Proceedings of the 2005 IEEE
International Conference on Robotics and Automation
Barcelona, Spain, April 2005

BSA: A Complete Coverage Algorithm

Enrique González, Oscar Álvarez, Yul Díaz, Carlos Parra, Cesar Bustacara
Grupos de Investigación SIDRe – SIRP
Pontificia Universidad Javeriana
Bogotá – Colombia
egonzal@javeriana.edu.co

***Abstract* – The Backtracking Spiral Algorithm (BSA) is a coverage strategy for mobile robots based on the use of spiral filling paths; in order to assure the completeness, unvisited regions are marked and covered by backtracking mechanism. The BSA basic algorithm is designed to work in an environment modeled by a coarse-grain grid. BSA has been extended to cover, not only the free cells, but also the partially occupied ones. In this paper, the concepts and algorithms used to extend BSA are introduced. The ideas used to extend BSA are generic, thus a similar approach can be used to extend most of the grid-based coverage algorithms. Finally, some simulation results that demonstrate that BSA performs a complete coverage are presented.**

Index Terms – Coverage Algorithm, Region Filling, Mobile Robots, Service Robots.

I. INTRODUCTION

The goal of robotics for the near future is to develop machines that collaborate with the humans in everyday tasks, thus generating new products that will be used massively. There are many emergent areas where robots can be used; however, some of the more promising ones include the automation of specialized tasks, at home or at work. In this kind of applications, the development is focused in the design of well-specialized robotic low-cost platforms that could be operated in a simple way by a non-qualified user. In some of these everyday tasks, the service robot must be able to autonomously perform jobs involving the filling/sweeping of the surface that is accessible. For instance, automatic floor cleaners, lawn mowers, agricultural robots, humanitarian demining and other surface maintenance or inspection machines must use an algorithm that drives a robot to cover completely its working area [1]. Some autonomous vacuum cleaners have been developed; between the most important commercial platforms, it can be mentioned the Roomba robot and Trilobite robot [2][3]. In these cases, the algorithm used to control the robot trajectory is based in a *brownian* motion, which is inefficient, even if it incorporates some structured paths. In these kind of applications, the goal can be reached in an autonomous way if the robot covers the entire accessible surface: this is the general statement of the coverage problem.

A path of complete coverage drives the robot to sweep all areas of free space in an environment in a systematic manner. The generated path is composed of multiple single paths that allow the robot to accomplish the global task. Other desired characteristics of a good coverage algorithm include that the trajectory length must be as short as possible and sweeping twice the same place must be avoided when possible. In order to accomplish a surface covering procedure, several problems, such as localization, map building, trajectory planning and motion control must also be solved.

The notion of completeness can be ambiguous; even if a complete coverage is guaranteed, different areas are effectively swept depending on the granularity of the model of the environment. When a coarse-grain model is used, the surface is seen as a grid, and the algorithm covers only the completely free cells; thus, some areas nearby the obstacles are not swept. Using a fine-grain surface model allows a better coverage; the surface is not partitioned, therefore the robot can cross closer to the obstacles. In practice, a more accurate approach to measure the quality of a coverage algorithm can be the effective coverage rate, the percentage of the accessible surface that is swept by the robot.

Some previous works have attempted to solve the region filling problem. Cao's algorithm [8] introduces the idea of using a well structured pattern to fill a region. Lumesky's spread seeder algorithm for terrain acquisition [6] can be adapted to fill a region, but it can generate very inefficient paths to get around obstacles. The indirect control algorithm [9] takes a potential field approach, it uses the concept of driving the robot near the unknown areas. The complementary regions algorithm [7], finds simple regions that can be filled by zig-zag like paths, while filling the region it looks for new adjacent regions that can be filled recursively. The algotithm proposed by Acar and Choset [10] is based in an exact cell decomposition, thus allowing to obtain a good effective coverage rate; however, the delimitation of the cells on-line is not an easy task. One of the most elegant solutions, based on spanning trees, was proposed by Gabrielly [11]; in the basic algorithm the coarse-grain cells are too big, thus generating low effective coverage rates, even if each one is visited only once; a modified version was proposed to reduce these problem, unfortunately, the modified on-line algorithm becomes too complex and makes the robot cover some cells twice. Other approaches include cooperative covering by a robot team [12] and the use of artificial markers in the environment [13].

