Proceedings of the 2002 IEEE
International Conference on Robotics & Automation
Washington, DC • May 2002

Exploiting Critical Points to Reduce Positioning Error for Sensor-based Navigation

Ercan U. Acar Howie Choset
Carnegie Mellon University
Pittsburgh, PA15213
eua,choset+@andrew.cmu.edu

Abstract— **This paper presents a planner that determines a path such that the robot does not have to heavily rely on odometry to reach its goal. The planner determines a sequence of obstacle boundaries that the robot must follow to reach the goal. Since this planner is used in the context of a coverage algorithm already presented by the authors, we assume that the free space is already, completely or partially, represented by a cellular decomposition whose cell boundaries are defined by critical points of Morse functions (isolated points at obstacle boundaries). The topological relationship among the cells is represented by a graph where nodes are the critical points and edges connect the nodes that define a common cell (i.e., the edges correspond to the cells themselves). A search of this graph yields a sequence of cells that directs the robot from a start to a goal. Once a sequence of cells and critical points are determined, a robot traverses each cell by mainly following the boundary of the cell along the obstacle boundaries and minimizes the accumulated dead-reckoning error at the intermediate critical points. This allows the robot to reach the goal robustly even in the presence of dead-reckoning error.**

Keywords— **Navigation, topological localization, critical points.**

1 Introduction

Classical motion planning techniques [10] have successfully supplied algorithms that determine a path between start and goal configurations assuming perfect position information is avaliable. Three methods have dominated the motion planning field: start-goal planners [11], [15], roadmaps [3], and exact or approximate cellular decompositions [4], [17]. This work exploits the structure of an *exact cellular decomposition* to plan a path between start and goal configurations that is less "sensitive" to dead-reckoning error. Exact cellular decompositions represent the free space as the union of non-overlapping regions called cells such that adjacent cells share a common boundary. Our work is mainly motivated by earlier work in covering unknown spaces using cellular decompositions [1]. The details of how to use a cellular decomposition to perform coverage is not pertinent to this paper. However, when incrementally constructing a cellular decomposition, the planner must use a start-goal planner to direct the robot from one cell to another. In this paper, we address the start-to-goal path planning problem from a localization perspective without heavily relying absolute position of the mobile robot determined using odometry which is prone to accrue error. We assume that the robot navigates through a free space that has been partially mapped or mapped via a cellular decomposition. Using this cellular decomposition, we develop a navigation algorithm as a first step towards automatically defining topologically meaningful natural landmarks and connections between them which the robot can use to robustly navigate from one point to another.

We used critical points (points on obstacle boundaries) where a topological change in the space occurs to define cells that have "simple" structure of non-intersecting "upper"and "lower" boundaries without critical points on them, and side boundaries that contain critical points (Fig. 1). When the planner determines a path, i.e., a sequence of cells, it has determined a topological path, not an exact one. The planner then determines the exact path by following the upper or lower boundary of each cell, possibly switching from an upper to a lower (or visa versa) at a critical point. Most of this path is robust to odometry error because the robot servos off of the boundary most of the time, only departing this "safe" area at some of the critical points. This method strikes resonance with probabilistic method of Roy et al. [16] which they term a coastal navigation approach. This approach directs the robot to remain close to obstacle boundaries which increases the chances of gathering information for localization.

The philosophy of utilizing topological information to position a robot is based on our previous work in simultaneous mapping and localization algorithms without explicit localization using the topology of generalized Voronoi diagrams (set of points equidistant to two obstacles) [6]. We used the nodes of the diagram as the topological features of the free space to localize. This method was based on Kuipers [9] that utilizes distinctive places for mobile robot navigation. It is also worth pointing out the work of Dudeck et al. [7] who also uses a topological method for localization.

