Proceedings of the 2003 IEEE/RSJ
Intl. Conference on Intelligent Robots and Systems
Las Vegas, Nevada · October 2003

A topological coverage algorithm for mobile robots

Sylvia C. Wong Bruce A. MacDonald
Department of Electrical and Electronic Engineering
The University of Auckland
New Zealand
E-mail: s.wong, b.macdonald at {auckland.ac.nz}

Abstract—**In applications such as vacuum cleaning, painting, demining and foraging, a mobile robot must cover an unknown surface. The efficiency and completeness of coverage is improved via the construction of a map of covered regions while the robot covers the surface. Existing methods generally use grid maps, which are susceptible to odometry error and may require considerable memory and computation. This paper proposes a topological map and presents a coverage algorithm in which natural landmarks are added as nodes in a partial map. The completeness of the algorithm is argued. Simulation tests show over 99% of the surface is covered; 85% for real (Khepera) robot tests. The path length is about 10% worse than optimal in simulation tests, and about 20% worse than optimal for the real robot, which are within theoretical upper bounds for approximate solutions to travelling salesman based coverage problems. The proposed algorithm generates shorter paths and covers a wider variety of environments than topological coverage based on Morse decompositions.**

I. Introduction

Coverage path planning is needed in a variety of applications such as vacuum cleaning, painting, humanitarian demining and foraging. In these tasks, a mobile robot must visit all reachable surface in an enclosed region. When given an unknown environment, the robot must use sensor information to avoid obstacles and to build a partial map to remember where it has been, so that it may return to cover remaining areas.

Coverage algorithms generally use uniform grid maps, where the value for each cell represents the probability of an obstacle [5]. Zelinsky *et al* [16] required a complete grid map of a known environment and used the distance transform to plan the coverage path. Ulrich *et al* [12] picked the direction with the most uncovered cells in a partial grid map as the new direction of travel whenever the robot reached a wall. Gabriely and Rimon [6] also used a partial grid map, but instead chose a new direction of travel at each cell. The first uncovered and unoccupied neighbour of the current cell found in an anti-clockwise direction was chosen as the next step. This behaviour created a spiral path that spiral outwards in open spaces. The grid based representation requires accurate localisation to create and maintain a coherent map. Also the size of the map grows quickly as resolution does not depend on complexity of the environment [11].

Topological maps represent the environment as a graph where landmarks are nodes, and edges represent the connectivity between landmarks [8]. Mammals appear to store spatial information topologically [10]. Topological maps are more compact than grid maps and they permit more efficient planning through the use of standard graph searches [11]. They also do not require accurate localisation. For example, Zimmer [17] successfully implemented a topological navigation and mapping system for a low budget platform with only light and touch sensors.

Very little attention has been given to applying topological maps for coverage tasks in known or unknown environments. One exception is Acar and Choset's [1] use of Morse decomposition, representing the cellular decomposition as a topological map. Each cell is covered in turn and the coverage is complete when the topological map is completely constructed. Landmarks in the maps are critical points in Morse functions, where a sweep algorithm uses surface normals of obstacles to create the topology. One inefficiency in the use of critical points is the need for a coverage pattern that includes retracing. In contrast, our approach uses corners as landmarks, and does not require retracing, so shorter coverage paths are generated. Moreover, a wider variety of environments is supported.

Some researchers approach the coverage problem with large teams of robots, using dynamic roadmaps to coordinate robots' behaviour over the desired region [3], or partitioning an environment dynamically without the need for global communication [7]. Our work focuses on coverage with a single mobile robot, which is more appropriate for tasks such as vacuuming. A team of vacuuming robots would be impractical and uneconomic in a domestic environment for example, whereas a team of inexpensive robots may be appropriate for a foraging task in a large environment.

II. Spatial Representation

It is important how the robot represents the surface it has already covered while it is covering an unknown environment. A classic approach to representing mobile robot environments is exact cell decomposition [9], which is commonly used in point to point navigation. Here, free

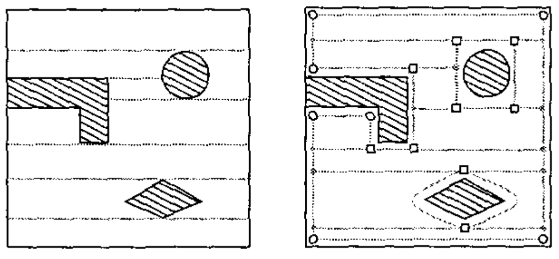

(a) (b)

Fig. 1. (a) Cellular decomposition formed with horizontal sweep line using our method. (b) Topological map representing this decomposition.

