown obstacle configurations. In the first two, we placed a single obstacle in a 2.75 m $\times$ 3.65 m room, and in the second we used two obstacles in a bigger 3 m $\times$ 5.2 m room. We used an inter-lap spacing that is equal to the robot's diameter (0.53 m). Note that, since the environment is not known *a priori*, we picked an arbitrary slicing direction.

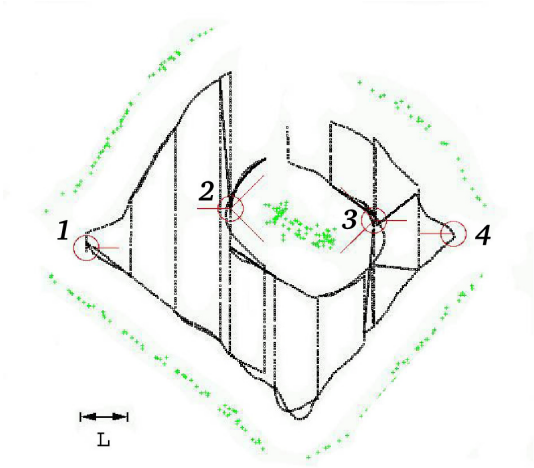
The path followed, obstacles (sonar returns), and sensed critical points during the experiments are shown in Figures 20, 21, and 22. Note that our coverage algorithm does not require storing the sensed locations of the obstacle boundaries and the path followed by the robot. To achieve complete coverage,



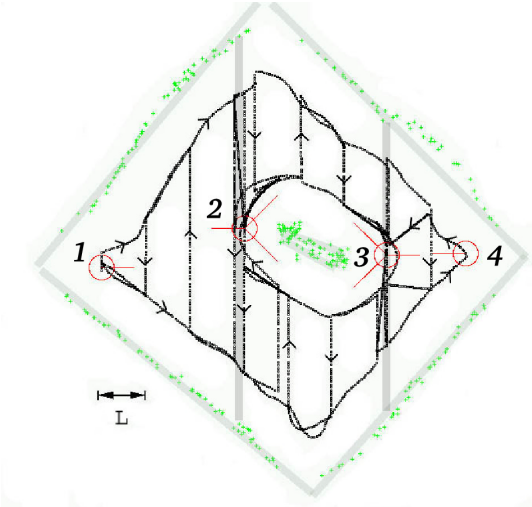
(*a*) Stage-1



(*b*) Stage-2



(*c*) Stage-3



(*d*) Stage-4

Fig. 20. Four stages of the coverage in an unknown environment. The coverage path followed by the robot is shown by dotted black lines. We depict the critical points as circles with lines emanating from them. The lines represent the directions of the corresponding adjacent cells. 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 (*d*), 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 (*d*). $L = 0.53$ m.

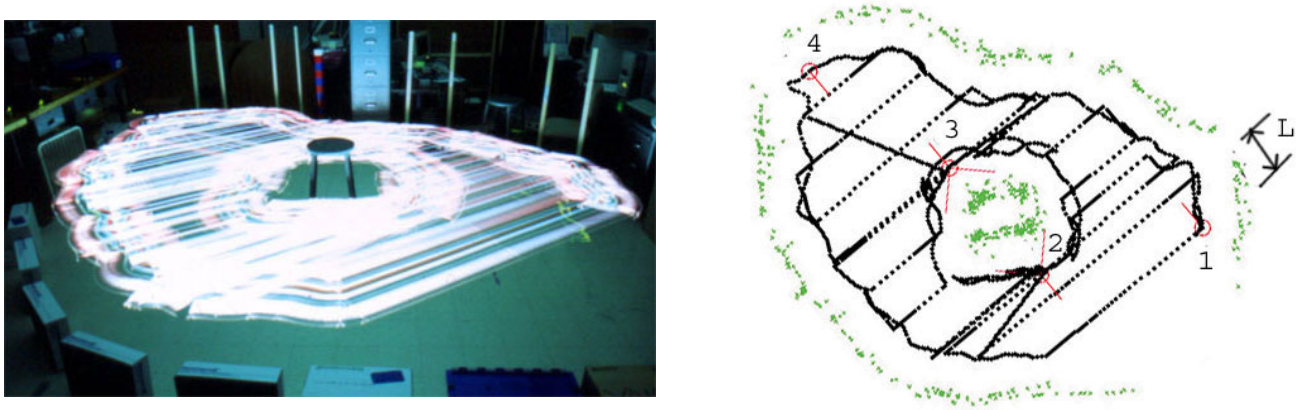


Fig. 21. The area swept by the robot is shown as a white light trace. The figure on the right shows the coverage path followed by the robot and the sensed locations of the obstacle boundaries. $L = 0.53$ m.

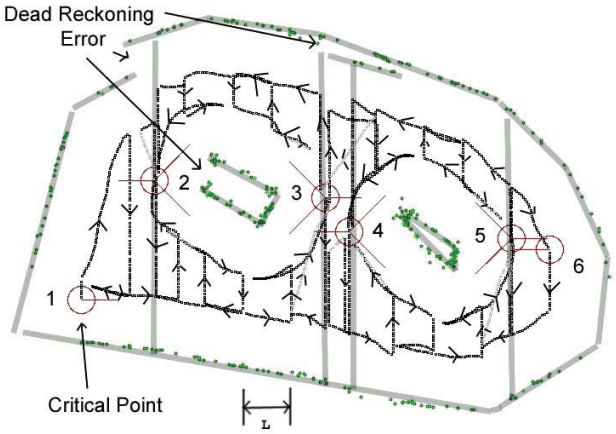


Fig. 22. The robot incrementally constructs the graph representation by sensing the critical points 1, 2, 3, 4, 5, 6, 4, 2 (in the order of appearance) while covering the *a priori* unknown space populated with two obstacles. The robot successfully covers the space. Even though this experimental run is successful, the effect of dead-reckoning is easily observed. When the robot comes back to the critical point 2, the robot's perception of the walls has not only been shifted, but also rotated. $L = 0.53$ m.

we only store the locations of the critical points and the graph representation.

5.1. Experiment 1

In the first experiment, we placed a single obstacle in a 2.75 m $\times$ 3.65 m room. Figure 20 shows the different stages of this coverage experiment. The dotted black lines represent the path traced by the robot. The vertical lines are the lapping portions of the path and the jagged-curved lines represent wall following. Note how the wall following path resembles the configuration space obstacle for the mobile robot. This makes sense because we are finding the critical points in the configuration space using workspace distance measurements.