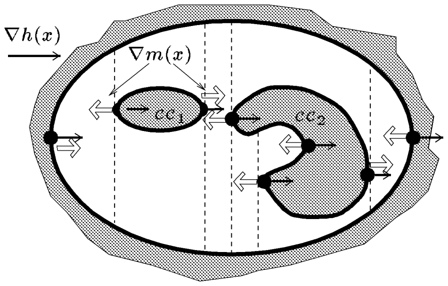


Fig. 1. The cellular decomposition using the critical points at which sweep direction $\nabla h(x)$ and surface normals $\nabla m(x)$ are parallel.

2 Exact Cellular Decomposition in Terms of Critical Points

The exact cellular decomposition is built upon "a slicing method" [3] that sweeps a slice $\mathcal{CS}_\lambda$ (a line segment) throughout the configuration space $\mathcal{CS}$ of a circular robot. The slice is the pre-image of a real-valued function, typically a *Morse function* [12], such as $h(x) = x_1$, i.e. $\mathcal{CS}_\lambda = \{x \in \mathcal{CS} | h(x) = \lambda, \lambda \in \Re\}$. Varying λ has the effect of sweeping the slice through the configuration space. While sweeping the slice[1], its connectivity in the free space changes when the slice encounters obstacles. Connectivity changes occur at points called *critical points* [12]. Just as Canny used these critical points to connect disconnected roadmap fragments [3], we defined the cells in our cellular decomposition with them [5]. The boundaries of the cells are the slices that contain critical points and the boundaries of obstacles between such slices.

We use the Reeb graph [8], [14] to represent the topology of the cellular decomposition. The Reeb graph's nodes correspond to critical points. Two nodes share an edge if their corresponding critical points (generically) define a cell. In other words, we assume that "left" or "right" most cell boundaries are each defined by a slice that has only one critical point each.

In our previous work [1], we presented an algorithm that incrementally constructs this cellular decomposition in unknown spaces. We know that at a critical point, the gradient of $h(x)$, $\nabla h(x)$, and the surface normals, $\nabla m(x)$, of the obstacles $\mathcal{CC}_i$ are parallel to each other [5] (Fig. 1). We use range sensors to determine the surface normals and thus to sense critical points with range sensors (Fig. 2).

Fig. 3 shows an experimental result of the decomposition of a $2.5 \times 3.1[m]$ room with a stool in the middle. The dotted black lines represent the path traced by the center point of the robot. The vertical lines are the lapping portions of the path and the jagged-curved lines

[1]Since there is no a priori information about the space, the sweep direction is chosen arbitrarily.

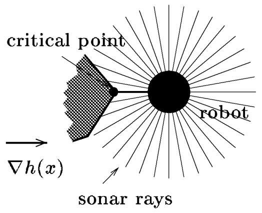


Fig. 2. The robot uses its range sensor to sense the critical point. When the line segment that is joining the center of the robot with the closest point to the robot on the obstacle is parallel to $\nabla h(x)$, the robot locates the critical point.

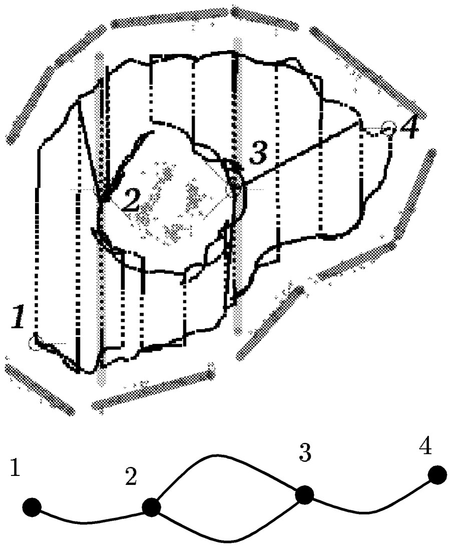


Fig. 3. The coverage path followed by the robot while it is incrementally constructing the graph representation of an unknown space by sensing critical points $1, 2, 3, 4, 2$ (in the order of appearance). For the sake of discussion, we outlined the boundaries of the obstacles and cells.

represent boundary following. Note how the boundary following path resembles the configuration space obstacle for the mobile robot. This makes sense because we are taking the center point of the circular robot as a reference point and we are finding the critical points in the configuration space using work space distance measurements. The robot senses the critical points $1, 2, 3, 4$ as it maps the space. When the robot senses the critical point 4, it travels back to the critical point 3 using the Bug2 algorithm.

