Mobile Robot Navigation with Complete Coverage of Unstructured Environments

E. Garcia * P. Gonzalez de Santos

Automatic Control Department, Industrial Automation Institute (CSIC), 28500 Arganda del Rey, Madrid, Spain

Abstract

There are some mobile-robot applications that require the complete coverage of an unstructured environment. Examples are humanitarian de-mining and floor-cleaning tasks. A complete-coverage algorithm is then used, a path-planning technique that allows the robot to pass over all points in the environment, avoiding unknown obstacles. Different coverage algorithms exist, but they fail working in unstructured environments. This paper details a complete-coverage algorithm for unstructured environments based on sensor information. Simulation results using a mobile robot validate the proposed approach.

Key words: Mobile-robot navigation, path planning, complete coverage, unstructured environments, humanitarian de-mining

1 Introduction

Path-planning algorithms are well understood for a variety of exploratory applications. Artificial-intelligence methods like learning algorithms are widely used to solve the problem of robot navigation in unstructured environments. However there are some types of mobile-robot applications that need a different path-planning technique. Scanning applications, like landmine detection, cleaning tasks and terrain-map generation, require not findind the shortest path to a point in the environment but scanning over all points in the environment and avoiding obstacles of unknown location. This is known as the complete-coverage problem in unstructured environments. A variety of coverage algorithms exist [1–5], but the most powerful ones are those that rely

* Corresponding author.
Email address: egarcia@iai.csic.es (E. Garcia).

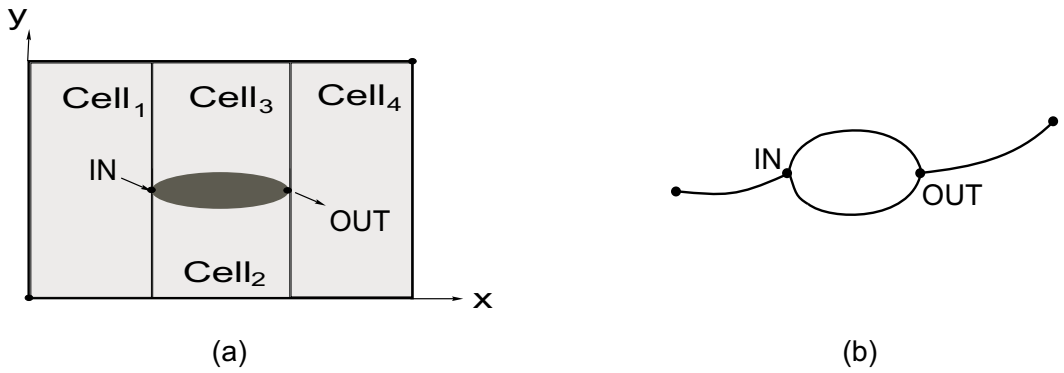

Fig. 1. (a) Exact cellular decomposition of a rectangular area with obstacles. (b) Adjacency graph.

on finding the critical points of a function to guarantee the completeness of the method [4–6]. The complete-coverage algorithm herein proposed is of this type and extends Choset's algorithm to achieve better performance in unstructured environments. Choset first developed his coverage algorithm for known spaces [7] and later adapted it to cover unknown ones [5,6], but the resulting algorithm fails in some cases to detect critical points.

Previous coverage algorithms that are based on finding critical points use critical points to achieve an *exact cellular decomposition* of the environment. This decomposition divides the environment into a finite number of regions called *cells*, which are free of obstacles. Figure 1(a) shows an example, the cellular decomposition of a rectangular area with an obstacle. A cell can then be swept by a simple zigzagging pattern of motion in which the robot moves back and forth along successive grid lines that sweep across the entire field from left to right. Two different types of critical points can be found: An IN critical point closes an existing cell and opens two new ones; an OUT critical point closes two existing cells and opens one new cell (see Figure 1(a)). Then, the complete coverage of the field is guaranteed by means of an adjacency graph, which assures that every cell is visited. In the adjacency graph each cell is modeled as an edge connecting two nodes, which represent the critical points. Therefore, an IN critical point is represented as a node with two diverging edges, and an OUT critical point is represented by a node where two edges converge (see Figure 1(b)). Although this type of coverage algorithm has been proved to be more efficient than previous approaches, two main problems arise when attempting to use this coverage algorithm in unstructured environments. The first one is critical-point detection. When the location and shape of the obstacles are unknown, the robot has to include a critical-point searching method in its zigzagging pattern of motion, and the success of the cellular decomposition relies on the efficiency of this method. The second problem that arises in unstructured environments is the on-line generation of the adjacency graph. Two consecutively detected IN and OUT critical points do not necessarily belong to the boundary of the same obstacle, depending on the robot's mo-

tion. Therefore, completion of the on-line adjacency graph is no trivial matter and requires an IN/OUT matching algorithm. These two problems arise when applying coverage algorithms from the literature, and this paper proposes an enhancement of Choset's coverage method to solve them.

The outline of the paper is as follows: first Section 2 describes the robot and the environment. Then, Section 3 briefly reviews the complete coverage problem in unstructured environments and shows some deficiencies of previous approaches. Section 4 explains the improvement of the proposed method in finding critical points and also shows how the method guarantees complete coverage by means of an extension of the adjacency graph. Finally, Section 5 shows some simulation results of the proposed approach and conducts a performance analysis. Lastly, Section 6 presents some conclusions.

