Critical Point Sensing in Unknown Environments

Ercan U. Acar Howie Choset
Carnegie Mellon University
Scaife Hall
Pittsburgh, PA15213

Abstract

Many motion planning algorithms use Morse functions to characterize the free space. Specifically, these algorithms look at the critical points of a Morse function to denote the topological changes in the free space. This paper introduces methods to sense critical points and ensure all critical points are "seen" by a coverage algorithm. Experimental results performed on a mobile robot are also presented.

1 Introduction

Many motion planning algorithms rely on critical points of a function to guarantee the completeness of their particular method. Critical points are powerful because they represent topologically meaningful events in the space. With the exception of Rimon and Canny's work [14], prior critical point-based approaches assume the location of critical points before the planning event. In this paper we propose to extend Rimon and Canny's notion of a "critical point sensor" which can be used for different motion planning strategies. Our method applies to robots with range sensors such as simple sonars and laser range finders. We develop an algorithm that guarantees to find all critical points in an unknown environment, thereby maintaining the completeness of critical point-based approaches.

Our method has the side benefit of ensuring complete coverage of unknown spaces, allowing for applications such as demining, floor cleaning, etc., all of which require a robot to pass over all possible points in an environment. We have experimentally verified the work described in this paper on a Nomad 200 Mobile robot base that has a ring of 16 ultrasonic sensors.

2 Prior Critical Point-Based Approaches

Canny's roadmap algorithm [1] is one of the classical motion planning techniques that uses critical points. Let $\mathcal{CS}$ be the configuration space of a robot (or any type of moving body). A roadmap is a connected set of one-dimensional curves that captures the connectivity of the free configuration space $\mathcal{FCS}$ of a robot. For the sake of discussion, we are going to explain his algorithm for a $\mathcal{CS}$ in $\Re^3$. A slice plane $\mathcal{CS}_\lambda$ defined by the pre-image of a real valued function, $h : \Re^3 \rightarrow \Re$, *i.e.* $\mathcal{CS}_\lambda = \{x \in \mathcal{CS} | h(x) = \lambda\}$ sweeps through $\mathcal{CS}$. Let $h(x) = x^1$ where x^1 is the first coordinate of x. Varying λ has the effect of sweeping the slice along the x^1-axis through $\mathcal{CS}$. For every slice, we pick a second direction, transversal to x^1-axis, say x^2-axis. We use extremal points along the x^2-direction of the free configuration space boundary, $\partial\mathcal{FCS}$, restricted to the slice. These extrema are points on the *silhouette curves* and this curve is traced out by sweeping the slice through configuration space (Fig. 1-(a)). These *silhouette* curves are in general not connected. The connectivity of the silhouette curves may change at the critical points of the projection function onto x^2 axis. At each critical point, the algorithm operates recursively on a two-dimensional slice-plane that contains the critical point, by sweeping a slice line through the plane. The new silhouette curves are used to connect the unconnected previously generated silhouette curves.

Critical points are also used by Canny et al [2], [3] to ensure the connectivity of another roadmap. Their algorithm, the *Opportunistic Path Planner* (OPP) traces out the local maxima of a distance function restricted to a slice, as the slice sweeps through $\mathcal{FCS}$. The traced out paths are termed *freeways*. Just like the silhouette curves, in general the set of freeways are not connected, and paths between neighboring freeways must be found. The OPP freeways are connected via *bridge curves* that are constructed on slices that pass through critical points. The union of bridge and freeway curves forms the one-dimensional roadmap. See Figure 1-(b).

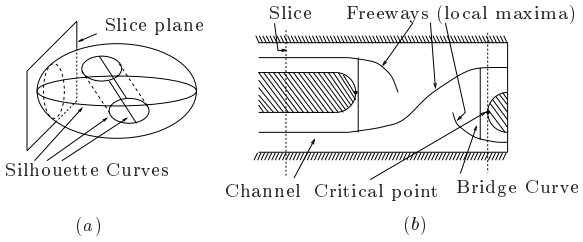

Fig. 1. (a) Silhouette curves are not connected. (b) Schematic of the OPP method.

Rimon and Canny [14] extend this work by suggesting a critical point sensor to identify the critical points and hence ensure the connectivity of the roadmap. However they do not specify the full implementation details of the critical point sensor. Their work has motivated the

work in this paper.

Cao *et al* [4] use "extrema values" to perform coverage in an unknown environment which is populated by only convex obstacles. Extrema values are the left-most and right-most points on the convex set. Their method senses the extrema values by circumnavigating the obstacles while looking for the left-most and right-most vertices. For environments with only convex obstacles, the only critical points are extrema values. The Boustrophedon Cellular Decomposition method [5] extends Cao's work to handle concave obstacles and provides a rich theoretical framework on which coverage in two or more dimensions is proven to succeed. They use critical points of a slice function to locate the boundaries of the cells where each cell can be covered by simple back and forth motions. In this paper, instead of forming an adjacency graph, we form a *dual graph* where nodes represent critical points and edges represent the cells (Figure 2).

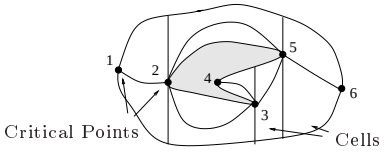

Fig. 2. Dual adjacency graph representation of the environment. Nodes represent critical points, edges represent cells.