In this experiment, dead-reckoning error did not cause any problems because of the small size of the space. However, even in a slightly larger environment or one with a carpeted floor, dead-reckoning error is going to cause the Bug2 algorithm to fail, thereby not directing the robot to the goal. In this paper, we propose an alternative approach to Bug2 that is less sensitive to dead-reckoning error. This could be used as a mobile robot navigation procedure, but only if a cellular decomposition, such as the one described above, is

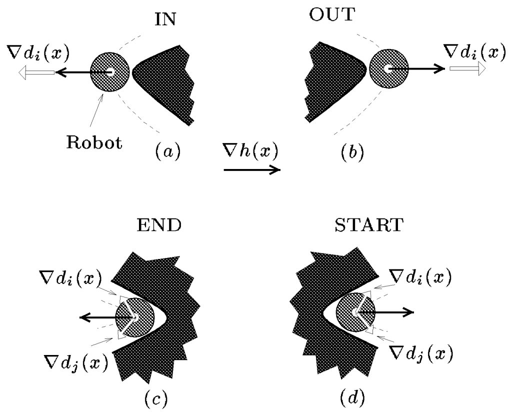


Fig. 4. The robot can use the direction of $\nabla d_i(x)$ with respect to $\nabla h(x)$ to identify four different types of critical points. (a) $-\nabla h(x) = \nabla d_i(x)$, IN (local minima) (b) $\nabla h(x) = \nabla d_i(x)$, OUT (local maxima) (c) $-\nabla h(x) \in CO\{\nabla d_i(x), \nabla d_j(x)\}$, END (local maxima) (d) $\nabla h(x) \in CO\{\nabla d_i(x), \nabla d_j(x)\}$, START (local minima).

available to the planner.

3 Characterization of Critical Points

In previous work, we characterized the critical points in terms of a distance function to reject bad sensor readings [2]. We describe the same characterization here to formulate our navigation approach. When we use our range sensor as a ring, we are really modeling the distance function $d_i(x)$ and its gradient $\nabla d_i(x)$. Since the surface normals are parallel to the distance gradient, critical points occur when $\nabla d_i(x)$ is parallel to $\nabla h(x)$.

Using the relative direction of $\nabla h(x)$ and $\nabla d_i(x)$ we classify the critical points as IN (I), OUT (O), START (S), END (E). When the obstacle is locally convex near the critical point, we have an IN or OUT critical point where $\nabla d_i(x) = -\nabla h(x)$ at IN critical points and $\nabla d_i(x) = \nabla h(x)$ at OUT critical points (Fig. 4(a,b)). When the obstacle is locally concave, we have a START or an END critical point. Note that at such critical points, the gradient of $d_i(x)$ is non-smooth. Then, at an END critical point $-\nabla h(x) \in CO\{\nabla d_i(x), \nabla d_j(x)\}$ where $CO\{\nabla d_i(x), \nabla d_j(x)\}$ is the convex hull of $\nabla d_i(x)$ and $\nabla d_j(x)$ (Fig. 4(c)). At a START critical point $\nabla h(x) \in CO\{\nabla d_i(x), \nabla d_j(x)\}$ (Fig. 4(d)). When the boundary's curvature is smaller than the robot's periphery, i.e., the gradient of $d_i(x)$ is smooth, we cannot distinguish the START and END critical points with IN and OUT critical points using range data (Fig. 5). However, by observing the relative history of the dead-reckoning data (passing a local maximum or minimum), we can make this judgment.