2 Robot and environment description

2.1 The robot

Unlike in most previous work, here the robot is not considered a point but a finite area. It can be a rectangular area or a circular area of width/diameter D_{rob} . This implies that the opening size between two obstacles it can pass through is lower-bounded by this amount. The robot has knowledge of its exact location at all times (drawn from a Global Positioning System, an Inertial Navigation System, etc.), and it is equipped with a sensor capable of detecting obstacle boundaries (range sensors, bumpers, etc.). The sensor is assumed to detect obstacles within an area of radius $R \geq D_{rob}/2$ around the robot's body. Finally, the robot is also equipped with a scanning sensor for carrying out its main task (mine detector, cleaning scrub, etc.). A navigation system allows the robot to perform four types of motions: Move forward, Rotate an angle, Follow obstacle contour to its right, and Follow obstacle contour to its left.

2.2 The environment

The terrain to be explored is a finite planar area populated with a finite but unknown number of obstacles of arbitrary shapes and unknown locations. The obstacles need not be visible from each other. The boundary of an obstacle is a simple closed curve. The terrain limits can be defined either by walls or by a rectangular area of dimensions $L \times W$.

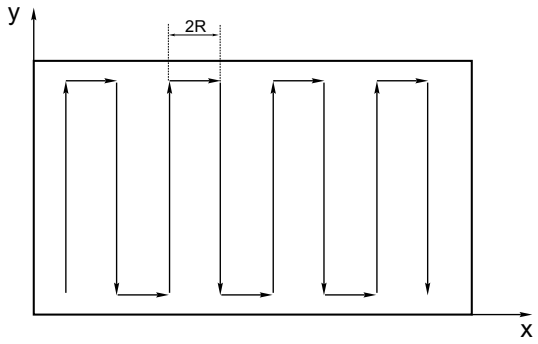

Fig. 2. Back-and-forth motions sweeping a cell from left to right

3 Complete coverage of unstructured environments based on critical-point detection

The complete-coverage algorithm that this work is based on relies on achieving an exact cellular decomposition of the environment while the robot generates a path of back-and-forth motions sweeping the field from left to right (see Figure 2). The exact cellular decomposition consists in dividing the field into regions or cells free of obstacles, so that the sum of resulting cells equals the total space free of obstacles. Then, each cell is completely covered by the robot with back-and-forth motions, and the coverage of the entire field results from assuring that the robot visits every cell. This has been well solved for known environments; however, two main problems have to be solved to achieve complete coverage of unstructured environments:

(1) On-line cellular decomposition based on sensed obstacles.
(2) Assurance that the robot visits every cell in the mine field.

Choset's method for achieving complete coverage in unknown spaces solves for both problems simultaneously during robot motion. The robot starts covering the space with back-and-forth motions until it detects an obstacle. When an obstacle is found, then a *critical point* is searched for. A critical point is a local minimum or maximum of the function:

$$h(x, y) = x, \quad \forall x, y \in \mathcal{C} \tag{1}$$

where x, y are the robot's coordinates in the field's reference frame and $\mathcal{C}$ is the object's contour. If the distance along the field's x-axis that the robot covers while surrounding the obstacle equals the detection diameter $2R$, then the robot gives up searching for a critical point and continues with back-and-forth motions. However, if the robot senses a critical point, existing cells will be closed and new ones will be opened. Then the cellular decomposition consists in detecting all critical points in the field. In this sense the method is complete, because it guarantees that every critical point in the environment is detected.

At the same time, an adjacency graph is incrementally constructed which has the overall information on already-visited cells.

3.1 Critical-point detection method

The critical-point detection method is depicted in Figure 3. The robot starts covering the field from a point at the origin of the field's reference frame, which will be considered to be CP_1. At a given instant, the sensor detects an object in the robot's path. Then the robot changes its trajectory to follow the object's contour, $\mathcal{C}$, until it finds a critical point, named CP_2, or travels a horizontal distance of $2R$. If a local minimum is found and the obstacle is locally convex at this point, it corresponds to an IN critical point, and at this time the current cell is closed, and two new cells are opened. An IN point opens two new cells, while a local maximum (OUT point) closes two existing cells. If the robot travels a horizontal distance of $2R$ around the obstacle and finds no critical points, it goes arround the obstacle back to the location where the obstacle was detected and continues sweeping the cell with back-and-forth motions. Figure 3(b) shows the detection of an IN critical point (CP_2), which closes one existing cell, C_1, and opens two new cells, C_2 and C_3, while Figure 3(c) shows the detection of an OUT critical point (CP_3), which closes two existing cells, C_2 and C_3, and opens one new cell, C_4. Choset also defines two more types of critical points for non-convex obstacles. When the obstacle is locally concave, we have a START point at a local minimum or an END critical point at a local maxima.

3.2 Adjacency-graph construction

The adjacency graph represents the critical points as nodes and the cells as edges. Each time a critical point is sensed, a new node is plotted. If the critical point is a minimum (IN point), two edges diverge, and if it is a maximum (OUT point), two edges converge at the new node. When the last corner in the field (named CP_4) is found, then the robot is guided to any critical point with disconnected diverging edges. Such edges represent cells not visited. The adjacency graph ensures that every cell in the mine field is visited by the robot. The right side of Figure 3 shows the incremental construction of the adjacency graph.