The robot starts to cover the unknown environment at critical point 1 (Figure 20(a)). First, the robot senses critical point 2 while covering the space. After finishing the cell between critical points 1 and 2, it moves back to critical point 2.

From this point on, the robot has two new cells to explore, a lower cell and an upper cell. After performing a depth first search on the graph representation, the robot chooses the lower cell. The robot completes exploring the lower cell when it senses critical point 3 (Figure 20(b)).

Now the robot has two more new cells to cover, one to the right and another one up and to the left. Even though the upper left cell is already pointed out when the robot senses the critical point 2, it does not know that they are the same cell yet. After choosing the cell to the right, the robot senses critical point 4 and completes covering the cell between critical points 3 and 4.

At this point, the robot determines all the critical points with unexplored edges and picks the unexplored edge of critical point 3. First, it goes back to critical point 3 and then starts to cover the cell (Figure 20(c)). When the robot senses critical point 2 again, it checks the (x, y) positions of the previously

explored critical points and finds out that it has already sensed this critical point (Figure 20(d)). At this point, the robot has explored all the edges of all critical points. Therefore the robot concludes that it has found all the critical points in the unknown environment, and hence covered the whole space.

5.2. Experiment 2

We perform another experiment to show the covered area using a time-exposure photograph of the robot (Figure 21). We mount a light source on top of the robot that is as wide as the robot. In a completely dark room, we run our coverage algorithm and expose the film for the whole duration of the coverage. Since the lap-width is equal to the robot's width, the trace of the light source shows the area swept by the robot.

These two experiments demonstrate that our coverage algorithm constructs the graph representation successfully and thus achieves complete coverage of an unknown environment.

5.3. Experiment 3

We performed another experiment in a larger (3 m $\times$ 5.2 m) unknown environment in the presence of two obstacles to demonstrate the need for localization. Figure 22 shows the result of this experiment. For the sake of discussion, we outlined the boundaries of the obstacles and cells, and numbered the critical points in the order of appearance.

The robot starts to explore the space from critical point 1. It encounters critical points 2, 3, 4, 5 and 6 consecutively without problem. When it returns to critical point 3, we can observe the dead-reckoning error. The ceiling of the cell between critical points 3 and 4 is shifted upwards by approximately 17 cm and rotated. At this point, the slicing direction has also changed. The robot continues to cover the upper cell between critical points 2 and 3. When the robot senses the critical point 2 again, all edges of all critical points have been explored. Therefore, complete coverage is achieved.

In this experiment, the accumulation of dead-reckoning error is easily observed. The robot's perception of the upper wall on top of the critical point 2 has not only been shifted, approximately 25 cm, but has also been rotated. While in this experiment this has not caused any problem, in a larger environment with more obstacles dead-reckoning error would definitely cause problems. As a part of future work, we plan to address the localization problem for coverage.

6. Conclusion

We introduced a complete sensor-based coverage algorithm that is based on a Morse decomposition of a space. Morse decompositions partition the space, using critical points of Morse functions, into cells such that each cell can be covered by back and forth motions. Generically, two critical points determine each of the cells' boundaries. Therefore, complete coverage is guaranteed, once the robot encounters each of the cells' critical points. We reduced complete coverage to an incremental construction of a graph representation that encodes the topology of the Morse Decomposition. To achieve this, we developed a method to sense the critical points and an algorithm that guarantees a robot will encounter all the critical points in an unknown space.

Our critical point sensing method extends Rimon and Canny's (1994) notion of a critical point sensor. We showed that one can develop a method that uses raw sensor data to locate topologically interesting events, i.e., critical points, and use these events to design a path planner for coverage. Also, our method can use a wide range of sensing systems. In other words, any sensor system that allows the robot to follow the boundary of the obstacles is sufficient to determine the locations of the critical points. Our method requires more sensing capabilities than random or heuristic strategies. However, we believe this is the price one pays for completeness.

Prior to our work, many researchers presented coverage algorithms that use raster-scan-like motions. These algorithms will not work in all planar environments because they are designed for a limited classes of obstacle configurations. Our provably complete coverage algorithm not only works in these limited cases but also improves the conventional raster-scan-like motions to handle a more general classes of spaces where other algorithms fail. However, our algorithm generates a longer coverage path which is necessary for complete coverage. In the future, we plan to take advantage of long range sensors to shorten the path length.

We demonstrated our coverage algorithm and critical point sensing method by performing experiments using a Nomad mobile robot with 16 ultrasonic sensors in two different size rooms. In both of the rooms, our algorithm achieved complete coverage without *a priori* information. While the robot completely covered the space in the larger room, we observed the serious effect of localization error. In a much bigger room, we expect the algorithm to fail because of the dead-reckoning error. Therefore as a part of future work, we plan to develop localization algorithms that exploit the geometric features of the environment such as the critical points and structure of the cellular decomposition to minimize the dead-reckoning error.

In some applications, such as searching for people with a thermal camera, we need to consider a detector range wider than the robot's width. In this case, the robot can perform back and forth motions with a lap-width greater than the robot's diameter. While our critical point sensing method is still valid, we are going to extend our coverage algorithm to deal with this case. In this work, we restricted our coverage domain to planar surfaces and mainly focused on complete coverage of unknown spaces. However, coverage of surfaces in three dimensions for applications such as car painting and inspection need not only complete but also uniform coverage. Since it is very hard to develop algorithms for uniform coverage of unknown spaces, we plan to extend our incremental construction approach to uniform coverage of known spaces.

Appendix A: The Cycle Path

In this appendix, we prove that the cycle path Π_{cycle}, defined in Section 3.2.2, is a simple closed path. First we establish that the paths in the forward phase together with the first lapping and wall following paths in the reverse phase form a simple connected path. Then we show that along any reverse wall following path, other than the final one, there can be only one critical point where the gradients of the distance and slice functions are in opposite directions (Lemma A2). We also show that along the final reverse wall following path, the robot can only encounter critical points at which the gradients of the distance and slice functions are in the same direction. Then we use these results to show that the robot always reaches the slice in which it has started the cycle algorithm (Lemma A3 and Lemma A4). It is possible that at the end of the reverse phase, the robot may not reach the starting point of the cycle path. In this case, the robot executes the closing phase. For the closing phase, we demonstrate that the robot returns back to the starting point of the cycle path by lapping and wall following. Finally, we establish the fact that the cycle path is a simple path.