The Backtracking Spiral Algorithm, BSA [4][5], is based on the execution of spiral filling paths, instead of "zig-zag" like paths used by most of the precedent algorithms [7][14][10]; in consequence, it is more robust to the robot´s initial orientation. BSA completeness is guaranteed by the backtracking mechanism included in the algorithm.

The simplicity of BSA allows the robot to work properly with low sensorial and computational demands. The generated paths can be easily executed by a differential drive platform, making BSA suitable for building low cost robots. BSA is an on-line algorithm, it builds incrementally a very simple grid like map in order to control the spiral filling procedure and the backtracking mechanism.

The basic BSA has been developed using a grid-like model. In this paper, an extension of BSA, that easily and naturally can cover surfaces nearby the obstacles, is introduced. BSA has been extended to fulfill fine granularity requirements, thus obtaining high effective coverage rates.

II. Basic BSA Algorithm

The BSA coverage algorithm was introduced in [5]. The basic algorithm uses a grid-based model, and assures the complete coverage of non-occupied cells; partially occupied cells are not considered. In this section, the conceptual framework of the basic BSA is introduced and some validation results are presented.

A. BSA Conceptual Framework

The BSA algorithm is supported by two main concepts: covering of simple regions using a spiral-like path and linking of simple regions based on a backtracking mechanism. A model of the environment is constructed incrementally as the robot moves. The surface is modeled by a coarse-grain occupancy grid, each square cell is of the size of the robot. Initially, all cells are marked as *unknown*; as the robot covers a cell, it is denoted as a *virtual obstacle*; cells where obstacles are detected, even if they are partially covered, are marked as a *real obstacle*.

A simple region is defined as an area that can be filled using a spiral path. In BSA, spiral paths are composed of concentric rings that form a continuous path from the region's boundary to a central ending point: Before starting a spiral path the robot is located nearby an obstacle, which is situated in the reference lateral side RLS. RLS indicates the relative direction where obstacles are to be referenced during the spiral filling procedure. RLS is fixed in advance and can't be modified. OLS is the opposite lateral side, it identifies the antipode of RLS. The following set of four simple reactive rules allow the correct execution of the spiral coverage procedure:

RS1 IF (obstacles_all_around)
THEN ending_spiral_point_detected

RS2 IF (NOT obstacle_in_RLS)
THEN turn_to(RLS) and move_forward

RS3 IF (obstacle_in_front)
THEN turn_to(OLS)

RS4 OTHERWISE move_forward

When these RS rules are evaluated, cells marked as *virtual obstacles* are considered as obstacles. Normally, the first external ring of a spiral path is performed nearby partially occupied cells, *virtual obstacles* are used as reference while covering internal rings.

A single spiral path is capable of partially covering the accessible areas. A backtracking mechanism is employed to get back to unvisited areas, where a new spiral filling procedure can be performed. Backtracking points (BP) are detected and stored during the execution of a normal spiral path. Before moving the robot forward, BSA evaluates if there is more than one unknown cell in its neighborhood; in other words, if there is an alternative possible path to follow. If this is the case, the cells that could be the starting point of future alternative paths are denoted as potential backtracking points.

Once the robot has reached the central ending point of a spiral filling path, the backtracking mechanism is invoked. First, the stored potential BPs are validated; BPs that are no longer marked as unknown cells are discarded. Then, one candidate point is selected; this selection can be based in different criteria, for instance the Euclidean distance between the robot and the BP. Finally, the robot reaches the selected BP using a path crossing over already visited cells, the ones that have been marked as virtual obstacles. At this point, the robot is ready to start a new spiral path beginning at the selected BP.