While the robot is constructing the cellular decomposition shown in Fig. 3, it encounters all possible types of critical points. The robot starts to cover the space at a START critical point and encounters IN and OUT,

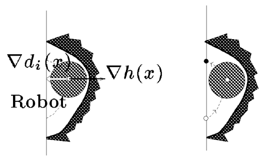


Fig. 5. The curvature of the boundary is smaller than $1/r$ where r is the radius of the robot. In this case the robot uses relative position data to sense and determine the type of the critical point.

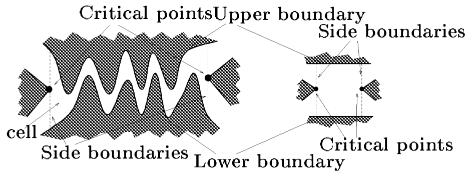


Fig. 6. A cell (complex or not) has upper and lower boundaries that do not intersect, and side boundaries that contain critical points. Lower and upper boundaries never contain any critical points. This simple structure of the cell simplifies the navigation between the critical points.

and finally END critical points. In this paper, our focus is not in this "mapping" stage but rather planning a path using this map representation that is less sensitive to dead-reckoning error.

4 Navigation Algorithm to Traverse a Cell

We assume that the robot starts at a critical point and requires a path to a goal critical point. Note that this path may pass through multiple cells as directed by the Reeb graph. Our approach for determining this path is to define a sub-path within each cell and then concatenate each of the sub-paths. Whereas with conventional Bug algorithms, dead-reckoning error accumulates through out the entire navigation mission, with our approach dead-reckoning error is minimized at the end of each cell, i.e., at intermediate critical points along the path, the robot minimizes the error. First we describe how to traverse just one cell and then how to concatenate the paths.

Each cell has one ceiling, one floor[2] and two critical points that lie on slices that form side boundaries[3]. We know that ceiling and floor boundaries never intersect each other and Morse Theory [12] assures us that there cannot be any critical points in their interiors (Fig. 6). Therefore, to traverse any cell, the robot can either follow the ceiling or the floor of the cell between the slices that form the side boundaries. Since the robot travels along the boundary (at a safety distance), this

[2]We borrowed the terms ceiling and floor from computational geometry literature [13]. Ceiling refers to the upper boundary of a cell and floor refers to the lower boundary.

[3]Note that when we have a START or an END critical point, corresponding side boundary length is zero.

approach has the main advantage of not heavily relying on odometry. In other words, the robot's path is determined by a control law that servos off of the obstacle boundaries, until it passes by a slice that bounds the cell. Note that the robot cannot exactly terminate its motion on the slice because of positioning error.

The algorithm that we propose has three phases:

- **Boundary-following (BF) phase:** The robot follows the ceiling or floor of the cell being traversed according to a slice sweeping direction directed by the Reeb graph such that inner product of the velocity vector $V(x)$ and $\nabla h(x)$, $\langle V(x), \nabla h(x)\rangle$ has the same sign for all points x on the path. This motion should terminate, when the robot arrives at the slice that contains the cell's goal critical point. If the goal critical point is already on the boundary being followed, the robot finds the slice by sensing the goal critical point using its range sensors. However, if the goal critical point is not on the followed boundary, then the robot must rely on odometry to determine that it has reached the slice that contains the goal critical point. To accommodate the dead-reckoning error, the robot overshoots this slice by a pre-determined constant amount l that is less than the robot's diameter D_{robot}. Note that if the robot senses another critical point before reaching the "overshoot" slice, it also terminates boundary-following phase.
- **Lapping (LP) phase:** When neither the starting nor goal critical points are located in the ceiling or floor, the robot moves along a slice to alternate the boundary that is to be followed. This motion terminates when the robot encounters a boundary.
- **Reverse boundary-following (RBF) phase:** When, as a result of the boundary-following phase, the goal critical point is not found, the robot performs the lapping phase to alternate the boundary. Now, the robot follows the new boundary in the reverse direction of the previous boundary following event. In other words, $\langle V(x), \nabla h(x)\rangle$ has the same sign which is opposite of the boundary-following phase. This motion ends when the robot senses the goal critical point.