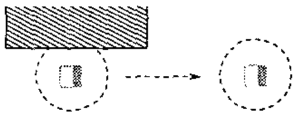


Fig. 2. Detection of convex landmarks using time series data. Dashed circle indicates the area covered by onboard sensors.

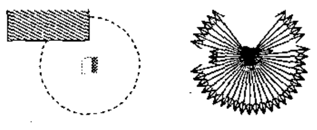

(a) (b)

Fig. 3. (a) Detection of convex landmarks from a single position. (b) Using a ring of 48 sonar sensors.

space in a robot's environment can be characterised by a decomposition into a collection of (non-overlapping) subregions. A connectivity graph which represents the adjacency relation among the subregions is constructed and searched for a path leading from a initial to a goal position. An example of exact cell decomposition is the trapezoidal decomposition [4], where a line L is swept across the environment and creates subregion boundaries whenever a vertex is encountered. Each subregion of the trapezoidal decomposition is thus either a trapezoid or a triangle.

In our work, exact cell decomposition is used for coverage rather than finding a point to point path. If the sweep line L intersects a subregion at exactly two points, then the subregion can be fully covered by a zigzag pattern that is parallel to L. The coverage of the environment can therefore be achieved by covering all subregions. A suitable decomposition can be formed by creating subregion boundaries whenever L enters or leaves an obstacle, as shown in Fig. 1(a).

The trapezoidal decomposition is not used in this work because of the following two reasons. Firstly, trapezoidal decomposition can only handle polygonal obstacles. Secondly, the decomposition formed using our method has larger subregions than trapezoidal decomposition. For example, two subregions will be formed to the left of the quadrilateral obstacle in the bottom of Fig. 1(a) using trapezoidal decomposition compared to only one here. Even though the subregion created in our method is a non-convex polygon, it can be fully covered by one zigzag pattern.

Rather than calculating the decomposition ahead of time from a known map, we assume the map is unknown and in this paper we show how the decomposition can be constructed online, using natural landmarks, as the robot progresses to cover the unknown environment.

The decomposition can be represented with a planar graph G, which consists of a set of nodes $N(G)$, and a set of edges $E(G)$, illustrated in Fig. 1(b).

There are four types of nodes in $N(G)$ – concave (○), convex (□), unexplored (×) and joint (•). Concave and convex are natural landmarks in the environment. Unexplored nodes represent directions leading to subregions detected and to be covered later. Joint nodes are needed to join different landmarks together along boundaries.

Concave landmarks exist when obstacles are on two adjacent sides of the robot. Convex landmarks exhibit a discontinuity on one side. Depending on the sensors present, convex landmarks can be detected in two different ways. If the robot can detect range at very few different angles or has only proximity sensors, a time series approach can be used. This is illustrated in Fig. 2, which shows a wall to the north of the robot disappearing. If the sensor information is significantly long range and rich, the discontinuity can be detected from a single position, as shown in Fig. 3 [14].

Edges store the types of motion required to travel between nodes they are incident upon. For example, whether the edge is next to a wall and which side the wall is on. They also store estimated distances separating the two nodes they connect.

III. Topological Coverage Algorithm

To completely and efficiently cover an environment, the robot must cover the current subregion while recording any neighbouring subregions in G so that it may subsequently cover them as well. The overall path through the environment should minimise travel between subregions.

The algorithm is implemented as a finite state machine with three states – boundary, normal and travel. Fig. 4 shows the transition between the states. The boundary state handles the situation where the robot is on the border between two different subregions. The normal state directs the robot to cover all floor area within a subregion using a zigzag pattern. Finally, the travel state generates paths for moving between subregions.

It is assumed that the robot starts the coverage process from a corner of the environment. This means the execu-

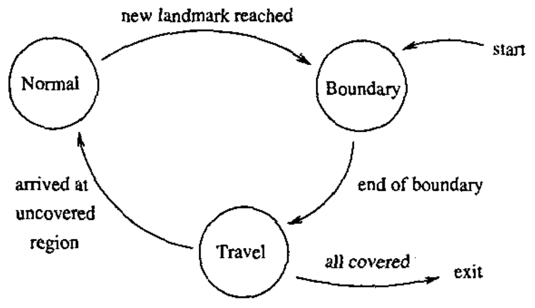


Fig. 4. State transition diagram.

tion of the algorithm always starts in the boundary state. This restriction is not a shortcoming because it is easy to program a robot to seek a corner by using simple forward and wall following movements.