3 Critical Point Sensing

Many robots are equipped with range sensors, such as ultrasonic rings. Since range sensors provide distance information, we use a distance function to sense critical points. The distance between a point x and an object C_i is the shortest distance between x and all points in the object. The distance function and its gradient are

$$d_i(x) = \min_{c_0 \in C_i} \|x - c_0\| \quad \text{and} \quad \nabla d_i(x) = \frac{x - c_0}{\|x - c_0\|},$$

Let the boundary M of an obstacle be defined by the pre-image of a regular value of a real valued function, $m : \Re^m \to \Re$. $\nabla m(x)$ is the surface normal. It is shown ([5]) that $\nabla m(x)$ is parallel to the gradient of a slice function $\nabla h(x)$ if and only if $x \in M$ is a critical point of $h|_M$ where $h : \Re^m \to \Re$ and $h|_M$ is the restriction of $h(x)$ to M . See Figure 3-(a).

3.1 Critical Point Sensing with a Zero Range Sensor

In this section, we describe how to sense critical points when the robot is following the boundary of an obstacle. The robot only needs to sense the location and surface normal of the boundary to sense critical points. We use a sonar ring to achieve this capability. However a high resolution tactile sensor can also be used. The following lemma states how to use distance measurements to sense critical points.

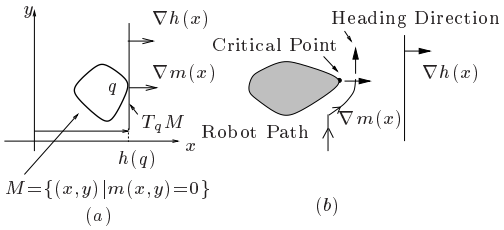

Fig. 3. (a) At the critical point q, $\nabla h(q)$ is parallel to $\nabla m(q)$.(b) During wall following, sweep direction is parallel to the gradient vector at the critical point.

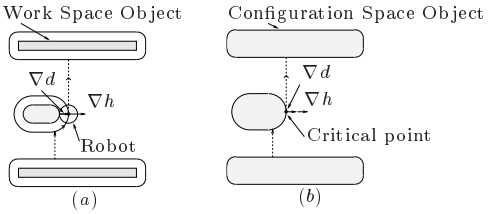

Fig. 4. (a) Robot is located at a critical point in the configuration space.(b) The objects are configuration space objects of (a) and the robot is represented by a point. It is located at a critical point. Note that $\nabla d \parallel \nabla h$.

LEMMA 3.1 *The gradient of the distance function $\nabla d_i(x)$ is parallel to the gradient of the slice function $\nabla h(x)$ if and only if $x \in M$ is a critical point of $h|_M$.*

Proof: By the properties of the distance function ([8]), $\nabla d_i(x)$ is parallel to the surface normal of the obstacle $\nabla m(x)$, *i.e.* $\nabla d_i(x) \parallel \nabla m(x)$. As shown in [5], $\nabla h(x) \parallel \nabla m(x)$ if and only if $x \in M$ is a critical point of $h|_M$. Since $\nabla d_i(x) \parallel \nabla m_i(x)$, then $\nabla h(x) \parallel \nabla d_i(x)$ when a critical point occurs. Hence sensor measurements can be used to sense critical points (Fig. 3-(b)). ■

Even though, we developed a method to sense the critical points only for smooth objects, generalized gradients [8] allow our method to work in environments populated with non-smooth objects such as polygons.

3.2 Configuration Space Versus Work Space

Our method of critical point sensing works in the configuration space, without explicitly generating the configuration space obstacles, CC_i. Consider a disk-like robot which in the configuration space is represented by a point at the center of the disk. The minimum distance between the disk and an obstacle in the free space is the same as the distance between a point and an obstacle in configuration space. Likewise, the corresponding gradients are parallel. Consider the environments shown in Figure 4. Figure 4-(b) contains the configuration space of the environment shown in Figure 4-(a) for a circular robot. The configuration, *i.e.* location of the robot in Figure 4-(a) corresponds to a critical point in the configuration space denoted in Figure 4-(b). At a critical point x, the gradients $\nabla d(x)$ and $\nabla h(x)$ are parallel to each other in both of the configuration and work spaces as

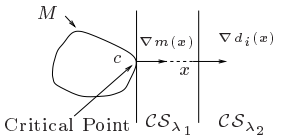

Fig. 5. The robot senses the critical point c while it is traveling along $\mathcal{CS}_{\lambda_2}$. At point x, $\nabla d_i(x) \perp \mathcal{CS}_{\lambda_2}$.

shown in Figure 4. Therefore any critical point sensed using work space distance measurements can be easily mapped to configuration space for a disk robot.

3.3 Critical Point Sensing with a Finite Range Sensor

Searching for critical points is enhanced if the robot can sense the critical points remotely. Lemma 3.1 provides a condition for the existence of critical points on slices that pass through them. The following lemma (we omit the proof due to space restrictions) supplies a condition that senses critical points from slices that do not pass through critical points. For this lemma, assume that the closest object is locally convex and smooth.