Here, we assume the relative size of each cell compared to the size of the entire space is small. When the robot needs to depart from the ceiling or floor, it only needs to use one coordinate (x) but not both (x, y). Essentially, with this approach we reduce the dimension of the problem for localization by one. Moreover when the robot needs to look for a critical point, it only requires to know the type of the critical point because it can use its range sensors to locate the critical point.

The execution order of the phases depends on the types of starting and goal critical points that defines the cell being traversed and the cell's position with respect to the critical point, UP (U) or DOWN (D) or MIDDLE (M). Since each cell has two critical points, cell position has two arguments, each with respect to the corresponding critical point. If the corresponding critical point is

- in the cell's floor, cell position is UP,
- in the cell's ceiling, cell position is DOWN,
- in the interior of a slice that forms a side boundary, cell position is MIDDLE[4].

Note that all this information is encoded in the Reeb graph. For illustration, in Fig. 7, each cell position with respect to the corresponding critical points are shown. For example the cell wedged in between obstacles c1 and c2 has IN and OUT critical points. The cell's position with respect to IN critical point is DOWN and with respect to OUT critical point is UP.

In the following, we explain some cases with the execution order of the phases. The other possible combinations can be obtained by taking mirror images of these.

- Starting critical point is IN, goal critical point is IN.
  - Cell position is (U,M). The robot performs boundary-following, lapping and reverse boundary-following phases (Fig. 8(a)).
  - Cell position is (U,M). The robot performs lapping and boundary-following phases (Fig. 8(b)).

  Note that in these cases, the cell position is the same. However, we still can determine the correct order of the phases because the cell position with respect to the starting critical point is different.

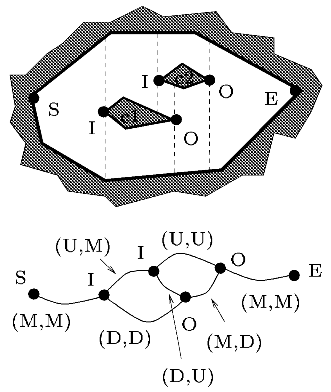
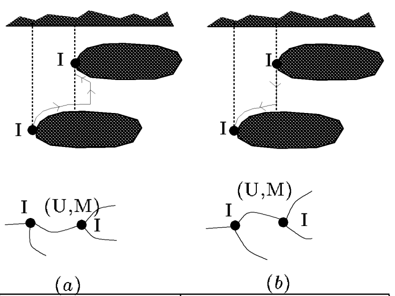


Fig. 7. All information needed to traverse a cell is encoded in the Reeb graph. Each cell (edge) has two critical points and its relative location with respect to corresponding critical point is known. For example, the right most cell has OUT and END critical points. The cell's position with respect to OUT and END critical points is (M,M).

[4]With respect to START and END critical points, cell position is always MIDDLE.



Start/Goal	I/I	
Cell Position	(U,M)	(M,U)
Phase Order	BF LP RBF	LP BF

Fig. 8. The robot starts at an IN critical point and reaches an IN critical point. Even though the information encoded in the edge is the same for both cases, the robot executes the correct phases because cell's position with respect to the starting critical point is different.

- Starting critical point is IN, goal critical point is OUT.
  - Cell position is (U,D). The robot performs lapping and boundary-following phases (Fig. 9(a)).
  - Cell position is (D,D). The robot only performs boundary-following phase (Fig. 9(b)).
  - Cell position is (M,M). The robot performs lapping, boundary-following, lapping and reverse boundary-following phases (Fig. 9(c)).
- Starting critical point is END,
  - Goal critical point is OUT (Fig. 10 (a)). Cell position is (M,M). The robot first performs boundary-following, then lapping and finally reverse boundary-following phases.
  - Goal critical point is IN (Fig. 10 (b)). Cell position is (D,M). The robot only performs boundary-following phase.