BSA stops when there are no more candidates BPs; which means that there are no more unvisited alternative paths, thus all the free accessible cells have been covered. The backtracking mechanism performs an exhaustive deep-first search, in consequence it guarantees the completeness of BSA. Nevertheless, the basic BSA covers only the cells that are completely free, which may not be appropriated for many applications. In fact, most of the practical tasks require to cover the whole area, even the surface nearby the obstacles. As the basic BSA is based in a coarse-grain model of the surface, even if it is conceptually complete, it is not complete for practical purposes. An example of the basic BSA coverage is shown in Figure 1. A more detailed explanation and discussion of the basic BSA can be found in [4][5].

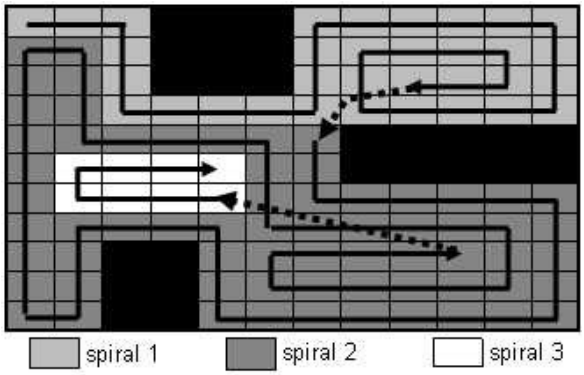

Fig. 1 Example of the basic BSA algorithm.

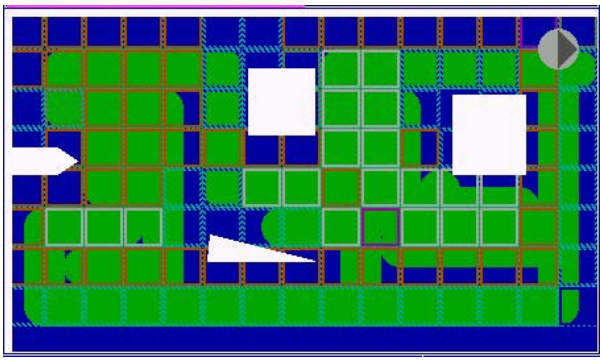

Fig. 2 Typical coverage of the basic BSA in the fine-grain simulator.

B. Basic BSA Results

The basic BSA was validated in three different simulation environments.

- In the coarse-grain simulator, the robot moves through the grid, thus allowing to perform debugging and performance evaluation in a well-controlled environment.
- In the fine-grain simulator, uncertainty is introduced in the detected state of cells; some simple reactive mechanisms were included in BSA to correct these problems. Figure 2 displays the results in this kind of environment; as can be observed, the free cells are covered, but the surface nearby the obstacles is not swept.
- Finally, the Saphira simulator was used [15], which also introduces real sonar models and motion uncertainty, demonstrating that BSA can be implemented efficiently as a set of parallel tasks, and that it could be implemented in a real robot.

Some experiments were performed in order to compare BSA with other algorithms [4]. The most meaningful performance measure was the percentage of multi-swept cells (MS). BSA performs better, as obtains 23% in the average MS measure, while CRA [7] obtains 91% and ICA [9] 307%. The other performance indicators, path length and energy consumption, are related to MS and have a similar behavior. The results show that BSA is very competitive when covering a grid-like environment.

III. Extension of BSA

An extension of the basic BSA was designed in order to improve the effective coverage. The solution is based on the idea that the partially-occupied cells are part of the external ring of the spiral path. These cells are covered by a wall-following procedure, the BSA rules to mark virtual obstacles and to detect backtracking points are adapted and extended, taking care of maintaining the original semantics of the algorithm. The most difficult problem to solve is how to get back to a selected BP, taking into account that the robot may need to cross over partially-occupied cells during this procedure.