We assume that all the configuration space obstacles are compact sets with finite boundaries that are homeomorphic to a circle. Recall that the lateral distance between two consecutive laps is bounded by $2r$, i.e., $LW(\mathcal{CS}_{\lambda_i}, \mathcal{CS}_{\lambda_{i+1}}) = \lambda_{i+1} - \lambda_i \leq 2r$. We also assume that there is no more than one critical point on the same slice that defines the same cell. To simplify the notation, Cp_s denotes a critical point where the gradient vectors are in the same direction, i.e., $\nabla h(Cp_s) = \nabla d_i(Cp_s)$ and Cp_o denotes a critical point where the gradient vectors are in the opposite directions, i.e., $\nabla h(Cp_o) = -\nabla d_i(Cp_o)$.

PROPOSITION A1. The cycle path is a simple closed path, i.e., $\forall\, t_1 \neq t_2 \in (0, 1)$, $\Pi_{cycle}(t_1) \neq \Pi_{cycle}(t_2)$ and $\Pi_{cycle}(0) = \Pi_{cycle}(1)$.

Proof. First, we demonstrate that the paths in the forward phase together with the first lapping and wall following paths in the reverse phase form a simple connected path.

Without loss of generality, the robot starts to follow a forward lapping path, l_f, along $\mathcal{FCS}^j_{\lambda_i}$ from the ceiling of the current cell. Then it follows a forward wall following path, w_f in the forward direction, i.e., $l_w(1) = w_f(0)$. Let $\mathcal{FCS}_{\lambda_{i+1}}$ be the slice that contains the end point of the forward phase, i.e., $w_f(1) \in \mathcal{FCS}_{\lambda_{i+1}}$. The robot starts the reverse phase from $w_f(1)$ with reverse lapping along $\mathcal{FCS}_{\lambda_{i+1}}$, i.e., $w_f(1) = l_{r_1}(0)$. When the robot is done with the first reverse lap, it follows the first reverse wall following path, i.e., $l_{r_1}(1) = w_{r_1}(0)$. Thus the path $l_f \bullet w_f \bullet l_{r_1} \bullet w_{r_1}$ is connected.

Next, we show that the robot gets closer to and finally reaches the starting slice $\mathcal{FCS}_{\lambda_i}$ by executing reverse wall following paths. Note that, since reverse lapping takes place along a slice that is parallel to the starting slice $\mathcal{FCS}_{\lambda_i}$, we only need to focus on reverse wall following motions. We use the results of the following lemma to prove that at the end of each reverse wall following path, the robot gets to closer to $\mathcal{CS}_{\lambda_i}$.

Let M be the total number of reverse wall following paths indexed by n, i.e., $w_{r_n}, n \leq M$. We know that the robot terminates each reverse wall following motion when it encounters a critical point Cp_o or reaches the starting slice $\mathcal{FCS}_{\lambda_i}$. We will show that other than the final reverse wall following path w_{r_M}, along any w_{r_n}, $(n \neq M)$ the robot encounters only one critical point $Cp_o \in Ra(w_{r_n})$ where $h|_{\partial \mathcal{CC}_i}(Cp_o)$ can only be a local minimum. We will also show that the robot only encounters critical points Cp_o along w_{r_M}, but not Cp_s. The following lemma asserts these facts which ensure that the robot returns to the starting slice.

LEMMA A2. Let $w_{r_n}(0) \in \mathcal{CS}_\lambda$ where $\lambda_i < \lambda \leq \lambda_{i+1}$. For all $n < M$, the robot encounters one critical point $Cp_o \in Ra(w_{r_n})$ at which, (1) it terminates the reverse wall following path, (2) $h|_{\partial \mathcal{CC}_i}(Cp_o)$ is a local minimum. For $n = M$, the robot encounters critical points Cp_s along w_{r_M}, but $h|_{\partial \mathcal{CC}_i}(Cp_s)$ can be a local minimum or maximum.

Proof. We know that the robot terminates $w_{r_n} (n \neq M)$ at a critical point Cp_o. First, we prove that at the critical point Cp_o along $w_{r_n} (n \neq M)$, $h|_{\partial \mathcal{CC}_i}(Cp_o)$ is a local minimum. Assume the contrary that $h|_{\partial \mathcal{CC}_i}(Cp_o)$ is a local maximum. We know that the robot starts the reverse wall following path w_{r_n} initially in the reverse direction, i.e., $\langle \dot{w}_{r_n}(0), \nabla h(w_{r_n}(0)) \rangle < 0$ in $\mathcal{CS}_\lambda$ where $\lambda \leq \lambda_{i+1}$. Also at the starting point of w_{r_n}, the robot lies in the ceiling of the cell, i.e., $\langle \nabla d_i(w_{r_n}(0)), (\nabla h(w_{r_n}(0)))^\perp \rangle > 0$ where $(\nabla h(w_{r_n}(0)))^\perp$ is the vector orthogonal to $\nabla h(x)$ and points towards floor. Since the value of $h|_{\partial \mathcal{CC}_i}$ must increase from a local minimum before it reaches a local maximum at Cp_o, the robot must encounter a critical point $Cp^* \in Ra(w_{r_n})$ (before encountering Cp_o) where $h|_{\partial \mathcal{CC}_i}(Cp^*)$ is a local minimum. At Cp^* there are two possibilities for the directions of the gradient vectors, (1) $\nabla h(Cp^*) = -\nabla d_i(Cp^*)$ (Figure 23(a)), (2) $\nabla h(Cp^*) = \nabla d_i(Cp^*)$ (Figure 23(b)). Consider the first case. Since the robot terminates reverse wall following at a critical point Cp^* with $\nabla h(Cp^*) = -\nabla d_i(Cp^*)$ and starts to lap, it will never encounter Cp_o. Therefore $\nabla h(Cp^*) \neq -\nabla d_i(Cp^*)$.