5 Complexity of the algorithm

We formulate the complexity of traversing a single cell in terms of the total length of the perimeter of the obstacles P_{cell} within a cell, diameter D_{space} of a disk that completely encloses the entire space and the diameter of the robot, D_{robot}. To calculate the total path length traveled, we first analyze the boundary-following phase. In the worst case, the boundary-following path length within the cell being traversed cannot be greater than P_{cell}. However, during boundary-following phase, the robot also follows the boundary of an obstacle outside of the cell. Since the amount of distance traveled laterally after reaching the slice that contains the goal critical point cannot be greater than the robot's diameter D_{robot}, this boundary following path is bounded above by $D_{space} + D_{robot}$. Therefore total path length traveled during boundary-following phase cannot be greater than $P_{cell} + D_{space} + D_{robot}$. During lapping phase the robot moves along a straight line. Since the space is bounded by the D_{space} diameter disk, total lapping path is bounded by $2D_{space}$. Finally in the searching phase, the robot follows the boundary of an obstacle outside of the cell. The path length in this phase is also bounded by $D_{robot} + D_{space}$. Therefore in the worst case total path length traveled is

$$4D_{space} + 2D_{robot} + P_{cell}.$$

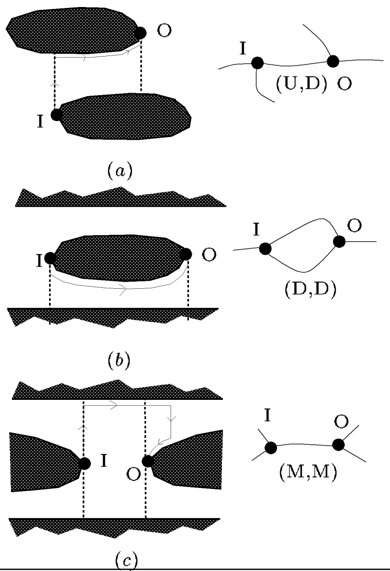
For an average case where the cell being traversed and the neighboring cells are small in size relative to the D_{space} diameter disk, complexity becomes

$$2D_{space} + 2D_{robot} + \frac{P_{cell}}{2}.$$

Note that Bug2 algorithm may guide the robot along the perimeter of the entire space to traverse a cell that has critical points that are not in line of sight of each other (Fig. 11).

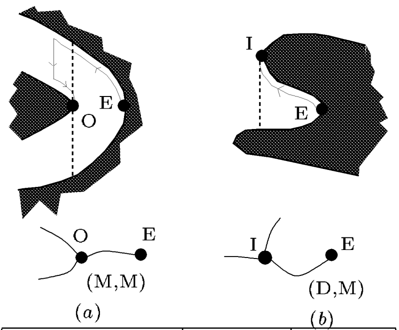
6 Navigation between Cells without a Common Boundary

When the starting and goal critical points do not belong to the same cell, we use the Reeb graph to determine a sequence of cells to traverse. The robot then uses the algorithm presented in the previous sec-



Start/Goal	I/O		
Cell Position	(U,D)	(D,D)	(M,M)
Phase Order	LP BF	BF	LP BF LP RBF

Fig. 9. The robot starts at an IN critical point and reaches an OUT critical point.



Start/Goal	E/O	E/I
Cell Position	(M,M)	(D,M)
Phase Order	BF LP RBF	BF

Fig. 10. The robot starts at an END critical point and reaches an IN or OUT critical point.