The operation of the normal state can be summarised as follows:

```
if at landmark then
  if landmark not in $G$ then
    update $G$
  end if
  state $\Leftarrow$ boundary
else
  zigzag coverage pattern
end if
```

Here the robot moves in a zigzag pattern until it arrives at a landmark, ie a concave or convex landmark. This is because landmarks are always on the borders of subregions. If the robot starts the zigzag pattern on the other border of the current subregion, reaching this border means a complete coverage of the current subregion. If the landmark has never been visited before, G is updated accordingly. The process of updating and maintaining G is described in section III-A.

The operation of the boundary state can be summarised as follows:

```
if at new landmark then
  update $G$
end if
if end of border strip then
  state $\Leftarrow$ travel
else
  move forward
end if
```

Here the robot moves along borders between subregions. This is to expose all subregions neighbouring a particular border. G is updated whenever the robot discovers a new landmark. When the robot reaches the end of the border, it switches to the travel state.

The following describes the operation of the travel state:

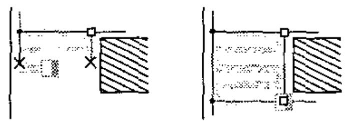


Fig. 5. Representation of a subregion being covered.

```
if just changed to travel state then
  $T(n) \Leftarrow$ search $G$
  if $T(n) = \emptyset$ then
    exit
  end if
end if
$n_g \Leftarrow$ first node in $T(n)$
if at $n_g$ then
  if $n_g$ = unexplored node then
    state $\Leftarrow$ normal
  end if
  $T(n) \Leftarrow T(n) - \{n_g\}$
end if
move towards $n_g$
```

When operation first enters the travel state, a breadth-first search is used on G to find the closest unexplored node. The search returns a list of nodes, $T(n)$, leading from the current node to the selected unexplored node. If the search returns an empty list, the environment is completely covered and the algorithm exits. To reach the selected uncovered subregion, the robot moves from one node in $T(n)$ to the next. A node is removed from $T(n)$ when the robot reaches the corresponding area. When the robot arrives at the last node in $T(n)$, ie the unexplored node, operation switches to the normal mode in the chosen uncovered subregion.

A. Representation Update

G is updated whenever a new landmark is visited. A new node is created to represent the newly found landmark. Unexplored nodes are added for each direction connecting the new landmark to newly discovered subregions. New edges and joint nodes may be needed to connect the new landmark and subregions to G.

In the normal state, the current subregion being covered has an arrangement of edges and nodes similar to that shown in Fig. 5(a). The subregion is bounded on one side by a horizontal edge. This horizontal edge is on the boundary where this subregion is discovered. It is also where the coverage of this subregion starts. The two vertical edges are connected to two unexplored nodes (×). When the robot reaches the border on the other end, a second horizontal edge is added and the nodes around the subregion become fully connected (Fig. 5(b)).

Sometimes both the top and bottom boundaries of an

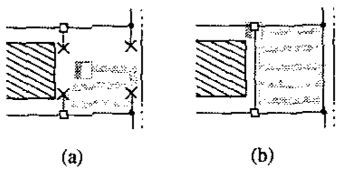


Fig. 6. A subregion where both top and bottom boundaries are known.

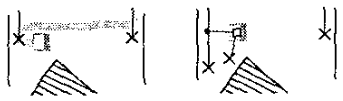


Fig. 7. Adding a convex landmark.

uncovered region have been exposed during two different boundary explorations. A situation where this can happen is around a free standing obstacle and one side of the obstacles has been covered. Fig. 6(a) shows a robot covering a subregion where both top and bottom boundaries are known. Note that there are two sets of unexplored nodes (×), one for the top boundary, one for the bottom. This duplication of unexplored nodes occurs because boundary exploration of both the top and bottom borders classify the subregion as uncovered. The robot does not know that both borders belong to the same subregion during the earlier boundary explorations. When the robot reaches the top border and arrives at the convex landmark (□), it is recognised as a previously visited landmark. The two sets of unexplored edges are merged to form a single subregion as shown in Fig. 6(b).