The only possibility left is item 2. By Lemma B2, we know that if the robot encounters Cp^* with $\nabla h(Cp^*) = \nabla d_i(Cp^*)$ where $h|_{\partial \mathcal{CC}_i}(Cp^*)$ is a local minimum before sensing Cp_o, the path w_{r_n} intersects with a slice $\mathcal{CS}_{\lambda_a}$ such that $\lambda_a - \lambda_i \geq 2r$. We know that $\lambda_{i+1} - \lambda_i \leq 2r$. Hence $\lambda_a - \lambda_{i+1} > 0$. Therefore w_{r_n} should intersect the slice $\mathcal{CS}_\lambda$ in which w_{r_n} has started and at the intersection point, the robot lies in the upper boundary of an obstacle, i.e., $\langle \nabla d_i(w_{r_n}(t)), (\nabla h(w_{r_n(t)}))^\perp \rangle < 0$. However, we know that at the starting point of the w_{r_n}, the robot lies in the lower boundary of an obstacle, i.e., $\langle \nabla d_i(w_{r_n}(0)), (\nabla h(w_{r_n}(0)))^\perp \rangle > 0$. Therefore, it is impossible for the robot to reach the starting point of the w_{r_n} along

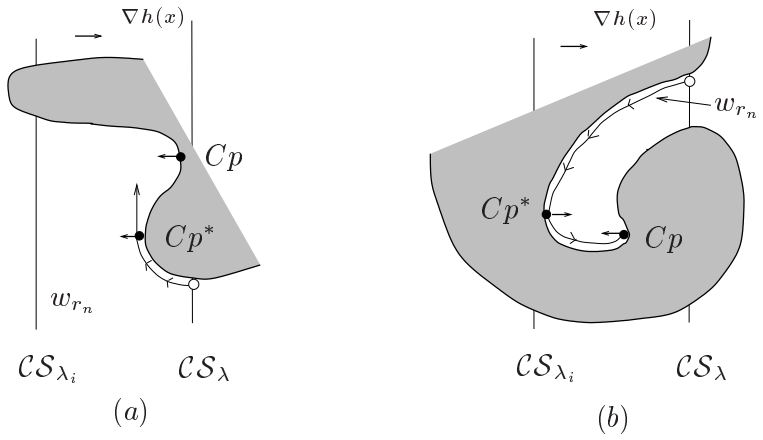


Fig. 23. The robot starts to follow the reverse wall following path w_{r_n} in the slice $\mathcal{CS}_\lambda$ where $\lambda_i < \lambda \leq \lambda_{i+1}$. The function $h|_{\partial \mathcal{CC}_i}$ takes a local maximum at Cp and a local minimum at Cp^*. (*a*) If at Cp^* gradient vectors have the opposite directions, i.e., $\nabla h(Cp^*) = -\nabla d_i(Cp^*)$, the robot terminates the reverse wall following at Cp^* and starts to lap, and never encounters Cp. (*b*) If at Cp^* gradient vectors have the same directions, i.e., $\nabla h(Cp^*) = \nabla d_i(Cp^*)$, the robot would never be able to reach the start point of w_{r_n} during the reverse phase.

$\mathcal{CS}_{\lambda_i}$ during the reverse phase. Thus at Cp^*, $\nabla d_i(Cp^*)$ and $\nabla h(Cp^*)$ cannot have the same direction either.

Since $\nabla h(Cp^*) \neq \pm \nabla d_i(Cp^*)$, Cp^* cannot exist. Hence $h|_{\partial \mathcal{CC}_i}(Cp_o)$ cannot be a local maximum and must be a local minimum.

Now we prove that there can be only one critical point Cp along w_{r_n} for all $n < M$. We know that the robot terminates wall following at a critical point Cp_o. Therefore, if there exists any other critical point Cp_a along $Ra(w_{r_n})$, the distance and slice function gradients at Cp_a should have the same directions, i.e., $\nabla d_i(Cp_a) = \nabla h(Cp_a)$. Assume that such a critical point Cp_a exists. Lemma B1 asserts that $|h(Cp) - h(Cp_a)| \geq 2r$. Because of the inter-lap spacing $(2r)$, $|h(Cp) - h(w_{r_n}(0))| \leq 2r$. Therefore the path w_{r_n} should intersect the slice $\mathcal{CS}_\lambda$ in which w_{r_n} starts and at the intersection point, the robot lies in the upper boundary of an obstacle, i.e., $\langle \nabla d_i(B(t)), (\nabla h(B(t)))^\perp \rangle < 0$. However we know that at the starting point of the w_{r_n}, the robot lies in the lower boundary of an obstacle, i.e., $\langle \nabla d_i(w_{r_n}(0)), (\nabla h(w_{r_n}(0)))^\perp \rangle > 0$. Therefore, it is impossible for the robot to reach the starting point of the w_{r_n} along $\mathcal{CS}_{\lambda_i}$ during the reverse phase. Therefore, for all $n < M$ such a critical point Cp_a with $\nabla d_i(Cp_a) = \nabla h(Cp_a)$ cannot exist in $Ra(w_{r_n})$. This contradicts the initial assumption that there exists a critical point Cp_a with $\nabla d_i(Cp_a) = \nabla h(Cp_a)$. Therefore the only possibility left is to have only one critical point Cp_o along w_{r_n} for all $n < M$.

Having shown that for $n < M$, the robot can only encounter one critical point Cp_o along w_{r_n} where $h|_{\partial \mathcal{CC}_i}(Cp_o)$ is a local minimum, now we focus on the final reverse wall following path w_{r_M}.

We show that the robot can only encounter critical points Cp_s along w_{r_M}, but not Cp_o. Assume that a critical point Cp_o exists along w_{r_M}. Since $w_{r_M}(0)$ is in $\mathcal{FCS}_\lambda$ and $w_{r_M}(0)$ lies in the lower boundary of an compact obstacle, the robot must encounter a critical point Cp^* with $\nabla h(Cp^*) = \nabla d_i(Cp^*)$ before it encounters Cp_o. However by Lemma B2, we know that if the robot encounters Cp^* with $\nabla h(Cp^*) = \nabla d_i(Cp^*)$ before sensing Cp_o, the reverse wall following path w_{r_M} intersects with a slice $\mathcal{CS}_{\lambda_a}$ such that $\lambda_a - \lambda_{i+1} > 0$ and the robot lies in the upper boundary of an obstacle, i.e., $\langle \nabla d_i(w_{r_n(t)}(t)), (\nabla h(w_{r_n(t)}))^\perp \rangle < 0$. Since the robot starts the reverse phase along $\mathcal{CS}_{\lambda_{i+1}}$ and $\lambda_a - \lambda_{i+1} > 0$, the robot would never be able to reach the starting point of the w_{r_M} during the reverse phase (Figure 23(*b*)). This contradicts the initial assumption that $Cp_o \in Ra(w_{r_M})$. Therefore the only possibility left for the directions of the distance and slice function gradient vectors is to be in the same direction at all the critical points encountered along w_{r_M}. □

We use the previous lemma to state that at the end of the each reverse wall following path w_{r_n}, the end point of the path always gets closer to $\mathcal{CS}_{\lambda_i}$ than the starting point of the path.