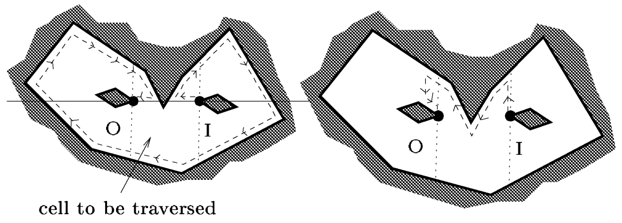


Fig. 11. The path followed by the robot to traverse a cell between IN and OUT critical points using Bug2 and our algorithm in a worst case scenario.

tion to traverse each one. Fig. 12 depicts an example. The robot starts from right most END critical point and follows the ceiling of the cell CE_1 between critical points 6 and 5. Then the robot traverses the cell CE_2 between critical points 4 and 5 by performing lapping, boundary-following, lapping and reverse boundary-following phases. From critical point 4, the robot only performs boundary-following phase to reach critical point 3. Finally the robot arrives at the goal critical point by executing lapping, boundary-following, lapping and reverse boundary-following phases.

The total path length between starting and goal critical points that belong to different cells is bounded above by

$$N(2D_{space} + 2D_{robot} + \frac{P_{cell}}{2}).$$

where N is the number of cells traversed.

7 Experiment

We demonstrate our navigation algorithm using Nomad Scout2 mobile robot that has 16 sonar sensors and wheel encoders for dead-reckoning while it is covering an unknown space. The robot operates in a $7 \times 3[m^2]$ room with one object which has carpeted and vinyl tiled floor. Note that the carpeted floor introduces more slippage than vinyl floor. Fig. 13 shows a sketch of the

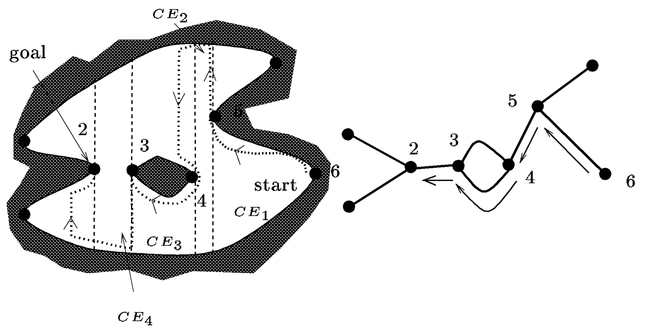


Fig. 12. The robot traverses the cells between the critical points 6 − 5, 5 − 4, 4 − 3, 3 − 2 to reach the goal critical point 2. Even though the robot could have just followed the lower boundary of environment, visiting intermediate critical points helps to reduce the dead-reckoning error.

environment and the result of the experimental run. When the robot is done covering the cell between the OUT and END critical points using back and forth motions[5], it needs to travel back to the OUT critical point so that it can start to cover a new cell. The robot first executes the boundary-following phase until it reaches the slice that is laterally half of the robot's diameter away from the slice that contains the OUT critical point. Then the robot performs the lapping phase until it reaches the boundary of the stool. Finally, the robot executes the reverse boundary-following phase until it senses the OUT critical point. In this experiment, the observable dead-reckoning is 25 cm. Note that if we were using Bug2, the robot will be never able to reach the OUT critical point because it would have tried to reach (x, y) coordinate of the critical point. On the other hand with our algorithm the robot uses its sonar sensors to locate the critical point.

8 Conclusions

In this paper, we presented an algorithm to travel between two points that does not heavily depend on dead-reckoning. We used an exact cellular decomposition to simplify the problem that has cells with simple structure. The structure of cells allowed us to guide the robot mainly along the boundaries of the obstacles. Since our formulation of the cellular decomposition was parameterized using only one variable, we reduced the dimension of the problem for localization by one. Therefore, whenever the robot needed to depart from the boundary, it only used one coordinate, but not two. Moreover when the robot was searching a goal position which is a critical point, it only used its sonar sensors. We exploited the structure of a graph representation of the decomposition to determine the execution order of our algorithm. We successfully veri-

[5]For the sake of clarity, we do not show the path data related to the exploration phase.