Fig. 7 illustrates the situation where the robot is covering from top to bottom and reaches a convex landmark. A new convex landmark (□) is created. This new landmark is linked to G through a new joint node (•). Unexplored nodes (×) are added to both the convex (□) and the joint node (•). This is because both nodes lead to the uncovered subregion underneath. Execution also switches to the boundary state after this update.

Fig. 8 shows a different example where the robot visits a new concave landmark. The unexplored node (×) is converted to a concave landmark (○). Since this new landmark does not lead to any new uncovered subregions, no unexplored nodes are added.

IV. Completeness of Algorithm

Assume the zigzag pattern covers a given subregion with strips in the horizontal direction. Then all reachable

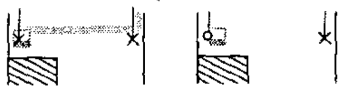


Fig. 8. Adding a concave landmark.

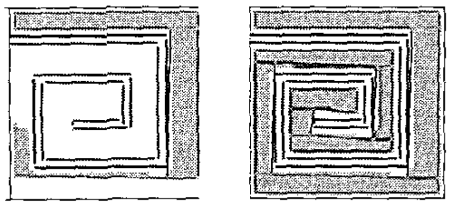


Fig. 10. Simulation of coverage of a spiral environment.

subregions share at least one horizontal boundary with another subregion. Those with only one shared boundary have obstacles as either their top or bottom neighbours.

All subregions coverage starts from a boundary. This is because the robot enters a subregion either from the top or bottom boundary to start the zigzag coverage pattern. The only exception to this is the initial subregion, where coverage starts on the obstacle side. All subregions intersect the sweep line L at exactly two points, hence the zigzag pattern will always lead the robot to the other boundary. The shared boundary will be fully explored to discover neighbouring subregions. Therefore any subregions neighbouring a known subregion will also be added to the G.

All reachable subregions share a boundary with another subregion that ultimately leads back to the initial subregion. This implies that all reachable subregions will always be discovered. The algorithm continues until $E(G)$ contains no unexplored nodes. Thus a subregion will be covered if it is added to G.

In conclusion, any subregion connected to the initial subregion, either itself or via other subregions, are added to G. All subregions in G are always covered. Therefore the algorithm achieves complete coverage of all reachable floor area for a given environment.

V. Results

The coverage algorithm has been tested in both simulation and on the miniature Khepera robot. The simulated robot is circular in shape and has 8 range sensors evenly distributed around its circumference. Fig. 9 and Fig. 10 show two sets of simulation results. Fig. 9 is a simple rectangular environment with two polygonal and one circular obstacles. Fig. 10 shows a more complex spiral environment.

The Khepera is equipped with 8 IR proximity sensors for detecting obstacles. The robot is 53mm in diameter and the IR sensors can detect objects up to 30mm to 40mm away. Dead reckoning is used to calculate the position of the robot. The situation then is analogous to a blind person trying to cover a room. It is impossible for the robot, or a blind person, to know whether it is following the horizontal strips of the zigzag cleaning pattern. This can be seen in Fig. 11(a), where the path taken is no longer

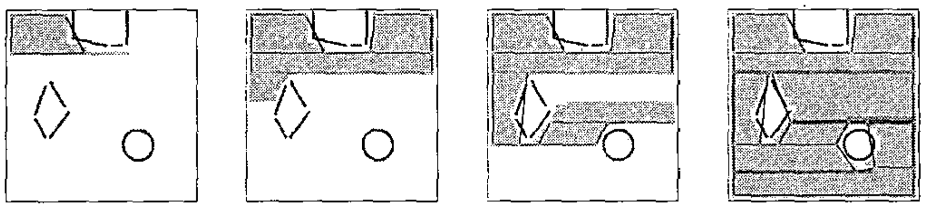
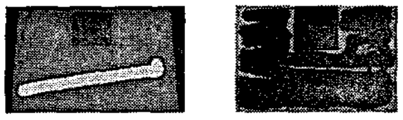


Fig. 9. Simulation results.