LEMMA 3.2 *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 $x - \nabla d_i(x) d_i(x)$.*

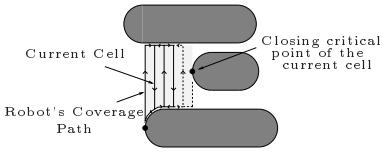

Fig. 6. While the robot is covering the current cell, it is looking for the closing critical point that identifies the completion of the current cell.

4 Guaranteeing to Encounter All Critical Points with Zero Sensor Range

Now that we have shown how to sense critical points, we need to ensure that the robot "sees" all of them to guarantee the completeness of any sensor based algorithm that uses critical points. For coverage, the robot incrementally constructs the dual adjacency graph of the cellular decomposition as it encounters critical points online (Figure 2). Coverage is complete when all cells (edges) associated with discovered critical points are covered.

Generically each cell is characterized by two critical points. Assume that the robot is covering a cell, termed the *current cell*, and thus generically has already discovered one of its defining critical points. While the robot covers the current cell, it simultaneously looks for the *closing critical point* which identifies the completion of the current cell. See Figure 6.

The challenge now is to ensure that the robot finds the closing critical point of the cell. Most conventional coverage algorithms ([7], [10], [11], [15]) perform the bulk of their coverage using a raster scan type of motion: lap to an obstacle, follow the obstacle boundary for a lateral distance equal to inter-lap spacing, and repeat. This alternates wall following between the "ceiling" and "floor" of the cell as shown in Figure 7$-a$. Unfortunately, this raster scan approach can miss closing critical points. In Figure 7$-a$, since the robot did not follow the boundary of the ceiling, it cannot sense the ceiling critical points using the critical point sensing method described in Section 3 even with the unlimited sensor range. Therefore, the robot must perform wall following along the ceiling in the reverse direction so that it will sense the critical points related to the ceiling. We call this motion *reverse wall following*. In fact, for zero or limited range sensors, reverse wall following motion is still not sufficient. As can be seen in Figure 7$-b$, the robot must undergo additional motion to detect the closing critical.

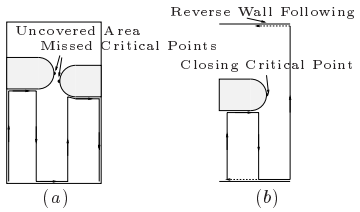

Fig. 7. (a) Critical points in the ceiling are missed with conventional coverage algorithms. (b) At the end of the reverse wall following, the robot has not sensed the closing critical point. (Obstacles are configuration space obstacles).

The algorithm described in this section addresses the issue of finding the closing critical point, which can lie in the ceiling or floor of the current cell, or interior of an interval (Figure 8). An *interval*[1] $\mathcal{FCS}_\lambda^j$ is a connected component of the *free configuration space slice* $\mathcal{FCS}_\lambda$ which is the portion of the slice $\mathcal{CS}_\lambda$ contained in the free configuration space. Note that $\mathcal{FCS} = \bigcup_\lambda \mathcal{FCS}_\lambda$ and $\mathcal{FCS}_\lambda = \bigcup_j \mathcal{FCS}_\lambda^j$.

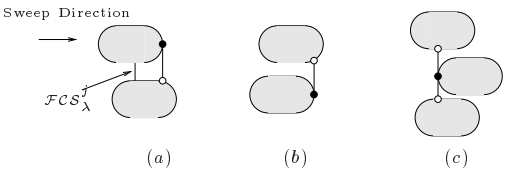

Fig. 8. Closing critical point, shown as a black dot, can be in the ceiling (a), floor (b) or interior (c) of the interval.

[1]Canny and Lin [2] term intervals as channel slices

4.1 Cycle Algorithm

We have designed an algorithm, termed the *cycle algorithm* to enable the robot to look for the closing critical point while it is performing coverage.

We will demonstrate that while the robot is executing the cycle algorithm, it consecutively follows wall following and lapping 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 know that at a critical point $Cp \in \partial\mathcal{CC}_i$, $\nabla d_i(Cp) \parallel \nabla h(Cp)$.

We assume that, without loss of generality, the robot starts the cycle algorithm at $S_i = \partial\mathcal{FCS}^j_{\lambda_i}$ that lies in the ceiling of the current cell and λ increases as the slice sweeps in the $\nabla h(x)$ direction. From this point the robot looks for critical points via the following phases.