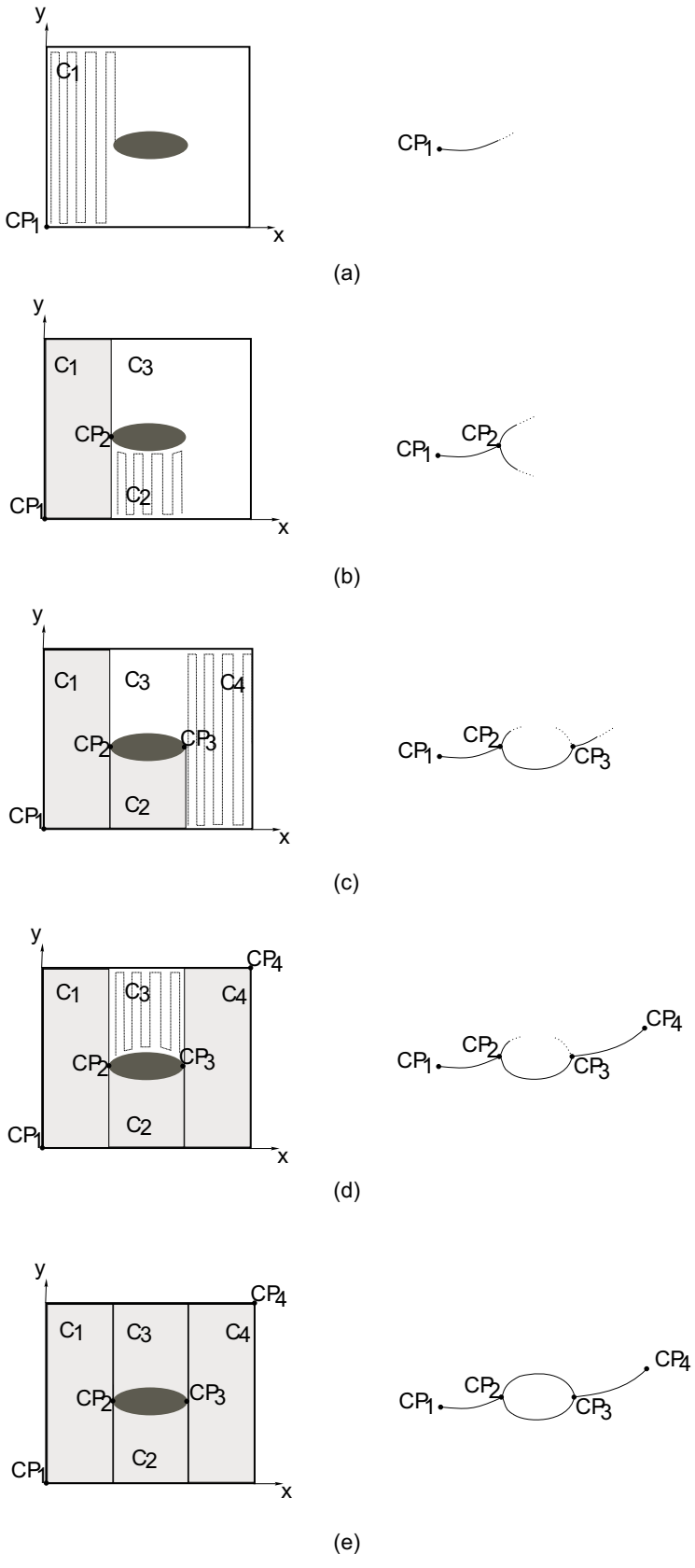

Fig. 3. Incremental cellular decomposition and adjacency-graph construction

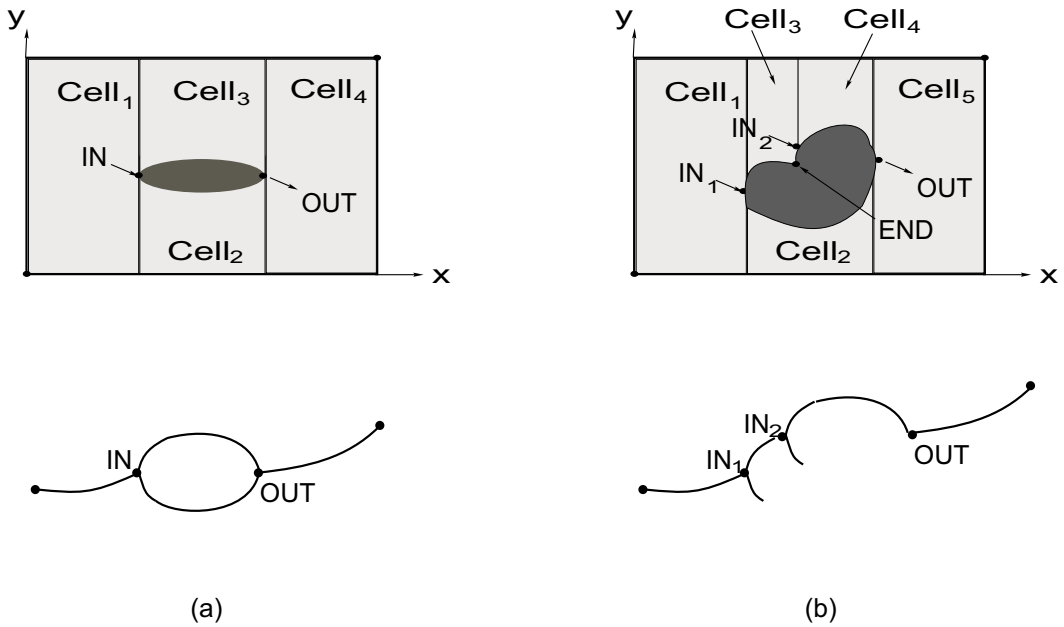

Fig. 4. IN and OUT critical points (a) for convex obstacles; (b) for non-convex obstacles