A. General Approach

As a consequence of the coarse resolution in the representation of the surface, any algorithm that uses a grid-based approach to model the environment will produce an unsatisfactory effective coverage. A complementary strategy must be used to cover the surface nearby the obstacles. For instance, [16] performs a predictive wall-following procedure after all the free cells have been swept. This approach generates longer delays, a lot of cells are swept several times while the robot gets close to each of the obstacles. The main drawback of this approach is that the wall-following phase does not match in any way with the filling path used by the general coverage algorithm. It would be preferable that the coverage of the partially-occupied cells be integrated in a natural fashion into the filling path.

Most of the coverage algorithms use one of two basic structured filling paths: spiral or zig-zag like paths. As these filling paths are composed of well-defined and structured sequence of simple paths, it is easy to extend them in order to include the partially-occupied cells located in the frontier of the region filled by the path.

In some cases, the boundary of the region formed by the free cells covered by the basic spiral path is surrounded by partially occupied-cells. These cells can be covered by an initial wall-following procedure before sweeping the free cells. The original region is expanded; a new external ring is added to the spiral path; this ring is covered by a reactive wall-following procedure, then the original spiral rings are covered as in the basic path. However, notice (Fig. 1 and 2) that in some cases the regions generated while executing a spiral path can include “holes” in the middle of the region. Thus, when an obstacle is found while executing the spiral filling path, new internal boundaries appear, the region must be immediately expanded to include them and assure complete coverage. The expansion must be performed on-line; as soon as an obstacle is found, a wall-following procedure must be executed.

A similar approach can be applied for other structured filling paths. The BSA extension aims to partially validate this hypothesis.

B. Components of the Extended BSA

The extended version of BSA must respect the basic conceptual basis of the algorithm: spiral paths and backtracking mechanism. These requirements are achieved if, even when covering partially-occupied cells, virtual obstacles are correctly marked and all potential backtracking points are detected and stored. In practice, cells must be marked as a virtual obstacle while the robot is executing a wall-following procedure; this marking permits to identify if an obstacle has been previously visited. In the same way, the detection of alternative paths must be performed while circumnavigating an obstacle.

As it was pointed out in the general approach, the idea is to expand the covered regions by performing a wall-following procedure when an obstacle is found during the execution of a normal spiral filling path. In consequence, the expanded BSA includes two components: a modified grid-based spiral path functionality and a special wall-following procedure. The algorithm switches from one component to the other as unvisited obstacles are detected and visited. While executing a spiral path, the algorithm continuously looks for obstacles that have not been visited; if an unvisited one is detected, the wall-following component is activated in order to circumnavigate the obstacle. While executing a wall-following procedure, the algorithm continuously verifies if the obstacle has been completely circumnavigated; if this is the case, the spiral path component is activated.

The only modification that must be included in the basic BSA is to add a call to the wall-following procedure when a real obstacle is detected. In order to avoid the circumnavigation of already visited obstacles, the cell where the obstacle is detected must be marked as unknown.

C. Wall-Following Procedure

The new component of the extended algorithm follows in a reactive fashion the obstacle's boundary by keeping the robot at a predefined security distance from it. A closed control-loop generates motion commands that aim to align the robot with the boundary, while trying to maintain the robot at the desired distance. Before starting, the robot is moved nearby the obstacle and aligned, the obstacle must be in the reference lateral side (RLS), and the starting position of the robot is stored. The component must be stopped when the robot gets back in an area around the stored starting point. In order to comply with the BSA basic mechanism, virtual obstacles and backtracking points are detected and marked while executing this procedure.

D. Return Path by a Virtual Pipe

When the ending point of a spiral filling path is found, BSA must validate and select a backtracking point (BP). A return path must be generated from the ending spiral point to the selected BP. In the basic BSA procedure, only already visited free cells are considered when looking for a return path in the grid map. A simple breadth first search algorithm is used to obtain the shortest path. As the robot has visited the candidate cells, it is always possible to find a return path.

However, in the extended BSA, it can occur that it is not possible to find a return path composed exclusively of free cells. When this situation occurs, partially-occupied cells must be also considerer as candidates while finding the shortest path. An additional extension, included in the return path planning algorithm, is that a cost is associated to each candidate path that is found.