(a) (b)

Fig. 11. Condensed image showing (a) heading error with only proximity sensors. (b) path taken by robot.

horizontal. However, if the environment is rectilinear and reasonably populated, the robot can adjust its heading using wall following whenever it comes near an object. In this case, wall following is done by repeatedly moving closer to and further from the object because the relative angle to the wall cannot be calculated. Fig. 11(b) shows the result from one experiment. The robot was initially situated in the top left corner and facing right. In the top centre was a square obstacle. Thus the environment was divided into three smaller subregions – two smaller ones on top and one larger one below. The irregular shaped interior region is the path taken by the robot.

In [15], we proposed two performance measures for robotic coverage experiments. These performance measures evaluate the effectiveness and efficiency of coverage algorithms. Effectiveness is measured as the percentage of coverage, ie the ratio of covered area to reachable floor area. In simulation, this can be calculated as the ratio of the number of uncovered cells to empty cells. With a real robot, a video camera can be used to record the experiments. The captured frames are combined to create a condensed image, like the ones in Fig. 11. With the perspective distortion removed from the condensed images [13], the floor areas can be approximated as number of pixels.

The percentages of coverage of the simulations in Fig. 9 and Fig. 10 are 99.8% and 99.2% respectively. Note that cells adjacent to occupied cells are not accessible, thus leaving a white border around obstacles. Using pixel count on the condensed image in Fig. 11(b), the percentage coverage is found to be approximately 85%.

Efficiency of a coverage algorithm is measured as the comparison of the actual path taken, P_a, with the shortest path. However, finding the shortest coverage path is NP-hard [2]. Therefore, the actual path is compared with the idealised minimal path, P_m, instead. Efficiency is calculated as $\|P_a\|/(\|P_m\|\times C)$, where $\|P_a\|$ is the Euclidean distance of P_a and C is the amount of coverage. The idealised minimal path is the shortest coverage path for a mobile robot that can teleport with no cost associated with the teleport operation. All environments can be covered by such a robot with no retracing. P_m is therefore equal to or shorter than the realisable shortest path. $\|P_m\|\times C$ gives the minimum path scaled by the amount of coverage. This is important as a robot that poorly covers an environment but travels little will have a low $\|P_a\|/\|P_m\|$ ratio.

The efficiency of the simulations in Fig. 9 and Fig. 10 are 1.106 and 1.061 respectively. The efficiency for the real robot experiment in Fig. 11(b) is 1.20.

VI. Discussion

All the missed cells in simulations are along borders of subregions. In the boundary state, both wall following and move forward behaviours are carried out. The robot follows a path that spans more than one row. Some of the cells along the border are missed during the boundary state exploration. These missed cells will not be covered later if they are on the side of the border that belongs to a previously covered subregion. This is because G records coverage information using subregions. There is no direct representation of specific surface positions. As these missed cells are not recorded in G, the robot will not return to cover them later. However, the amount of missed cells is found to be small with coverage over 99% achieved.

Arkin et al. used travelling salesman tours for approximating coverage paths. They gave an upper bound of $1.325N$ for such tours on grid graphs with holes inside, where N is the number of cells to be covered [2]. As each cell is at a distance of one away from its 4-neighbours, a coverage path of length N happens when each cell in the environment is visited once and only once. The idealised minimal coverage path P_m used in the efficiency measure always has a length of N because it is defined to be the path with no retracing, ie it covers N cells in a path of length N. The upper bound of $1.325N$ in [2] indicates the length of travelling salesman coverage tours are at most 1.325 times longer than N. In terms of our efficiency measure, the upper bound is thus $\frac{\|P_a\|}{\|P_m\|} = \frac{1.325N}{N} = 1.325$.

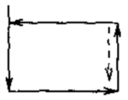


Fig. 12. Rectangular coverage pattern in [1]

This means the efficiency figures of 1.106 and 1.061 in the simulated experiments are within this bound. Note that the upper bounds calculated in [2] were for known environments, while the proposed algorithm works on unknown environments.

In [1], a rectangular pattern that repeats its traversal was used (Fig. 12). Due to the use of Morse decomposition, the coverage algorithm in [1] required wall following on the vertical boundaries of subregions to detect other subregions connected to the sides. In contrast, the proposed algorithm searches for new subregions in front and on both sides of the robot. This means that all area along the vertical boundaries are scanned for new subregions. As such the shorter zigzag pattern that does not include repeated traversal can be used. Therefore it generates shorter coverage paths compare to [1]. Also the algorithm in [1] cannot cover environments with obstacle boundaries parallel to the sweep direction L. This type of environment is supported in our algorithm as illustrated in the simulation shown in Fig. 10.