4 Enhancement of the method

The method for complete coverage of unstructured environments proposed by Choset is complete in the sense that the method guarantees that every critical point in the environment is detected. However, the method is not complete in the sense of covering the whole free space. There are certain cases where the method fails. It fails in the critical-point detection algorithm when an obstacle is not convex, it has more than one IN critical point but only one OUT critical point, and the END point where the obstacle is locally concave cannot be detected using range data because the boundary's curvature is smaller than the robot's periphery. Another problem of Choset's coverage algorithm relies on matching IN and OUT critical points that define a given cell. When the environment is unknown and critical points are detected by the way the robot sweeps the terrain, sometimes the last OUT critical point detected does not belong to the same obstacle as the last IN critical point found. Therefore an IN/OUT critical-point matching process is required. In this section these two problems will be covered in detail and an enhancement of Choset's method will be proposed.

4.1 Problems in on-line critical-point detection

A critical point is detected when a local extreme of equation (1) is found. A convex obstacle will normally have two critical points, one IN and one OUT,

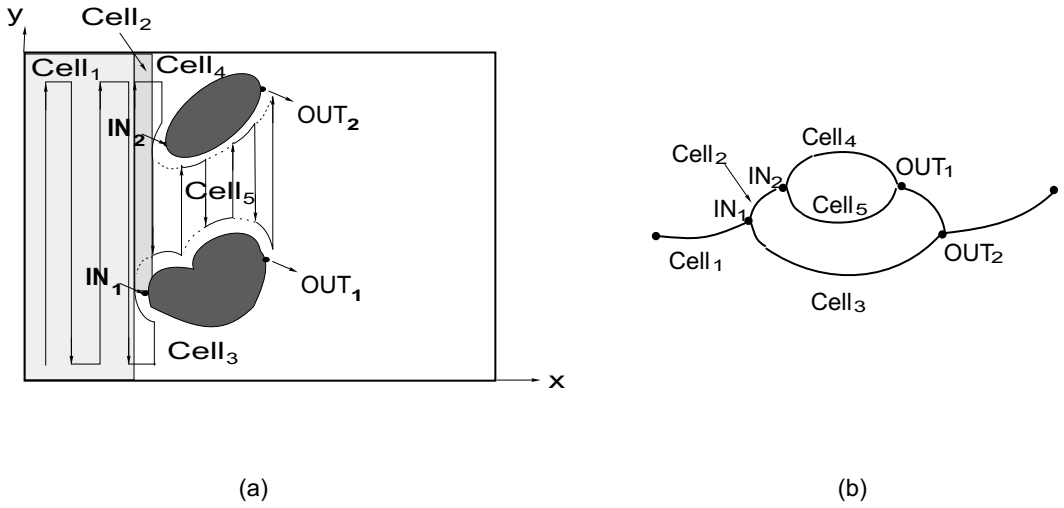

Fig. 5. Problems in completing the adjacency graph: (a) robot path; (b) adjacency graph

that define two cells (the upper cell and the lower cell), as shown in Figure 4(a). However, non-convex obstacles, as shown in Figure 4(b), might have more than one local minimum, that is, more than one IN critical point. Also an END critical point exists at the local maximum point where the obstacle is locally concave, however, as Choset advances in his paper, this END point cannot be detected by range data when the boundary's curvature is smaller than the robot's periphery. When this happens a problem arises when attempting to generate the adjacency graph, because one IN critical point matches one OUT critical point, defining two cells, but the remaining IN critical point opens two new cells that do not actually exist, and the algorithm will fail when it attempts to detect the corresponding OUT point. It is clear that the second IN critical point needs to be removed from the robot's memory to let the algorithm succeed. If that is done, only critical point IN_1 in Figure 4(b) would be considered, and Cell_3 and Cell_4 would be merged as one. To solve this problem, we propose two changes in the coverage algorithm:

Definition 4.1 *An IN critical point is the minimum of all local minima of the function* $h(x, y) = x, \quad \forall x, y \in \mathcal{C}$.

Definition 4.2 *An OUT critical point is the maximum of all local maxima of the function* $h(x, y) = x, \quad \forall x, y \in \mathcal{C}$.

With these two modifications, only one IN and one OUT critical point will be detected per obstacle, solving the problem of previous approaches.