LEMMA A3. Let $w_{r_n}(0) \in \mathcal{CS}_{\lambda_1}$ and $w_{r_n}(1) \in \mathcal{CS}_{\lambda_2}$ $(\lambda_i < \lambda_1 \leq \lambda_{i+1}, \lambda_i < \lambda_2 < \lambda_{i+1})$. Then $\forall n < M$, $h(w_{r_n}(1)) < h(w_{r_n}(0))$, i.e., $\lambda_2 < \lambda_1$.

Proof. By Lemma A2, there can be only one critical point $Cp_o = w_{r_n}(1)$ of $h|_{\partial \mathcal{CC}_i}$ along $Ra(w_{r_n})$ and $h|_{\partial \mathcal{CC}_i}(Cp_o)$

is a local minimum. Since Cp_o is the only critical point, $h|_{\partial \mathcal{CC}_i}(Cp_o)$ is a local minimum and no other critical point exists in $Ra(w_{r_n})$, the value of $h|_{\partial \mathcal{CC}_i}$ strictly decreases along the path $Ra(w_{r_n})$ from $h(w_{r_n}(0))$ to $h(w_{r_n}(1))$. Hence $h(w_{r_n}(1)) < h(w_{r_n}(0))$ (see Figure 24). □

Having shown that the end point of each reverse wall following path w_{r_n} gets closer to $\mathcal{CS}_{\lambda_i}$, i.e. $\left|h(w_{r_n}(1)) - \lambda_i\right|$ gets smaller, now we show that the end point of the reverse phase lies in $\mathcal{CS}_{\lambda_i}$.

LEMMA A4. The end point of the reverse phase, $w_{r_M}(1)$, always lies in $\mathcal{CS}_{\lambda_i}$.

Proof. By Lemma A3, we know that $\forall n < M$, $h(w_{r_n}(1)) < h(w_{r_n}(0))$. Since $h(w_{r_{n+1}}(0)) = h(w_{r_n}(1))$ and there are finite number of obstacles,

$$\begin{aligned}\lambda_{i+1} \quad = \quad & h(w_{r_1}(0)) > h(w_{r_2}(1)) > \cdots > \\ & h(w_{r_n}(1)) > h(w_{r_{n+1}}(1)) = \lambda\end{aligned}$$

where $\lambda > \lambda_i$.

Let $h(w_{r_M}(0)) = \lambda$. We know that $\mathcal{CC}_i$ is compact and the only intersection point of w_{r_M} with $\mathcal{CS}_\lambda$ is the starting point of w_{r_M}. We also know that by Lemma A2, there does not exist a critical point Cp_o in $Ra(w_{r_M})$. Therefore the robot never terminates wall following path w_{r_M} and starts to lap. Hence the end point of w_{r_M} lies in $\mathcal{CS}_{\lambda_i}$ □

After showing that the end point of the reverse phase always lies in $\mathcal{CS}_{\lambda_i}$, we need to show that it lies in the "upper" portion of $\mathcal{FCS}^j_{\lambda_i}$ with respect to $l_f(0)$.

LEMMA A5. Let $\mathcal{FCS}^j_{\lambda_i} = Ra(l_f)$. Then $w_{r_M}(1) \in \partial \mathcal{FCS}^q_{\lambda_i}$ where $q \geq j$.

Proof. We know that w_f is a simple path with $w_f(1) \in \mathcal{FCS}_{\lambda_{i+1}}$ and thus the starting point of the reverse phase lies in the slice $\mathcal{FCS}_{\lambda_{i+1}}$, i.e., $l_{r_1}(0) \in \mathcal{FCS}_{\lambda_{i+1}}$. In the reverse phase, the robot always laps in the reverse direction of forward lapping. We also know that $\forall n$, $(Ra(l_f)\backslash\{l_f(0)\}) \bigcap Ra(w_{r_n}) = \emptyset$. Therefore $w_{r_M}(1)$ can only lie in $\mathcal{FCS}^q_{\lambda_i}$ where $q \geq j$. Since wall following takes place in the boundary, $w_{r_M}(1) \in \partial \mathcal{FCS}^q_{\lambda_i}$, $q \geq j$. □

After completing the reverse phase, the robot starts the closing phase. It laps along $\mathcal{FCS}_{\lambda_i}$ towards the starting point of the cycle algorithm. If the robot encounters an obstacle $\mathcal{CC}^c_i$, it has to decide on an initial direction $\nabla h(x)$ or $-\nabla h(x)$ to follow the boundary. Let $\mathcal{SL}$ be the set of points that lie in between $\mathcal{CS}_{\lambda_i}$ and $\mathcal{CS}_{\lambda_{i+1}}$, i.e., $\mathcal{SL} = \{x \in \mathcal{FCS} | h(x) \in [\lambda_i, \lambda_{i+1}]\}$. Since all the paths followed in the reverse phase are obstacle free, the boundary of $\mathcal{CC}^c_i$ that lies in $\mathcal{SL}$ does not intersect any path followed in the reverse phase, i.e., $\forall n$, $(\partial \mathcal{CC}^c_i \bigcap \mathcal{SL}) \bigcap Ra(w_{r_n}) = \emptyset$. Therefore the robot should start to follow the boundary of $\mathcal{CC}^c_i$ in the forward direction, i.e., $\langle \dot{w}_{c_n}(0), \nabla h(w_{c_n}(0)) \rangle > 0$ to arrive the starting slice, i.e., $w_{c_n}(1) \in \mathcal{CS}_{\lambda_i}$. Otherwise, it may never get back to $\mathcal{CS}_{\lambda_i}$. Since

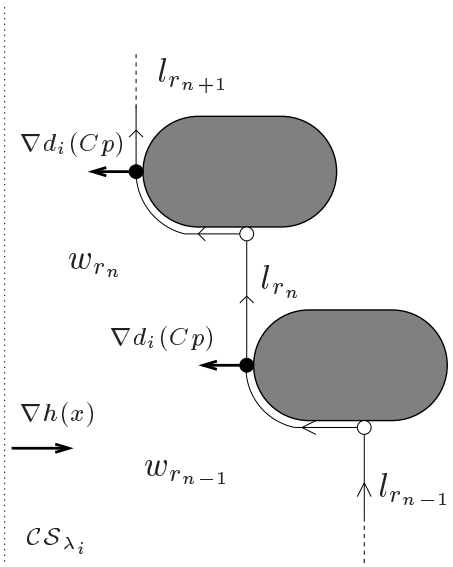


Fig. 24. The robot terminates each reverse wall following motion at a critical Cp_o where $h|_{\partial \mathcal{CC}_i}(Cp_o)$ is a local minimum. At the end of each reverse wall following motion, the robot gets closer to the starting slice $\mathcal{CS}_{\lambda_i}$.