1. ***Forward phase:*** First the robot follows a forward lapping path, $l_{forward}$, along $\mathcal{FCS}^j_{\lambda_i}$ from ceiling towards floor. Then it follows the wall along the path, $w_{forward}$, in the forward direction, *i.e.* $\langle \dot{w}_{forward}(t), \nabla h(w_{forward}(t)) \rangle > 0$.
   The robot terminates forward wall following if it laterally moves one lap width or encounters a critical point in the floor. Note that $w_{forward}(1) \in \mathcal{FCS}_{\lambda_{i+1}}$ and if the robot moves one lap width then $|\lambda_{i+1} - \lambda_i| = 2r$. In Figure 9, $l_{forward} \bullet w_{forward}$[2] is the dashed path between points S_i and 2 .
2. ***Reverse phase:*** The reverse phase is an interleaved sequence of reverse laps, $l_{reverse_r}$ (from floor towards ceiling), and reverse wall following paths, $w_{reverse_r}$ (initially starts in $-\nabla h(x)$ such that $\langle \dot{w}_{reverse_r}(0), \nabla h(w_{reverse_r}(0)) \rangle < 0$.
   Each reverse wall following operation terminates when the robot senses a critical point, Cp_k, for which $\nabla d_i(Cp_k) = -\nabla h(Cp_k)$, or when it returns back to the slice that contains the start point, *i.e.* $w_{reverse_m}(1) \in \mathcal{FCS}_{\lambda_i}$. In Figure 9, $l_{reverse_1} \bullet w_{reverse_1} \bullet \cdots \bullet l_{reverse_m} \bullet w_{reverse_m}$ is the solid path between points 2 and 3 for $m = 3$. Note that $m \geq 1$.
3. ***Closing phase:*** The robot may reach the start point at the end of the reverse phase, but if not, it executes the closing phase as follows. The robot follows one or more lapping paths, l_{close_r}, along the slice $\mathcal{CS}_{\lambda_i}$, possibly inter-mixed with wall following paths, w_{close_r} which initially starts in the forward direction, *i.e.*$\langle \dot{w}_{close_r}(0), \nabla h(w_{close_r}(0)) \rangle > 0$. and terminates when the robot returns to $\mathcal{FCS}_{\lambda_i}$. Note that if $w_{close_r}(0) \in \partial\mathcal{FCS}^{q+1}_{\lambda_i}$ then $w_{close_r}(1) \in \partial\mathcal{FCS}^{q}_{\lambda_i}$. In Figure 9, $l_{close_1} \bullet w_{close_1} \bullet \cdots \bullet l_{close_n} \bullet w_{close_n}$ is the dotted path between points 3 and S_i for $n = 2$. Note that $n \geq 0$.

[2] $f \bullet g$ is the concatenation of paths f and g [12]

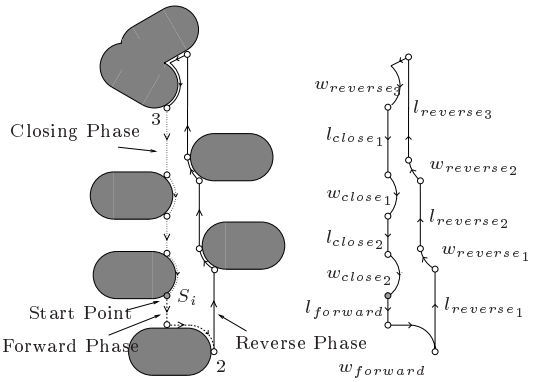

Fig. 9. The robot follows the dashed path between points S_i and 2 while it is executing the forward phase. While it is performing the reverse phase, it follows the solid path between points 2 and 3. During the closing phase, it follows the dotted path between points 3 and S_i. The resulting cycle path is the product of the consecutive wall following and lapping paths.

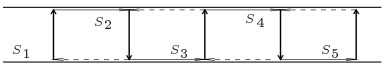

Fig. 10. S_i refers to start point of each cycle.

Simple Example: A simple example path generated by the algorithm is shown in Figure 10. The robot first executes the forward phase. It starts to lap at point S_1, encounters an object and performs wall following. Then it performs *reverse lapping* and at the end of *reverse wall following* returns back to S_1. The robot completes the *cycle*. Now, the robot must perform the next cycle. So first, it "undoes" the prior reverse wall following step. Let S_2 be the beginning of the next cycle, but we locate it where the robot first began lapping. The robot does not need to drive to S_2, because it has already traveled along previous lap. So it completes the next cycle starting with wall following, pretending as if it has started at S_2.

4.2 Completeness of the Cycle Algorithm

In this work, we consider the case where the slice is a straight line, but in actuality all results presented here are valid for any slice function with constant gradients. The maximum distance between two slices $\mathcal{CS}_{\lambda_i}$ and $\mathcal{CS}_{\lambda_{i+1}}$ is defined as

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

In this work we consider an inter-lap spacing of $2r$ where r is the radius of the robot. Then $LW(\mathcal{CS}_{\lambda_i}, \mathcal{CS}_{\lambda_{i+1}}) \leq 2r$.

The cycle algorithm makes the robot execute a sequence of wall following and lapping motions. The resulting cycle path is the product of the consecutive wall following and lapping paths. See Figure 9.

$$\begin{aligned}\Pi_{cycle} \quad = \quad & l_{forward} \bullet w_{forward} \bullet l_{reverse_1} \bullet w_{reverse_1} \\ & \bullet \cdots \bullet l_{reverse_m} \bullet w_{reverse_m} \bullet l_{close_1} \bullet w_{close_1} \\ & \bullet \cdots \bullet l_{close_n} \bullet w_{close_n}\end{aligned}$$

such that $l_{forward}(1) = w_{forward}(0)$, $w_{forward}(1) = l_{reverse_1}(0)$, *etc.*. Note that $m \geq 1$ and $n \geq 0$.

The Lemma 4.1 states that the cycle path generated by the cycle algorithm is a simple closed path. We assume that the free space is bounded and it is populated with finite number of obstacles.

LEMMA 4.1 *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* $\Pi_{cycle}(0) = \Pi_{cycle}(1)$.