4.2 Problems in completing the adjacency graph

In the critical-point detection process in unstructured environments, the robot searches for critical points while sweeping the environment from left to right with back-and-forth motions. When an obstacle is found in the robot's trajectory, a critical point is searched for while the robot follows the obstacle's contour. If a critical point is then detected, the robot continues with back-and-forth motions until it reaches another obstacle and then searches for another critical point. This method of critical-point detection has proven to be very efficient in identifying critical points while the robot minimizes its trajectory. However, it is difficult to know which OUT critical point closes the cells opened by the last IN critical point found. Figure 5 shows an example where the robot first finds critical point IN_1, and therefore Cell_1 is closed and Cell_2 and Cell_3 are opened in the adjacency graph. The robot continues searching in Cell_2. The next critical point found is IN_2, which closes the current cell (Cell_2) and opens two more cells (Cell_4 and Cell_5). The problem arises when the robot detects the first OUT critical point, that is, OUT_1. This point belongs to the first obstacle, but the robot does not have that knowledge, and the adjacency graph assumes it belongs to the second one, closing Cell_4 and Cell_5. This is a big mistake, because the OUT point found should close Cell_3 and Cell_5. As a result, the environment will never be completed, because the robot will store the wrong information. To solve this problem, an IN/OUT critical-point matching process is here proposed. This algorithm is based on the following properties from computational-geometry theory [8]:

Property 4.1 *For every obstacle defined by a pair of (IN,OUT) critical points,* $x_{IN} < x_{OUT}$.

Property 4.2 *For a couple of obstacles, let us name the pair of IN/OUT critical points that define the first obstacle (*IN_1,OUT_1*), and the pair of critical points that define the second obstacle (*IN_2,OUT_2*). Then if* $y_{IN_1} < y_{IN_2}$ *then* $y_{OUT_1} < y_{OUT_2}$. *Otherwise the obstacles will cross.*

The new method is as follows:

While the robot is detecting critical points, they are stored in an array CP of n_{CP} elements until the last critical point is found. Then the following process is carried out:

`Do while` n_{CP} `is greater than` 0`:`

`Step 1: Search for the first OUT point in` CP`. Let us name it` OUT_1`.`
`Step 2: Search for all IN points in` CP `such that` $x_{IN_i} < x_{OUT_1}$ `is true. Let us name the array of those critical points` CP_{IN} `and the number of elements in` CP_{IN} n_{IN}`.`

Step 3: If n_{IN} equals 1 then name the IN critical point IN_1 and go to Step 8.
Step 4: Order the elements in CP_{IN} such that $y_{IN_i} < y_{IN_j}$ if $i < j$.
Step 5: Search for all OUT points in CP such that $y_{OUT_i} < y_{OUT_1}$ and store them in an array CP_{OUT}.
Step 6: Order the elements in CP_{OUT} such that $y_{OUT_i} < y_{OUT_j}$ if $i < j$.
Step 7: Name the first element of CP_{IN} IN_1 and the first element in CP_{OUT} OUT_1 .
Step 8: IN_1 and OUT_1 are a pair of critical points and define two cells. Assign OUT_1 to IN_1 and remove the pair from CP.

Once the IN/OUT critical-point matching process is finished, every IN point has a corresponding OUT point, and the process of sweeping the unvisited cells can be completed successfully. If new critical points were detected while sweeping unvisited cells, the matching algorithm must be executed again.

4.3 Finding the shortest path to critical points

The two subsections above improve previous coverage algorithms in unstructured environments. Inserting them into Choset's method ensures the completeness of the coverage process. However, to enhance the method's performance, one last consideration is required. Once the last x-coordinate of the environment has been visited, the robot needs to complete the adjacency graph to detect unvisited cells. Once the unvisited cells are determined by using the herein improved algorithm, the robot must be guided to those critical points that define the unvisited cells. There are some applications, like humanitarian de-mining, where the robot's path from its position to the critical point must be carefully planned, because the robot should not pass through unvisited cells. It should move along a path traversing the already-known space towards an IN critical point where the unvisited cell starts. Also, for performance enhancement, this path must be minimized. Here, a method for obtaining the minimum path from the robot's position to the unvisited IN critical point is also proposed based on visibility graphs [8]. First the unvisited area is delimited by polygons, and the visibility graph is constructed to determine the shortest path to the critical points. As long as unvisited cells are being visited, the set of polygons must be updated to increase the area where the path can be planned.

4.3.1 Polygon generation

Once the robot encounters the last x-coordinate in the environment, and after running the IN/OUT critical-point matching algorithm proposed in subsection 4.2, it only has the knowledge of IN/OUT critical-point pairs defining obstacles. However some information is needed to let the robot know the size of the corresponding unvisited cell. This information is necessary to complete the set of polygons that cover unvisited areas. Figure 6(a) shows an environment decomposed into cells. Five of them have already been visited (shaded cells) and still two unvisited cells remain (Cell_3 and Cell_4). At this time the robot does not have knowledge of the dimensions of unvisited cells; however, during the robot's trip around the obstacles' contour, the robot did have that knowledge. Therefore, we insert another new modification into the coverage algorithm.

Let us define two new types of critical points:

Definition 4.3 *A* **SUP** *critical point is the maximum of local maxima of the function* $g(x, y) = y, \quad \forall x, y \in \mathcal{C}$.

Definition 4.4 *An* **INF** *critical point is the minimum of local minima of the function* $g(x, y) = y, \quad \forall x, y \in \mathcal{C}$.

When the robot is following the obstacle's contour, it only senses one INF or one SUP critical point, because it only covers one of the two cells defined by an obstacle. Therefore, the INF or SUP critical point detected is assigned to the last IN critical point found (there is no need for matching processes here). Knowledge of the INF or SUP critical point lets the robot know if the unvisited cell is over or under the obstacle (along the y coordinate), and also the dimensions of the unvisited cell.