$\mathcal{CC}_i$ is compact with finite boundary, at the end of each closing wall following motion, the robot gets closer to the starting point of the cycle algorithm. Since there are finite number of obstacles, the robot eventually arrives to the starting point of the cycle path (see Figure 25).

Finally, we establish the fact that the cycle path is a simple path. We know that all lapping paths are parallel to each other and lie in between the slices $\mathcal{CS}_{\lambda_i}$ and $\mathcal{CS}_{\lambda_{i+1}}$. Therefore, none of them can intersect each other. We also know that none of the paths followed in the closing phase can intersect the paths followed in the reverse phase. Only the starting point of the closing phase and the end point of the reverse phase are in common. By construction, the end point of the forward wall following path is the starting point of the reverse phase. Hence $Ra(w_f)$ does not intersect any of the paths. Therefore, the cycle path is simple. □

Appendix B: A Property of Configuration Space Obstacles and Relations between the Positions of the Critical Points of a Slice Function in the Configuration Space

In this section, we provide lemmas that support the proposition in Appendix A. However, these lemmas are independent results. We know that if a function on a compact set has a

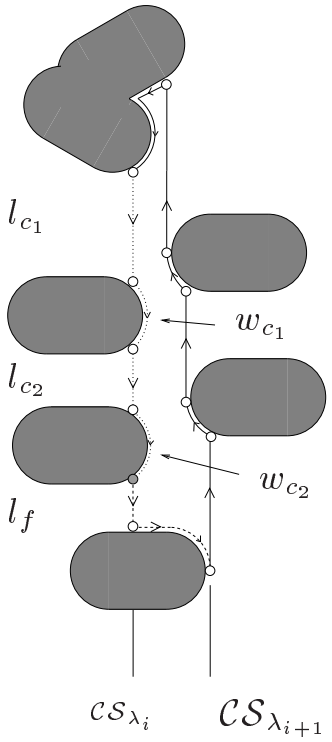


Fig. 25. The dotted path is followed by the robot during the closing phase and at the end of it, the robot gets back to the starting point.

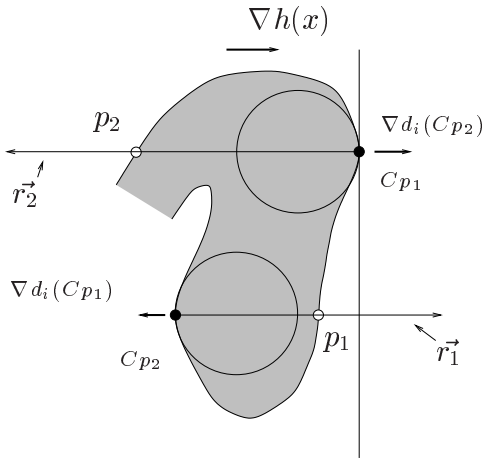


Fig. 26. The path starts and ends at the critical points Cp_1 with $\nabla h(Cp_1) = -\nabla d_i(Cp_1)$ and Cp_2 with $\nabla h(Cp_2) = \nabla d_i(Cp_2)$ respectively. The restriction of the slice function first takes a local minimum at Cp_1 and then a local maximum at Cp_2. Therefore $\left|h|_{\partial \mathcal{CC}_i}(Cp_1) - h|_{\partial \mathcal{CC}_i}(Cp_2)\right| > 2r$.

global minimum then it has a global maximum (Walter 1976). We use this fact to deduce that if the start and end points of a path along the boundary of an obstacle are the only critical points of $h|_{\partial \mathcal{CC}_i}$, then $h|_{\partial \mathcal{CC}_i}$ takes a local minimum and a local maximum or vice versa at the corresponding critical points. Note that in this work we are assuming that all critical points are non-degenerate. We have analyzed the configuration shown in Figure 26 in detail and introduce the following lemma which states that the lateral distance along the gradient of the slice function between Cp_1 and Cp_2 is always greater than $2r$.

LEMMA B1. Let $Cp_1 \in \partial \mathcal{CC}_i$ with $\nabla d_i(Cp_1) = -\nabla h(Cp_1)$ and $Cp_2 \in \partial \mathcal{CC}_i$ with $\nabla d_i(Cp_2) = \nabla h(Cp_2)$ be the only critical points of $h|_{\partial \mathcal{CC}_i}$ along the simple path $B : [0, 1] \to \partial \mathcal{CC}_i$ such that $B(0) = Cp_1$, $B(1) = Cp_2$. Let $h|_{\partial \mathcal{CC}_i}(Cp_1)$ be a local minimum and $h|_{\partial \mathcal{CC}_i}(Cp_2)$ be a local maximum. Then $\left|h|_{\partial \mathcal{CC}_i}(Cp_1) - h|_{\partial \mathcal{CC}_i}(Cp_2)\right| \geq 2r$.

Proof. We are given that along the path B, there only exist two critical points Cp_1 and Cp_2 with nothing in between them. Furthermore, $h|_{\partial \mathcal{CC}_i}(Cp_1)$ is a local minimum and $h|_{\partial \mathcal{CC}_i}(Cp_2)$ is a local maximum. Therefore the value of $h|_{\partial \mathcal{CC}_i}$ must strictly increase from $h|_{\partial \mathcal{CC}_i}(Cp_1)$ to $h|_{\partial \mathcal{CC}_i}(Cp_2)$ along B, i.e., $\forall t \in (0, 1]$, $h|_{\partial \mathcal{CC}_i}(B(t)) > h|_{\partial \mathcal{CC}_i}(Cp_1)$. Now, we establish the distance relation between Cp_1 and Cp_2.

By the properties of the configuration space, there exist two balls fully contained in $\mathcal{CC}_i$ such that Cp_1 and Cp_2 lie in the boundary of the corresponding ball (Figure 26). Let $\vec{r_1}$ be the ray that starts at Cp_1, passes through the center of the corresponding ball and extends in the ∇h direction. Similarly let $\vec{r_2}$ be the ray that starts at Cp_2, passes through the center of the corresponding ball and extends in the $-\nabla h$ direction. Since $\mathcal{CC}_i$ is a compact set, $\vec{r_1}$ and $\vec{r_2}$ intersect the boundary of $\mathcal{CC}_i$. Let p_1 and p_2 be the intersection points. Since each ball has a diameter of $2r$ and is fully contained in $\mathcal{CC}_i$, p_1 and p_2 must be at least $2r$ distance away from the respective critical point, i.e., $\left|h|_{\partial \mathcal{CC}_i}(Cp_1) - h|_{\partial \mathcal{CC}_i}(p_1)\right| \geq 2r$ and $\left|h|_{\partial \mathcal{CC}_i}(Cp_2) - h|_{\partial \mathcal{CC}_i}(p_2)\right| \geq 2r$.