The travel between subregions are done by following the list of landmarks returned by the search on G. The search favours paths with landmark nodes (concave and convex) over joint nodes. Therefore the traversal between subregions is robust against odometry error and sensor and actuator noise [8].

VII. Conclusions

A topological based coverage algorithm is presented. It uses natural landmarks in the environment to construct a planar graph representing a decomposition of reachable surface into simple subregions that can be covered by a zigzag pattern. The grid based methods generally used for navigation in coverage applications can require considerable memory and computation and is susceptible to odometry error. On the other hand, the topological map used in this paper is proven to be robust against sensor and actuator noise. It is also compact as its resolution corresponds to the complexity of the environment.

Simulation results show the algorithm successfully covers diverse environments with a coverage better than 99% and efficiency not more than 110% of the optimal. The algorithm was also implemented on a Khepera robot with infrared sensors. Current heading corrections are made under the simplifying assumption of a rectilinear environment. Nonetheless it shows the algorithm realisable under real, inexact conditions. The percentage of coverage was measured at about 85%. The efficiency for the real robot was 120% of the optimal path length. The efficiency measures on the experiments show the path length generated to be within bounds of that approximated in [2].

A comparison is given to another topology based system [1]. Our work uses a simpler coverage pattern and generates more efficient paths. It also uses a simpler technique for landmark detection, thus enabling the coverage of environments with obstacles parallel to the sweep direction, such as rectilinear environments.

References

[1] Ercan U. Acar and Howie Choset. Sensor-based coverage of unknown environments: Incremental construction of morse decompositions. *International Journal of Robotics Research*, 21(4):345–366, April 2002.

[2] Esther M. Arkin, Sandor P. Fekete, and Joseph S. B. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry*, 17(1-2):25–50, 2000.

[3] O. Burchan Bayazit, Jyh-Ming Lien, and Nancy M. Amato. Better flocking behaviors in complex environments using global roadmaps. In *Proceedings of the Workshop on Algorithmic Foundations of Robotics*, December 2002.

[4] B. Chazelle. *Algorithmic and Geometric Aspects of Robotics*, chapter Approximation and Decomposition of Shapes, pages 145–185. Lawrence Erlbaum Associates, 1987.

[5] Alberto Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, RA-3(3):249–265, June 1987.

[6] Yoav Gabriely and Elon Rimon. Spiral-STC: An on-line coverage algorithm of grid environments by a mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 954–960, Washington, DC, May 2002.

[7] Markus Jäger and Bernhard Nebel. Dynamic decentralized area partitioning for cooperating cleaning robots. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 3577–3582, May 2002.

[8] Benjamin J. Kuipers and Yung-Tai Byun. A robust, qualitative method for robot spatial learning. In *Proceedings Seventh National Conference on Artificial Intelligence AAAI-88*, pages 774–779, St Paul, Minnesota, August 1988.

[9] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer, 1991.

[10] J. O'Keefe and J. Dostrovsky. The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely moving rat. *Brain Research*, 34:171–175, 1971.

[11] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99:21–71, 1998.

[12] Iwan R. Ulrich, Francesco Mondada, and J. D. Nicoud. Autonomous vacuum cleaner. *Robotics and Autonomous Systems*, 19(3–4):233–245, March 1997.

[13] George Wolberg. *Digital Image Warping*. IEEE Computer Society Press, 1990.

[14] Sylvia Wong, George Coghill, and Bruce A. MacDonald. Natural landmark recognition using neural networks for autonomous vacuuming robots. In *Proceedings of 6th International Conference on Control, Automation, Robotics and Vision*, Singapore, 2000.

[15] Sylvia C. Wong, Lee Middleton, and Bruce A. MacDonald. Performance metrics for robot coverage tasks. In *Proceedings Australasian Conference on Robotics and Automation (ACRA)*, Auckland, New Zealand, 2002.

[16] A. Zelinsky, R. A. Jarvis, J. C. Byrne, and S Yuta. Planning paths of complete coverage of an unstructured environment by a mobile robots. In *International Conference on Advanced Robotics ICAR*, Tokyo, Japan, November 1993.

[17] Uwe R. Zimmer. Robust world-modelling and navigation in a real world. *Neurocomputing*, 13(2–4):247–260, 1996.