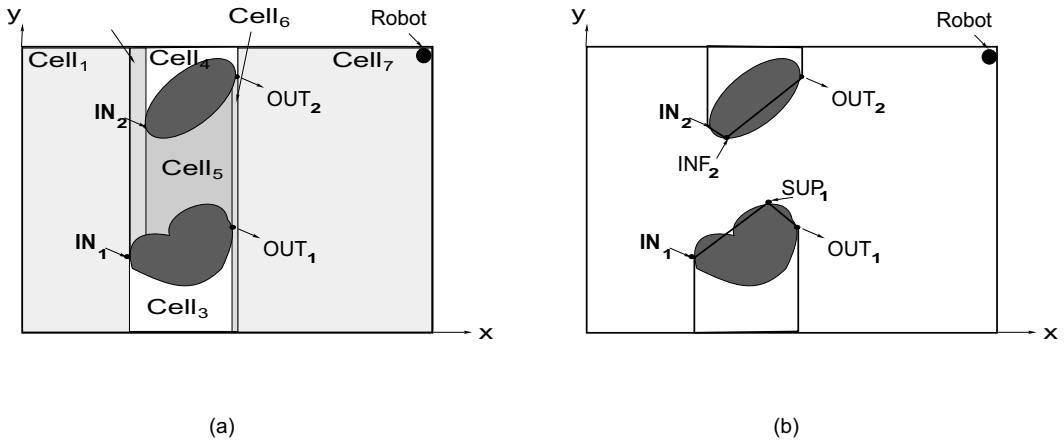

Fig. 6. Completion of the set of polygons of unvisited cells (a) cellular decomposition with unvisited cells in white; (b) Set of polygons

Definition 4.5 *If a SUP critical point is found, then the unvisited cell is under the current obstacle, that is, for every point* $P(x, y)$ *that belongs to the unvisited cell,* $y < y_{SUP}$.

Definition 4.6 *If an INF critical point is found, then the unvisited cell is over the current obstacle, that is, for every point* $P(x, y)$ *that belongs to the unvisited cell,* $y > y_{INF}$.

Now the polygon that contains the unvisited cell can be determined as the union of five points: (x_{IN}, y_{IN}), (x_{SUP}, y_{SUP}), (x_{OUT}, y_{OUT}), $(x_{OUT}, 0)$, $(x_{IN}, 0)$ if a SUP point is detected, or (x_{IN}, y_{IN}), (x_{INF}, y_{INF}), (x_{OUT}, y_{OUT}), (x_{OUT}, y_{max}), (x_{IN}, y_{max}) if an INF point is detected. Figure 6(b) shows the set of polygons that contain the unvisited areas.

4.3.2 The shortest path

Once the set of polygons that contain the unvisited cells is defined, the shortest path to IN critical points can be defined. To define the shortest path, again two properties of computational-geometry theory are considered (see Figure 7):

Property 4.3 *Any shortest path between two points* P_{start} *and* P_{goal} *among a set* $\mathcal{S}$ *of disjoint polygonal obstacles is a polygonal path whose inner vertices are vertices of S.*

Property 4.4 *The shortest path between two points* P_{start} *and* P_{goal} *among a set* $\mathcal{S}$ *of disjoint polygonal obstacles consists of arcs of the visibility graph* $G_{vis}(\mathcal{S}*)$*, where* $\mathcal{S}* = \mathcal{S} \bigcup \{P_{start}, P_{goal}\}$.

It is not within the scope of this paper to remind the reader how the visibility graph can be obtained. This information can be found elsewhere [9–11]. Then

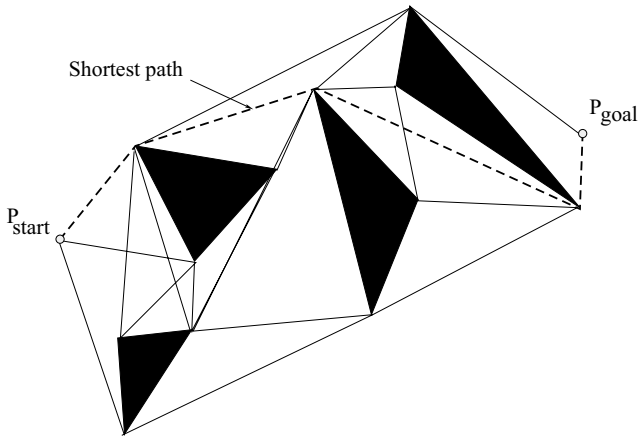

Fig. 7. Visibility graph and shortest path between two points among a set of polygonal obstacles

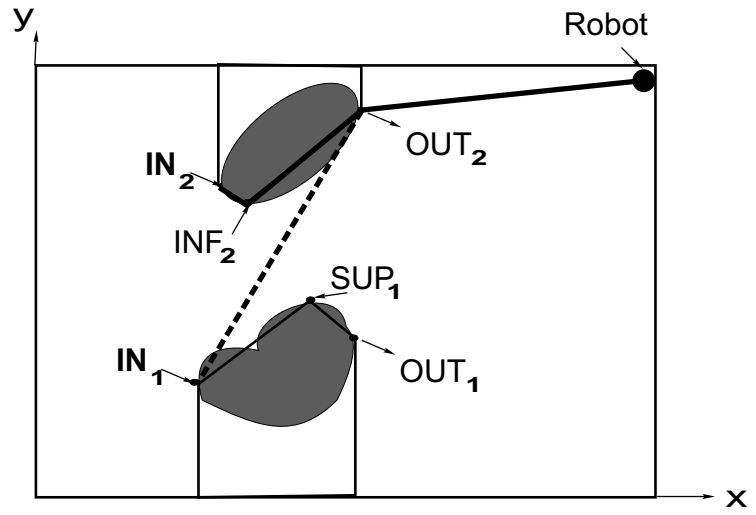

Fig. 8. The shortest paths from the robot's position to IN critical points using visibility graphs

the algorithm to cover unvisited cells is as follows:

Let us name the initial position of the robot before computing the shortest path (x_{rob}, y_{rob}). Then, the following steps have to be followed:

Do while the number of IN critical points on the adjacency graph is positive:

Step 1: Compute the shortest path to the last IN critical point on the adjacency graph.
Step 2: Follow the shortest path to the IN critical point.
Step 3: Sweep the current polygonal area S_n until the corresponding OUT point is found.
Step 4: Remove S_n from the set $\mathcal{S}$ of disjoint polygonal areas. Remove the last IN critical point from the adjacency graph.
Step 5: Let $x_{rob} = x_{OUT}, y_{rob} = y_{OUT}$. Go to Step 1.

Figure 8 shows the two shortest paths obtained for the example above. The thick continuous line depicts the first path from the robot's initial position to the last IN critical point on the adjacency graph, that is, IN_2. The thick dashed line depicts the shortest path from critical point OUT_2 to the remaining IN critical point on the adjacency graph, that is, IN_1. Note that the paths enter the obstacle area. It is assumed that the robot will follow the obstacle's contour until it reaches the path again.

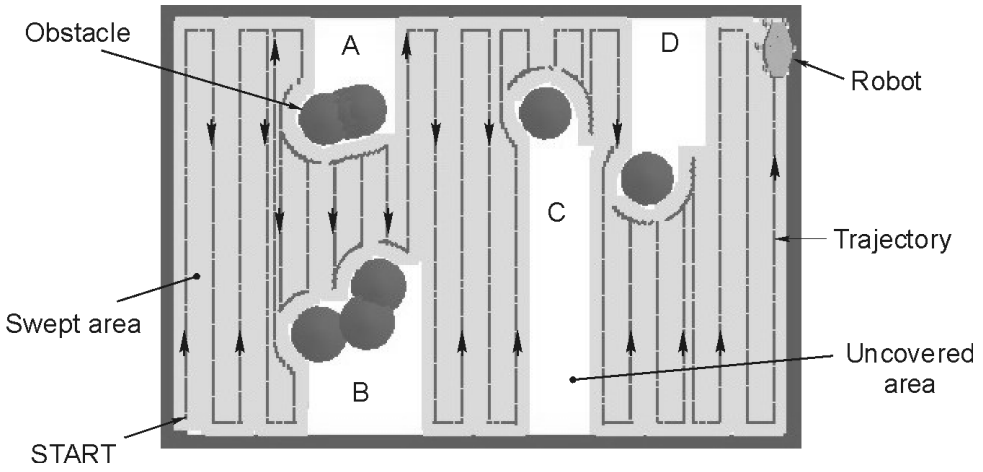

Fig. 9. Simulation results of cellular decomposition. Unvisited cells in white colour

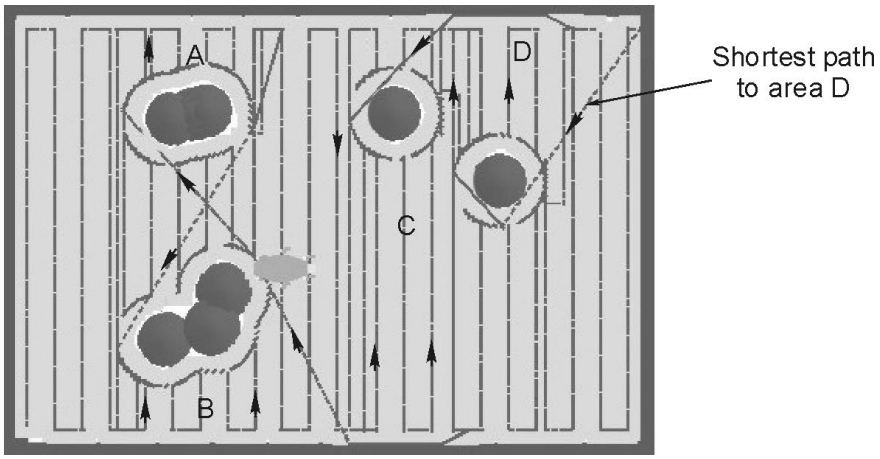

Fig. 10. Simulation results of the complete-coverage algorithm after applying the enhancements proposed

5 Experimental results and performance analysis

The proposed method for complete coverage of unstructured environments has been tested through simulation for a variety of situations (shape and number of obstacles). Different scenarios were obtained by generating obstacles using circles that were placed randomly inside the rectangular area that represents the field. This method of creating scenarios allows to generate complex obstacles. The method succeeded at achieving completeness in all scenarios. Figure 9 shows a complex scenario where a walking robot's path is shown by a dashed line and a shaded area represents the total swept area. Cell decomposition is performed and white areas show uncovered cells. Figure 10 shows the final result of applying the IN/OUT matching algorithm and the shortest-path method to cover the remaining cells. As may be observed from Figure 10, the algorithm detects all critical points and sweeps the whole free space.