We know that there exist only two simple paths between any two points on the boundary of a configuration space obstacle. Hence the path B may pass through p_1 or p_2. First, assume that the path B passes through p_1. Since $\left|h|_{\partial \mathcal{CC}_i}(p_1) - h|_{\partial \mathcal{CC}_i}(Cp_1)\right| \geq 2r$ and $h|_{\partial \mathcal{CC}_i}(p_1) \leq h|_{\partial \mathcal{CC}_i}(Cp_2)$ because $h|_{\partial \mathcal{CC}_i}$ strictly increases along B, $\left|h|_{\partial \mathcal{CC}_i}(Cp_1) - h|_{\partial \mathcal{CC}_i}(Cp_2)\right| \geq 2r$.

Second, we analyze the case where the path B passes through p_2. Assume that the lateral distance between Cp_1 and Cp_2 is less than $2r$, i.e., $\left|h|_{\partial \mathcal{CC}_i}(Cp_2) - h|_{\partial \mathcal{CC}_i}(Cp_1)\right| \leq 2r$. Since $\left|h|_{\partial \mathcal{CC}_i}(p_2) - h|_{\partial \mathcal{CC}_i}(Cp_2)\right| \geq$

$2r$ and $\left|h|_{\partial\mathcal{CC}_i}(Cp_2) - h|_{\partial\mathcal{CC}_i}(Cp_1)\right| \leq 2r$, $h|_{\partial\mathcal{CC}_i}(p_2) < h|_{\partial\mathcal{CC}_i}(Cp_1)$. Therefore, before $h|_{\partial\mathcal{CC}_i}$ takes its minimum value at Cp_1, the value of $h|_{\partial\mathcal{CC}_i}$ first increases and then decreases. Thus, there exists a critical point Cp^* with $\nabla d_i(Cp^*) = -\nabla h(Cp^*)$ along B before Cp_1. Therefore, if $\left|h|_{\partial\mathcal{CC}_i}(Cp_2) - h|_{\partial\mathcal{CC}_i}(Cp_1)\right| \leq 2r$, there exists more than one critical point along the path B. This contradicts the initial assumption that there exist only two critical points along the path B. Therefore the only possibility left is $\left|h|_{\partial\mathcal{CC}_i}(Cp_1) - h|_{\partial\mathcal{CC}_i}(Cp_2)\right| \geq 2r$. □

In the following lemma, we analyze the case that the robot encounters critical points along a path with gradient vectors both in the same and opposite directions. We assume that the robot first encounters a critical point Cp_1 of $h|_{\partial\mathcal{CC}_i}$ with $\nabla h(Cp_1) = \nabla d_i(Cp_1)$ along a path where $h|_{\partial\mathcal{CC}_i}(Cp_1)$ is a local minimum. Then it encounters another critical point Cp_2 with $\nabla h(Cp_2) = -\nabla d_i(Cp_2)$ of $h|_{\partial\mathcal{CC}_i}$ where $h|_{\partial\mathcal{CC}_i}(Cp_2)$ is a local maximum. The following lemma states that the path followed by the robot intersects with the slice $\mathcal{CS}_{\lambda_3}$ such that $\lambda_3 > h|_{\partial\mathcal{CC}_i}(Cp_2) > h|_{\partial\mathcal{CC}_i}(Cp_1)$ and $\lambda_3 - h|_{\partial\mathcal{CC}_i}(Cp_1) \geq 2r$ (see Figure 27).

LEMMA B2. Let $LW(\mathcal{CS}_{\lambda_1}, \mathcal{CS}_{\lambda_2}) < 2r$ $(\lambda_2 - \lambda_1 < 2r)$. Let $B : [0, 1] \rightarrow \partial\mathcal{CC}_i$ be a simple path such that $B(0) \in \mathcal{CS}_{\lambda_2}$. Let $B(t_1) = Cp_1$ with $\nabla h(Cp_1) = \nabla d_i(Cp_1)$ be a critical point $h|_{\partial\mathcal{CC}_i}$ such that $\lambda_1 < h|_{\partial\mathcal{CC}_i}(Cp_1) < \lambda_2$ and $h|_{\partial\mathcal{CC}_i}(Cp_1)$ is a local minimum. Let $B(t_2) = Cp_2$ be a critical point of $h|_{\partial\mathcal{CC}_i}$ with $\nabla h(Cp_2) = -\nabla d_i(Cp_2)$ such that $t_1 < t_2$ and $h|_{\partial\mathcal{CC}_i}(Cp_2)$ is a local maximum. Then $Ra(B) \bigcap \mathcal{CS}_{\lambda_3} \neq \emptyset$ such that $\lambda_3 > \lambda_2$ and $\lambda_3 - \lambda_1 \geq 2r$.

Proof. Since there exist two critical points Cp_1 with $\nabla h(Cp_1) = \nabla d_i(Cp_1)$ and Cp_2 with $\nabla h(Cp_2) = -\nabla d_i(Cp_2)$ along the path B, all the slices $\mathcal{CS}_\lambda$ $(h|_{\partial\mathcal{CC}_i}(Cp_1) < \lambda < h|_{\partial\mathcal{CC}_i}(Cp_2))$ intersect the path B. Now we show that all the slices $\mathcal{CS}_\lambda$ with $\lambda - h|_{\partial\mathcal{CC}_i}(Cp_2) \leq 2r$ also intersect the path B.