See the proof in the appendix.

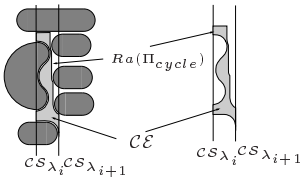

Fig. 11. The cycle path, Π_{cycle}, partitions $\mathcal{FCS}$ in two disjoint sets, the inside and the outside. $\mathcal{CE}$ denotes the inside, and it is bounded by the slices $\mathcal{CS}_{\lambda_i}$, $\mathcal{CS}_{\lambda_{i+1}}$ and the obstacles.

By the Jordan curve theorem [9], we know that the cycle path partitions $\mathcal{FCS}$ into two disjoint sets, the inside and outside. Let $\mathcal{CE}$ be the inside. See Figure 11.

LEMMA 4.2 *No obstacles can be fully contained in the interior of* $\mathcal{CE}$.

Proof: Consider the first two laps $l_{forward}$ and $l_{reverse_1}$ of the cycle path along the slices $\mathcal{CS}_{\lambda_i}$, $\mathcal{CS}_{\lambda_{i+1}}$. 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 $LW(\mathcal{CS}_{\lambda_i}, \mathcal{CS}_{\lambda_{i+1}}) = \lambda_{i+1} - \lambda_i \leq 2r$. See Figure 11. Hence a $2r$ diameter ball centered at x, $\beta_{2r}(x)$, can not be a subset of $\mathcal{CE}$. Since $\forall i, \exists\ x \in \mathcal{CC}_i$ such that $\beta_{2r}(x) \subset \mathcal{CC}_i$, $\mathcal{CC}_i$ can not be a subset of $\mathcal{CE}$. ■

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

COROLLARY 4.3 *No critical points exist in the interior of* $\mathcal{CE}$.

COROLLARY 4.4 *Critical points may lie in the* $\partial\mathcal{CE} = Ra(\Pi_{cycle})$.

Since critical points can only line in the boundary of $\mathcal{CE}$, if no critical points are sensed along the cycle path, then there does not exist any critical points inside of $\mathcal{CE}$.

We now introduce some terminology that will be useful to prove that the closing critical which may lie in the floor, interval or ceiling, is always going to encountered by the robot while it is executing the cycle algorithm. Let $\mathcal{FS}^J_\Lambda$ be an interval that contains a critical point

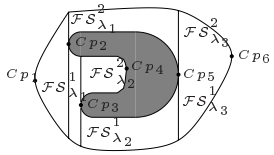

Fig. 12. Cp_k's denote critical points and $\mathcal{FS}^J_\Lambda$'s are the intervals that contain a critical point in its boundary. Cells are formed by taking away $\mathcal{FS}^J_\Lambda$'s from the free configuration space.

in its boundary, and $IC = \bigcup_{\Lambda,J} \mathcal{FS}^J_\Lambda$. Then a cell is a connected component of $\mathcal{FCS} \backslash IC$. See Figure 12.

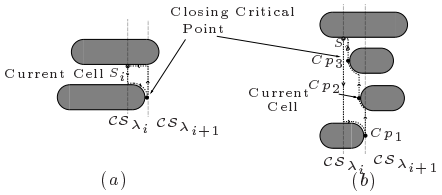

Fig. 13. Closing critical point is sensed during the forward phase in (a) and reverse phase in (b).

LEMMA 4.5 *If the robot senses a critical point* Cp_1 *only in the forward phase, then the critical point sensed is the closing critical point which is located in the floor (Fig. 13-*(a)*).*

Proof: Assume that the robot starts the cycle algorithm in $\mathcal{FCS}^j_{\lambda_i}$. By Corollary 4.3, $\forall\lambda,\ \lambda_i \leq \lambda < h(Cp_1)$, $\mathcal{FCS}^j_\lambda$ does not contain a critical point. Since Cp_1 is the only critical point along the cycle path, $\forall\lambda,\ \lambda_i \leq \lambda < h(Cp_1)$, $\partial\mathcal{FCS}^j_\lambda$ does not contain a critical point too. Cells are connected components of $\mathcal{FCS} \setminus IC$. Hence $\{\mathcal{FCS}^j_\lambda \in \mathcal{FCS} | \lambda_i \leq \lambda < h(Cp_1)\}$ is path connected to the current cell. Therefore Cp_1 is the closing critical point. ■

LEMMA 4.6 *If the robot senses zero or one critical point in the forward phase and one or more critical points during the reverse phase, but nothing else in the closing phase then the last critical point sensed during the reverse phase is the closing critical point (Fig. 13-*(b)*).*

Proof: Let $\{Cp_k\}$ be the set of critical points sensed during the forward and reverse phases, ordered in the order of appearance. By construction,

$$\lambda_{i+1} \quad = \quad h(w_{forward}(1)) > h(w_{reverse_1}(1)) >$$

$$h(w_{reverse_2}(1)) > \cdots >$$
$$h(w_{reverse_m}(1)) = \lambda_i$$

Therefore if $w_{forward}(1) = Cp_1$ then Cp_1 can not be the closing critical point.