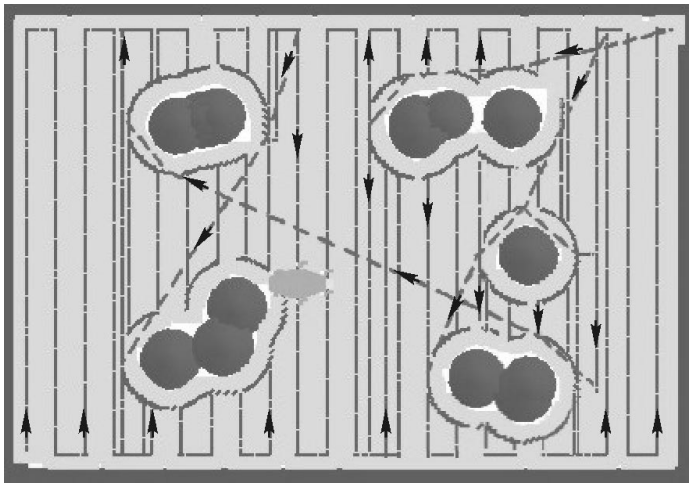

Fig. 11. Simulation results in complex scenarios

More complex environments have been tested and the method succeeded. For instance, when three or more obstacles have a common y coordinate, uncovered cells may include one or more obstacles, and therefore, new cells will be found while covering them. The proposed method can handle such environments. New critical points are found while sweeping unvisited cells. Then, the IN/OUT matching algorithm has to be executed again and new polygons are created. Figure 11 shows simulation results in such situations.

The performance of the algorithm, assessed in terms of the upper bound on the length P of the generated path as a function of obstacle perimeters and distance between obstacles, is:

$$P \leq Z + 2\sum_{i=1}^{n} p_i + \sum_{i=1}^{n-1} SP_{i,i+1} \tag{2}$$

where Z is the total path generated with back-and-forth motions, p_i is the perimeter of the ith obstacle, with n being the total number of obstacles found, and $SP_{i,i+1}$ is the length of the shortest path from obstacle i to obstacle $i+1$.

The total path generated with back-and-forth motions is upper bounded by:

$$Z \leq \frac{W}{2R} + L \tag{3}$$

which corresponds to the case of no obstacles. In this expression, W is the width of the field and L is the length of the field, with R being the scanning radius.

The performance of the algorithm is the same as Choset's provided that he used the shortest-path algorithm. However, note that Choset's method, al-

though the most efficient of the methods in the literature, is not complete for every possible situation, and the approach herein proposed is complete.

6 Conclusions

Some robotic applications require a complete-coverage algorithm to guarantee that the robot's path covers the whole obstacle-freee area. Prior coverage algorithms exhibit good performance when negotiating structured environments, but most of them fail in unstructured terrains. Some exhibit poor performance, while others fail to guarantee completeness of the scanned area. This paper describes some deficiencies of previous complete-coverage algorithms in unstructured environments and also proposes some relevant modifications to improve the method. The resulting method has been proven to successfully cover the whole area, and some simulation examples have been shown. Added to the completeness of the proposed method, the performance achieved is equivalent to that of the best previous approach. This method can be very useful in robotic de-mining applications or cleaning tasks.

Acknowledgements

This work has been partially funded by CICYT through Grant DPI2001-1595.

References

[1] A. Zelinsky, R. Jarvis, J. Byrne, S. Yuta, Planning paths of complete coverage of an unstructured environment by a mobile robot, in: Proc. Int. Conf. on Advanced Robotics, 1993, pp. 533–538, Tokyo, Japan.

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

[3] S. Hert, S. Tiwari, V. Lumelsky, A terrain covering algorithm for an AUV, Autonomous Robots 3 (2/3) (1996) 91–119.

[4] E. Rimon, J. Canny, Construction of c-space roadmaps using local sensory data – What should the sensors look for?, in: Proc. IEEE Int. Conf. Robotics and Automation, 1994, pp. 117–124, San Diego, CA.

[5] E. Acar, H. Choset, Sensor-based coverage of unknown environments: incremental construction of Morse decompositions, The International Journal of Robotics Research 21 (4) (2002) 345–366.

[6] E. Acar, H. Choset, Robust sensor-based coverage of unstructured environments, in: International Conference on Intelligent Robots and Systems, 2001, pp. 61–68, Maui, Hawaii.

[7] H. Choset, Coverage of known spaces: the boustrophedon cellular decomposition, Autonomous Robots 9 (2000) 247–253.

[8] F. Preparata, M. Shamos, Computational Geometry, An Introduction, Springer-Verlag, New York, 1985.

[9] H. Alt, E. Welzl, Visibility graphs and obstacle-avoiding shortest paths, Zeitschrift für Operations Research 32 (1988) 145–164.

[10] T. Asano, T. Asano, L. Guibas, J. Hershberger, H. Imai, Visibility of disjoint polygons, Algorithmica 1 (1986) 49–63.

[11] T. Asano, S. Ghosh, T. Shermer, Visibility in the plane, in: J.-R. Sack, J. Urrutia (Eds.), Handbook of Computational Geometry, Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000, pp. 829–876.

List of Figures