Assume that there exists a slice $\mathcal{CS}_{\lambda_3}$ such that $\lambda_3 - h|_{\partial\mathcal{CC}_i}(Cp_2) \leq 2r$ and does not intersect the path. We know that there exists a ball $\beta_{2r}(x)$ fully contained in $\mathcal{CC}_i$ such that $Cp_2 \in \partial\beta_{2r}(x)$. Since at Cp_2 the gradient vectors are in the opposite directions, i.e., $\nabla h(Cp_2) = -\nabla d_i(Cp_2)$, all the slices $\mathcal{CS}_\lambda$ with $\lambda - h|_{\partial\mathcal{CC}_i}(Cp_2) \leq 2r$ intersect the path B. Therefore the slice $\mathcal{CS}_{\lambda_3}$ (with $\lambda_3 - h|_{\partial\mathcal{CC}_i}(Cp_2) \leq 2r$) must intersect the path B. This contradicts the assumption that $\mathcal{CS}_{\lambda_3}$ does not intersect the path B. Therefore, the path B along the boundary of the obstacle CC_i at least intersects with a slice $\mathcal{CS}_{\lambda_3}$ with $\lambda_3 = h|_{\partial\mathcal{CC}_i}(Cp_2) + 2r$. Since $\lambda_2 > h|_{\partial\mathcal{CC}_i}(Cp_2)$, $\lambda_1 < h|_{\partial\mathcal{CC}_i}(Cp_1)$ and $\lambda_2 - \lambda_1 \geq 2r$, $\lambda_3 - \lambda_1 \geq 2r$. □

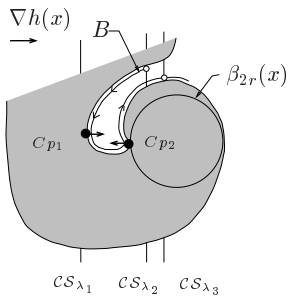


Fig. 27. The lateral distance between the slices $\mathcal{CS}_{\lambda_1}$ and $\mathcal{CS}_{\lambda_2}$ is less than $2r$. The robot first encounters critical point Cp_1 with $\nabla h(Cp_1) = \nabla d_i(Cp_1)$ where $h|_{\partial\mathcal{CC}_i}(Cp_1)$ is a local minimum and then Cp_2 with $\nabla h(Cp_2) = -\nabla d_i(Cp_2)$ where $h|_{\partial\mathcal{CC}_i}(Cp_2)$ is a local maximum. Since there exists a ball $\beta_{2r}(x)$ that has Cp_2 in its boundary and fully contained in the obstacle, the path B intersects with $\mathcal{CS}_{\lambda_3}$ where $\lambda_3 - \lambda_1 \geq 2r$ and $\lambda_3 > \lambda_2$.

Acknowledgements

We would like to thank Prasad Atkar, Dr. Alfred Rizzi, Dr. Jonathan E. Luntz, Dr. Teruko Yata, Douglas Hull and Devin Balkcom for their invaluable suggestions through out our work. The authors also 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., Choset, H., Rizzi, A. A., Atkar, P. N., and Hull, D. 2002. Morse decompositions for coverage tasks, *The Int. J. Robotics Research* 21(4):331–344.

Bondy, J. A., and Murty, U. S. R. 2000. *Graph Theory with Applications*. New York: John Wiley and Sons, Inc.

Butler, Z., Rizzi, A. A., and Hollis, R. L. 1998. Contact-sensor based coverage of rectilinear environments. In *Proc. IEEE Int. Symposium on Intelligent Control*.

Butler, Z., Rizzi, A. A., and Hollis, R. L. 2000. Cooperative coverage of rectilinear environments. In *Proc. of IEEE ICRA'00*, San Francisco, CA.

Cao, Z. L., Huang, Y., and Hall, E. 1988. Region filling operations with random obstacle avoidance for mobile robots. *J. Robotic Systems*, pp. 87–102.

Choset, H., Nagatani, K., and Lazar, N. 1999. The arc-transversal median algorithm: An approach to increasing ultrasonic sensor accuracy. In *Proc. IEEE Int. Conf. on Robotics and Automation*, Detroit.

Choset, H., and Pignon, P. 1997. Coverage path planning: The boustrophedon decomposition. In *Proceedings of the International Conference on Field and Service Robotics*, Canberra, Australia.

Clarke, F. H. 1990. *Optimization and nonsmooth analysis*. Philadelphia, PA: Society of Industrial and Applied Mathematics.

Consumer Reports. 2000. *Consumers Union of U.S., Inc.*, 9.

Cormen, H. T., Leiserson, C. E., and Rivest, R. L. 1990. *Introduction to Algorithms*. Cambridge, MA: MIT Press.

Courant, R., and Robbins, H. 1996. *What is Mathematics?* New York: Oxford University Press, Inc.

Fomenko, A. T., and Kunii, T. L. 1997. *Topological Modeling for Visualization*. Tokyo: Springer-Verlag.

Gabriely, Y., and Rimon, E. 2001. Spanning-tree based coverage of continous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence* 31:77–98.

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

Healey, A. J., McMillan, S., Jenkins, D., and McGhee, R. B. 1995. BUGS: Basic UXO gathering system. In *Proc. of the Autonomous Vehicles in Mine Countermeasures Symposium*, 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.

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

Huang, W.H. 2001. Optimal line-sweep-based decompositions for coverage algorithms. *IEEE Int'l. Conf. on Robotics and Automation*, Seoul,Korea, 1:27–32.

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

Lumelsky, V. and Stepanov, A. 1987. Path planning strategies for point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430.

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

MacKenzie, D., and Balch, T. 1996. Making a clean sweep: Behavior based vacuuming. *AAAI Fall Symposium, Instationating Real-World Agents*.

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

O'Rourke, J. 1998. *Computational Geometry in C*. Mountain View, CA: Cambridge University Press.

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

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.

Rekleitis, I., Dudek, G., and Milios, E. 1997. Multi-robot exploration of an unknown environment, efficiently reducing the odometry error. *Int. Joint Conference in Artificial Intelligence*, vol. 2, pp. 1340–1345. Nagoya, Japan: Morgan Kaufmann Publishers, Inc.

Rimon, E., and Canny, J. F. 1994. Construction of C-space roadmaps using local sensory data—What should the sensors look for? In *Proc. IEEE Int. Conf. on Robotics and Automation*, San Diego, CA, pp. 117–124.

Spires, V. S., and Goldsmith, S. Y. 1998. Exhaustive geographic search with mobile robots along space-filling curves. In *Proc. of First International Workshop on Collective Robotics*, Paris, France, pp. 1–12.

VanderHeide, J., and Rao, N.S.V. 1995. Terrain coverage of an unknown room by an autonomous mobile robot. No. ORNL/TM-13117, Oak Ridge National Laboratory, Oak Ridge, Tennessee.

Wagner, I. A., and Bruckstein A. M. 1995. Cooperative cleaners: A study in ant-robotics. *Center for Intelligent Systems*, Technion, Haifa.

Walter, R. *Principles of Mathematical Analysis.* McGraw-Hill, Inc.

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 *Proc. of Int. Conference on Advanced Robotics*, Tokyo, Japan, pp. 533–538.