Consider the intervals $\mathcal{FCS}^j_{h(Cp_k)}$ that contains a critical point Cp_k in its boundary. Since the cells are the connected components of $\mathcal{FCS}\backslash IC$, the subset of $\mathcal{CE}\backslash\bigcup\mathcal{FCS}^j_{h(Cp_k)}$ that forms a connected component with the current cell completes the current cell. This subset of $\mathcal{CE}$ has boundaries of $Ra(l_{forward})$, $\mathcal{FCS}^j_{h(Cp)}$, and subsets of $Ra(l_{forward})$ and $Ra(w_{reverse_m})$. Note that Cp is the closing critical point. By construction, along the path between the last critical point sensed in the reverse phase and S_i, there can not be any other critical points. Hence $\forall\lambda,\ \lambda_i \le \lambda < h(Cp)$, $\partial\mathcal{FCS}^j_\lambda$ does not contain a critical point. By Corollary 4.3, $\forall\lambda,\ \lambda_i \le \lambda < h(Cp)$, $\mathcal{FCS}^j_\lambda$ does not contain a critical point too. Hence $\{\mathcal{FCS}^j_\lambda \in \mathcal{FCS} | \lambda_i \le \lambda < h(Cp)\}$ is path connected to the current cell. Therefore Cp is the closing critical point. ■

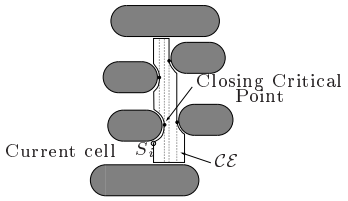

Fig. 14. The intervals that contain a critical point partitions $\mathcal{CE}$ into connected subsets. The closing critical point lies in the ceiling and the interval which contains it lies in the boundary of the current cell.

LEMMA 4.7 *If the robot senses one or more critical points during the closing phase and, does or does not sense other critical points during the reverse and the forward phases, then the last critical point along the cycle is the closing critical point located in the ceiling (Fig. 14).*

Proof: Let $\{Cp_k\}$ be the set of critical points along the cycle path, ordered in the order of appearance. Consider the intervals $\mathcal{FCS}^j_{h(Cp_k)}$ that contains a critical point Cp_k in its boundary. Since the cells are the connected components of $\mathcal{FCS}\backslash IC$, the subset of $\mathcal{CE}\backslash\bigcup\mathcal{FCS}^j_{h(Cp_k)}$ that forms a connected component with the current cell completes the current cell. This subset of $\mathcal{CE}$ has boundaries of $Ra(l_{forward})$, $\mathcal{FCS}^j_{h(Cp)}$,and subsets of $Ra(l_{forward})$ and $Ra(w_{close_n})$. Note that Cp is the closing critical point.

The cycle path is simple and closed, and its starting point is in the ceiling of the current cell. Hence the ceiling of the current cell is definitely followed by the robot while it is following the path w_{close_n}. By construction, along the path between the last critical point Cp, sensed along w_{close_n}, and S_i, there does not exist any other critical points . Therefore $\forall\lambda,\ \lambda_i \le \lambda < h(Cp)$, $\partial\mathcal{FCS}^j_\lambda$ does not contain a critical point. By Corollary 4.3, $\forall\lambda,\ \lambda_i \le \lambda < h(Cp)$, $\mathcal{FCS}^j_\lambda$ does not contain a critical point too. Hence $\{\mathcal{FCS}^j_\lambda \in \mathcal{FCS} | \lambda_i \le \lambda < h(Cp)\}$ is path connected to the current cell. Therefore Cp is the closing critical point. ■

Note that for the lemma above, if any one of the critical points sensed during the forward or the reverse phases was the closing critical point, then the closing phase would never be performed by the robot.

LEMMA 4.8 *The last critical point encountered along the cycle path is the closing critical point.*

Proof: The closing critical point can be sensed during the forward or reverse or closing phases. The robot performs all three phases, or none or one or two of them while executing the cycle algorithm. Lemmas 4.5, 4.6, 4.7 show that, whatever the case the last critical point sensed along the cycle is the closing critical point. ■

5 Planar Coverage with Finite Sensor Range

In some applications, the robot may have a detector, like a camera, with a non-zero range beyond the robot's perimeter. In this case the robot can perform coverage with a wider inter-lap spacing, while sensing the critical points remotely using the method described in Section 3.3. We are measuring the sensor range σ from the *perimeter* of the robot and the detector range, δ from the *center* of the robot. We updated the cycle algorithm by assuming $\sigma = \delta$ and the inter-lap spacing equal to 2δ. In this paper, we only outline the updated cycle algorithm. Our current work involves proving the completeness of this new algorithm. However the main steps of the updated cycle algorithm are the same as the zero range sensor version.

The addition to the forward phase of the algorithm is as follows. If the robot senses a critical point remotely while lapping, it only needs to perform forward wall following until it reaches the slice that contains the sensed critical point. See Figure 15. The extra step added to the reverse phase is as follows. If the robot senses a critical point in the reverse direction while it is lapping, then it terminates lapping and moves to the critical point. Then it performs reverse wall following at the bottom of the obstacle. See Figure 16.