The cost not only takes into account the path's length, but also it assigns a lower cost to paths that include fewer partially-occupied-cells. Thus, giving preference to the paths mainly composed of free cells.

In consequence, in the extended BSA, a return path is formed by a sequence of cells, where at least the first and last ones are free; the cells in the middle of the path can include partially-occupied cells. In fact, the obtained path can be seen as a sequence of free cells that can include gaps, which are subsequences of partially-occupied cells. When moving along the path, there isn't any exceptional requirement while traversing free cells, however a specialized treatment must be applied in order to cross over gap sequences.

A reference line is defined along the central points of the cells that compose the path; this line could be seen as the desired path that must be followed by the robot. When a gap is found, an obstacle will obstruct the desired path. In order to avoid the obstacle, a wall-following procedure is started, the robot circumnavigates the obstacle until it returns to the reference line; at this point, the normal procedure, moving along the reference line, can be executed.

The cells included in a gap sequence include a valid path to pass between two free cells. However, it is not possible to know in advance if the wall-following procedure must be performed with the reference obstacle to the left or to the right side of the robot. In order to solve this problem, a virtual pipe area is defined around the reference line. While performing the avoidance wall-following procedure, the algorithm verifies that the robot does not exit the virtual pipe; when this condition is detected, the robot must perform the wall-following procedure in the opposite direction.

A typical example of a return path that includes gap is presented in figure 3. In the shown situation, the gap must be traversed; if the robot turns to the left and tries to circumnavigate the obstacle having it to its right side, the central point of the robot will exit the virtual pipe; on the other hand, if it turns to the right, the robot will be able to return to the desired reference path, the avoidance wall-following procedure will succeed.

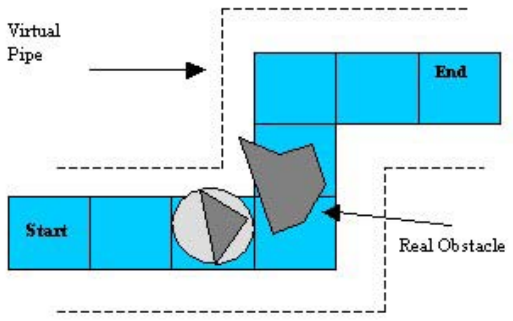

Fig. 3 Example of a virtual pipe.

IV. Experimental results

In order to validate and to evaluate the extended BSA, an experimental protocol, including five challenging scenarios was designed. The effective coverage rate is the main performance criteria that is evaluated. The experimental results obtained, running the extended algorithm in the fine-grain simulator, are summarized in table I.. As can be observed, the improvement in the effective coverage rate is notorious in all the cases.

In figure 4, the results of a typical coverage obtained by the extended BSA is shown. In this case, the same scenario of figure 2 is depicted, notice the improvement in the covered surface, the enhancement is about 30%.

V. Discussion

The basic completeness notion is a necessary condition that must be fulfilled by a coverage algorithm. However, as has been shown, it is not a sufficient condition to obtain an adequate performance in practical situations. The performance can be evaluated in a non ambiguous way by using the effective coverage rate, as proposed in this paper.

The approach used to extend BSA is general and can be applied to improve the performance of other grid-based algorithms that use structured filling paths. The advantage of this approach is that it is naturally incorporated in the original algorithm; the basic conceptual ideas of the algorithm are respected. Normally, the extension will include the invocation of wall-following procedures when unvisited obstacles are detected.