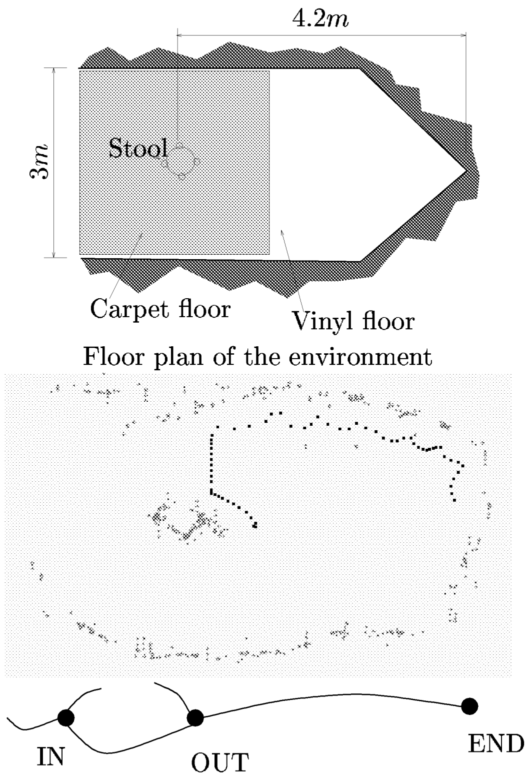


Fig. 13. The robot needs to travel from END critical point (rightmost corner of the room) to OUT critical point (on the boundary of the stool) so that it can start to cover a new cell. Our navigation algorithm first guides the robot along the ceiling of the cell and then along a slice. Finally the robot searches the OUT critical point by reverse boundary-following. Even though there is observable dead reckoning error (the edge of the stool is shifted) the robot finds the OUT critical point successfully using its sonar sensors.

fied our approach by performing experiments in generic environments. In the future, we are planning to extend our approach to non-generic spaces where critical points are not isolated points.

It is still possible that adequate dead-reckoning error can accumulate to fail the robot. However if the size of each cell is small, then relative dead-reckoning error is not drastic. It is also possible that the orientation of the robot can drastically change as the robot travels for long distances. To deal with this situation we are planning to use a cheap compass that is not necessarily very accurate but accurate enough up to 10 degrees.

Finally, we would like to point out that goal positions located at the boundary of obstacles are harder to reach even without accumulated dead-reckoning error. This is mainly because of the noisy sensor measurements. In our particular case, distance measurements that the robot makes using ultrasonic sensors are noisy. Therefore because of these noisy measurements, the boundary of the obstacles are not clean one dimensional sets; they are fuzzy regions with some "thickness". Since this thickness depends on the reflectivity of the obstacles and the sensor accuracy, it is not always possible to set a fixed uncertainty bound.

References

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

[2] E.U. Acar and H. Choset. Robust sensor-based coverage of unstructured environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, Hawai, 2001.

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

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

[5] H. Choset, E. Acar, A. Rizzi, and J. Luntz. Exact cellular decompositions in terms of critical points of Morse functions. In *Proc. of IEEE ICRA'00, Int. Conf. on Robotics and Automation*, pages 2270–2277, San Francisco, CA, 2000.

[6] H. Choset and K. Nagatani. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17:125 –137, April 2001.

[7] G. Dudeck, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation*, 7:859–865, Dec. 1991.

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

[9] B. Kuipers and Y.T. Byan. A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations. *Journal of Robotics and Autonomous Systems*, 8:47–63, 1991.

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

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

[12] J. Milnor. *Morse Theory.* Princeton University Press, Princeton, New Jersey, 1963.

[13] J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, 1998.

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

[15] E. Rimon and D. E. Koditschek. Exact Robot Navigation Using Artificial Potential Functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, October 1992.

[16] N. Roy, W. Burgard, D. Fox, and S. Thrun. Coastal navigation - mobile robot navigation with uncertainty in dynamic environments. In *IEEE Int'l. Conf. on Robotics and Automation*, pages 35 –40, Detroit,MI, May 1999.

[17] J.T. Schwartz and M. Sharir. On the "Piano Movers" Problem II: General Techniques for Computing Topological Properties of Real Algebraic Manifolds. *Advances in Applied Mathematics*, 4(1):298–351, 1983.