1. *Forward phase:* The robot laps while looking for critical points in its sensor range until it encounters an obstacle. Then, it follows the wall until it (1) reaches the slice of the closest sensed critical point

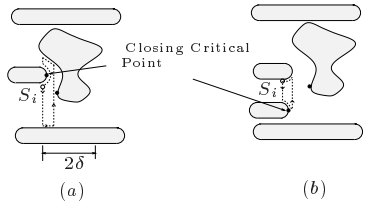

Fig. 15. (a) The robot senses a critical point remotely while during the foward phase and it performs forward wall following until it reaches the slice that contains the sensed critical point. (b) Even though the robot senses a critical point while forward lapping, it finds the closing critical point while following the boundary of the obstacle during the forward phase.

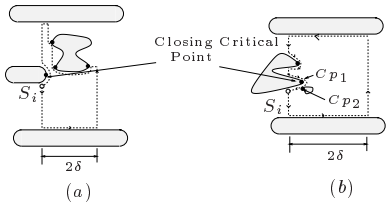

Fig. 16. (a) The robot senses a critical point during reverse lapping, moves to it and then completes the reverse and closing phases. (b) Even if the robot senses Cp_2 after sensing Cp_1 in the final phase, it senses the closing critical point during wall following .

in the forward direction (if any) or (2) encounters a critical point in the floor, if it exists, or (3) laterally moves one lap width. See Figure 15.

2. *Reverse phase:* The robot reverse laps until it (See Figure 16-(a))
   (a) encounters an obstacle. If the robot encounters an obstacle, it then reverse wall follows until
      i. it encounters a critical point and resumes reverse lapping or
      ii. laterally moves in the reverse direction by one lap width and terminates reverse phase
   (b) or a critical point in the reverse direction. The robot then travels directly to the critical point, follows the obstacle around its bottom until
      i. it encounters a critical point resumes reverse lapping or
      ii. latterly moves in the reverse direction by one lap width terminates reverse phase
3. *closing phase*, the robot executes one or more laps along the slice, possibly inter-mixed with wall following. Each wall following operation terminates when the robot encounters S_i or the slice in which S_i lies. Let Cp_1 be the last critical point encountered via wall following in the final phase. If, after sensing Cp_1, the robot remotely senses critical point Cp_2 in the forward direction that is closer to the start slice than Cp_1, this critical point is the closing critical point. See Figure 16-(b).

6 Experimental Results

We performed experiments with a Nomad 200 mobile robot equipped with 16 ultrasonic sensors. The result of an experiment performed in a 9×12 foot room with an inter-lap spacing equal to the robot's diameter is presented in Figure 17. We outlined the boundaries of the obstacles and cells, and numbered the critical points in the order of appearance for the sake of discussion. We placed arrows on the traced robot path. In the first experiment, the robot starts at critical point 1, laps for a short distance and then follows the wall for a robot width. The robot laps down and then returns to the starting point of this cycle, to ensure that no critical point was missed. In the next cycle, the robot finds critical point 2 and closes the cell. This procedure is repeated until the environment is covered.

The experiment in Figure 18 demonstrates the problem of dead-reckoning error while performing coverage in a larger environment. When the robot returns to critical point 3 we can observe the dead reckoning error because the ceiling of the cell between critical points 3 and 4 seems to have shifted upwards, when in actuality they have not moved. Also, from the robot's perspective, the environment has rotated, which from the free space's perspective, the robot has rotated its slice direction. Because of these changes, the robot cannot form an accurate decomposition of the space; in this experiment, the robots erroneously senses a seventh critical point in the upper cell between critical points 2 and 3. Note that we use the Bug2 [13] algorithm to return back to an already discovered critical point.

Solving the dead-reckoning problem is the topic of current research. We will exploit the topology of the dual adjacency graph to resolve this problem. The goal of this portion of work is to solve the simultaneous localization and coverage problem.

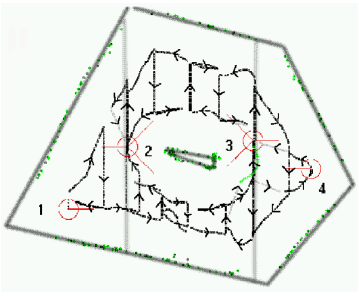

Fig. 17. The path followed by the robot is shown by dotted black lines. The gray dots represent the back track paths. The boundaries of the obstacles are outlined by straight lines.

7 Conclusion

In this paper, we presented methods to sense critical points and algorithms to encounter all of them. We ver-

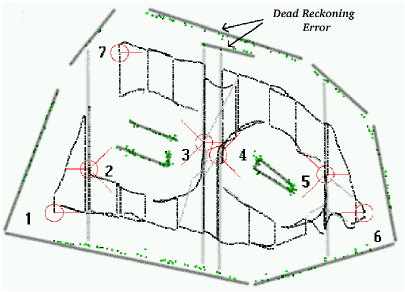

Fig. 18. In this experimental run effect of dead reckoning is easily observed.

ified our theoretical results by performing experiments. Our approach serves as a solution to the problem of sensor based coverage of unknown environments. Experimental results, however, showed the need for localization. Future work involves developing methods to solve the localization problem.

Appendix

Proof of Lemma 4.1.