TABLE I
Comparative Results of Effective Coverage Rate

Id	Basic BSA	Extended BSA
1	78.75 %	93.56 %
2	64.89 %	92.91 %
3	62.71 %	93.41 %
4	67.37 %	92.29 %
5	66.28 %	93.29 %

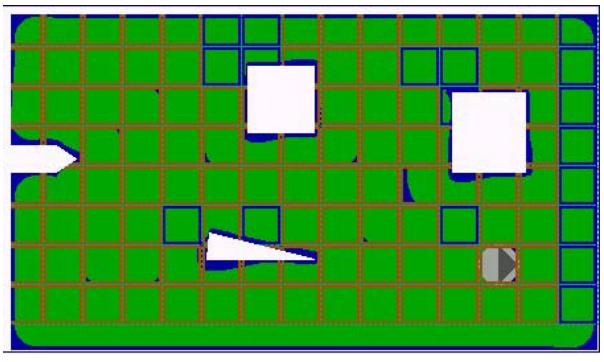

Fig. 4 Typical coverage of the extended BSA in the fine-grain simulator.

The results obtained demonstrate that BSA is an algorithm that can achieve a complete coverage of the accessible surface, not only taking into account a completeness notion, but also evaluating the effective coverage rate. Even if the extended BSA is more complex, it is still simple to implement, only some reactive rules must be added to invoke and to implement the wall-following procedure.

Actual work includes a more exhaustive comparative evaluation of BSA and other coverage algorithms. It is being implemented in a robot in order to validate that is feasible for real world tasks. This approach is based on simplicity, allowing to build low-cost robots, and looking forward to robot applications that collaborate with humans in everyday tasks, thus generating new products that will be used massively.

Acknowledgment

This project was funded by the Universidad Javeriana. The authors thank Camilo Otálora and Eduardo Gerlein for their contribution to this work.

References

[1] IEEE Robotics & Automation, "Technical Committee on Service Robots", http://www.service-robots.org, 2004.

[2] iRobots, "Official Roomba Homepage", http://www.roombavac.com, 2004.

[3] Electrolux, "Tribolite 2.0 Vacuum Cleaner", http://trilobite.electrolux.se/, 2004.

[4] E. González, M. Alarcón, P. Aristizábal, and C. Parra, "BSA: A Coverage Algorithm" Proc. IEEE-*IROS 2003*, pp. 1679-1684, 2003.

[5] E. González, P. Aristizábal, and M. Alarcón, "Backtracking Spiral Algorithm: A Mobile Robot Region Filling Strategy" Proc. ISRA *2002*, pp. 261-266, 2002.

[6] V.J. Lumelsky, et al, "Dynamic Path Planning in Sensor-Based Terrain Acquisition" IEEE Trans on Robotics and Automation, Vol. 6, No. 4, 1990, pp. 462-472.

[7] E. González, A. Suárez, C. Moreno, F. Artigue, "Complementary Regions: a Surface Filling Algorithm", ICRA96, pp. 909-914.

[8] Z.L. Cao, Y.Y. Huang, E.L. Hall, "Region Filling Operations with Random Obstacles Avoidance for Mobile Robots", Journal of Robotics Systems, Vol. 5, No2, 1988, pp. 87-102.

[9] A. Pirzadeh, W. Zinder, "A Unified Solution to Coverage and Search in Explored and Unexplored Terrains Using Indirect Control", IEEE, 1990, pp. 2113-2119.

[10] E. Acar, H. Choset, "Robust Sensor-based Coverage of Unstructured Environment", IROS'01, 2001, pp.61-68.

[11] Y. Gabrielly, E. Rimon, "On-Line Coverage of Grid Environments by a Mobile Robot", Technical repport Israel Institute of Technology, 2000.

[12] Z. Butler, A. Rizzi, R. Hollis, "Complete Distributed Coverage of Rectilinear Environments", 4th Int. Workshop on the Algorithmic Foundations of Robotics, 2000.

[13] M. Batalin, G. Sukhatme, "Efficient Exploration without Localization", ICRA'03, 2003.

[14] S. Hert, S. Tiwari, V. Lumelsky, "A terrain-covering Algorithm for an AUV", Autonomous Robots, 3(2-3), 1996, pp. 91-119.

[15] Active Media Inc., "Shapira Software Manual, 6.1", 1997.

[16] E. González, A. Suárez, C. Moreno, F. Artigue, "A Predictive Model for a Wall-Following Procedure of a Mobile Robot", SIRS96, pp. 57-64.