Proof: Without loss of generality, we assume that the robot starts in $\partial\mathcal{FCS}^j_{\lambda_i}$ and begins the reverse phase in $\mathcal{FCS}_{\lambda_{i+1}}$, *i.e* $l_{forward}(0) \in \mathcal{FCS}^j_{\lambda_i}$ and $l_{reverse_1}(0) \in \mathcal{FCS}_{\lambda_{i+1}}$. By construction $l_{forward}(1) = w_{forward}(0)$ and $w_{forward}$ is a simple path such that $< \dot{w}_{forward}(t), \nabla h(w_{forward}(t)) >> 0$. Therefore $w_{forward}(1) \notin \mathcal{FCS}_{\lambda_i}$. The design of the algorithm guarantees that $w_{forward}(1) = l_{reverse_1}(0)$ and $l_{reverse_1}(1) = w_{reverse_1}(0)$, moreover at the end of each of the reverse wall following step, the robot gets closer to $\mathcal{FCS}_{\lambda_i}$[3]. Since there are finite number of obstacles, the robot eventually reaches $\mathcal{FCS}_{\lambda_i}$, *i.e.*

$$\begin{aligned}\lambda_{i+1} &= h(w_{forward}(1)) > h(w_{reverse_1}(1)) > \\ &\quad h(w_{reverse_2}(1)) > \cdots > h(w_{reverse_m}(1)) = \lambda_i\end{aligned}$$

Hence $w_{reverse_m}(1) \in \mathcal{FCS}_{\lambda_i}$. Now we need to show that the end point of the reverse phase can only be the start point or lie in the "upper" portion of the $\mathcal{FCS}_{\lambda_i}$ with respect to $l_{forward}(0)$, *i.e.* $w_{reverse_m}(1) \in \mathcal{FCS}^q_{\lambda_i}$ where $q \geq j$. We know that $w_{forward}$ is a simple path with $w_{forward}(1) \in \mathcal{FCS}_{\lambda_{i+1}}$ and thus the start point of the reverse phase, $l_{reverse_1}(0) \in \mathcal{FCS}_{\lambda_{i+1}}$. In the reverse phase, the robot always laps in the reverse direction of forward lapping. Therefore $w_{reverse_m}(1)$ can only lie in $\mathcal{FCS}^q_{\lambda_i}$ where $q \geq j$. Since wall following takes place in the boundary, $w_{reverse_m}(1)$ should lie in the boundary of $\mathcal{FCS}^q_{\lambda_i}$, $q \geq j$.

Since the end point of the reverse phase lies in $\partial\mathcal{FCS}^q_{\lambda_i}$, $q \geq j$, the robot should move away (if $w_{reverse_m}(1) \neq l_{forward}(0)$) from the ceiling of the cell towards floor along $\mathcal{FCS}_{\lambda_i}$ to reach $l_{forward}(0)$. However lapping along the slice towards the start point is, by itself, not enough. When the robot encounters an obstacle, by construction, it initially starts to move in $\nabla h(x)$ direction along the path, w_{close_r}. Note that the range of any w_{close_r} can never intersect with the range of any $w_{reverse_r}$ or $l_{reverse_r}$, otherwise the robot would never perform the closing lapping and wall following motions that it has performed up to the intersection point. We know that if $w_{close}(0) \in \partial\mathcal{FCS}^{q+1}_{\lambda_i}$ then $w_{close}(1) \in \partial\mathcal{FCS}^q_{\lambda_i}$. Therefore at the end of the each step of the closing phase robot gets closer to $l_{forward}(0)$ along $\mathcal{FCS}_{\lambda_i}$[4]. Since there are finite number of obstacles in the environment, robot eventually reaches the start point of the cycle algorithm. ■

[3] We omit the proof of this statement due to space restrictions.

[4] We omit the proof of this statement due to space restrictions

References

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

[2] J.F. Canny and M. Lin. An opportunistic global path planner. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1554–1561, 1990.

[3] J.F. Canny and M. Lin. An opportunistic global path planner. *Algorithmica*, 10:102–120, 1993.

[4] Z. L. Cao, Y. Huang, and E. Hall. Region filling operations with random obstacle avoidance for mobile robots. *Journal of Robotic systems*, pages 87–102, February 1988.

[5] H. Choset, E. Acar, A. Rizzi, and J. Luntz. Exact cellular decompositions in terms of critical points of morse functions. In *Submitted to IEEE ICRA'00*, San Francisco, CA, 1999.

[6] H. Choset, K. Nagatani, and N. Lazar. The Arc-Transversal Median Algorithm: an Approach to Increasing Ultrasonic Sensor Accuracy. In *Proc. IEEE Int. Conf. on Robotics and Automation*, Detroit, 1999.

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

[8] F. H. Clarke. *Optimization and Nonsmooth Analysis*. Society of Industrial and Applied Mathematics, Philadelphia, PA, 1990.

[9] R. Courant and H. Robbins. *What is Mathematics?* Oxford University Press,Inc., New York, NY, 1996.

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

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

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

[13] V. Lumelsky and A. Stepanov. Path Planning Strategies for Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape. *Algorithmica*, 2:403–430, 1987.

[14] E. Rimon and J.F. Canny. Construction of C-space Roadmaps Using Local Sensory Data — What Should the Sensors Look For? In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 117–124, San Diego, CA, 1994.

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