Robotica: (2021) volume 39, pp. 1141–1162. © The Author(s), 2020. Published by Cambridge University Press.
doi:10.1017/S026357472000096X

CPC Algorithm: Exact Area Coverage by a Mobile Robot Using Approximate Cellular Decomposition†

K. R. Guruprasad‡* and T. D. Ranjitha¶

‡*Department of Mechanical Engineering, National Institute of Technology Karnataka, Surathkal 575025, India*
¶*Telit Communications India Pvt Ltd, Bengaluru, India*
E-mail: ranjithatd.26@gmail.com

(Accepted September 10, 2020. First published online: October 7, 2020)

SUMMARY
A new coverage path planning (CPP) algorithm, namely cell permeability-based coverage (CPC) algorithm, is proposed in this paper. Unlike the most CPP algorithms using approximate cellular decomposition, the proposed algorithm achieves exact coverage with lower coverage overlap compared to that with the existing algorithms. Apart from a formal analysis of the algorithm, the performance of the proposed algorithm is compared with two representative approximate cellular decomposition-based coverage algorithms reported in the literature. Results of demonstrative experiments on a TurtleBot mobile robot within the robot operating system/Gazebo environment and on a Fire Bird V robot are also provided.

KEYWORDS: Coverage path planning; Exact coverage; Spanning tree; Cell permeability; Minor nodes; Clustered minor nodes; Mobile robot.

1. Introduction

Area coverage is an important task for mobile robots in many real-world applications such as land mine detection/demining,[1,2] spray painting,[3] autonomous floor cleaning,[4,5] lawn mowing,[6,7] manufacturing,[7] and structural inspection.[8] Apart from ground robots such as automated guided vehicles, the problem is also relevant to aerial robots such as uninhabited/unmanned aerial vehicles, where it may be used potentially for persistent surveillance, monitoring, or spraying pesticides in agricultural applications, or underwater applications using autonomous underwater vehicles. Here, a robot having an associated tool of a given shape, often corresponding to the relevant sensor or actuator range, must visit every point within a given bounded work-area, possibly containing obstacles. Depending on the application, the coverage tool could be a mine sensor, spray painting gun, vacuum cleaner, grass cutter, pesticide sprayer, or simply the field of view of a camera. Since the tool size is typically much smaller than the work-area, the robot's task consists of finding and moving along a path that will take the tool over the entire work-area. Therefore, coverage path planning (CPP) refers to the task of determining a path that passes over all points of an area of interest while avoiding the obstacles. A CPP algorithm (or a coverage algorithm for brevity) should cover an area *completely*, with minimal (or no) retraces (or overlap).

Several coverage algorithms have been proposed in the literature in the past to solve this problem. Survey of various coverage algorithms is provided in refs. [9] and [10]. In ref. [9], the author classifies

† Part of this work was carried as part of Masters' thesis of T. D. Ranjitha in the Department of Mechanical Engineering, National Institute of Technology Karnataka, Surathkal, Mangaluru 575025, India.

* Corresponding author. E-mail: krgprao@gmail.com

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

coverage algorithms as (a) off-line or on-line (sensor based), based on availability of a priori information about the area of interest, (b) heuristic or provably complete, and (c) approximate or exact, based on cellular decomposition. Gabriely and Rimon[11,12] proposed spanning tree-based coverage (STC) algorithms based on approximate cellular decomposition. These concepts have been extended and adapted to multi-robotic scenarios in refs. [13, 14]. However, we restrict our attention to single-robot coverage problems in this paper. Gonzalez et al.[15,16] proposed backtracking spiral algorithms (BSA) and extended-BSA based on spiral filling paths for covering a region decomposed into cells. In ref. [17], the authors combine exploration and goal-seeking problems, which may achieve complete coverage of the region, though coverage itself is not the problem being solved here. Choset ref. [18] proposes a Boustrophedon decomposition-based coverage (BDC) algorithm based on an exact cellular decompositioning scheme. Yang and Luo[19] proposed a CPP algorithm using neural networks. Xu et al.[20] provide an efficient coverage algorithm for aerial vehicles using Boustrophedon decomposition for a known arbitrary environment. Mannadiar and Rekleitis[21] provide an algorithm for optimal coverage of an arbitrary environment which is known a priori to the robots. Strimel and Veloso[22] proposed a coverage algorithm known as BC sweep algorithm, based on Boustrophedon decomposition, for a robot with battery or fuel capacity constraint. In ref. [23], the authors focus on a problem where the robot has limited battery (measured in terms of maximum distance it can travel when fully charged) and may have to come back to the starting location, get charged, and then continue coverage. Song and Gupta[24] present an approximate cellular decomposition-based online coverage algorithm incorporating exploration into the algorithm. An exploratory Turing machine generates a coverage path online using multi-scale adaptive potential surfaces. In many practical situations, complete a priori information about the environment may not be available, and hence online coverage algorithms are more suitable for autonomous robots. In this paper, we address an online CPP problem.

Most approximate cellular decomposition-based CPP algorithms[11,12,16,24] cover only cells completely free of obstacles, where the cell size is of the order of the robot/coverage tool footprint. Also, in most graph search-based coverage algorithms such as BSA[15] or extended-BSA,[16] based on approximate cellular decomposition, or BDC,[18] based on exact cellular decomposition, retracing of the path, due to the need for retracting and restarting on encountering a dead end, cannot be completely avoided. In the STC[11] algorithm, a graph is constructed by considering a combination of four cells, each of the size of the robot footprint as nodes, providing a return path through the cells that have not yet been covered, thus avoiding path retracing and, hence, avoiding coverage overlaps. However, with this algorithm, cells four times the size of the robot footprint are left not covered even when they are only partially occupied by obstacles. The full-STC algorithm proposed in ref. [12] improves the coverage resolution to that of the size of the robot footprint. In both these spanning tree-based algorithms, partially occupied cells are considered occupied and, hence, are not covered. Extended-BSA,[16] though based on approximate cellular decomposition, provides exact coverage. However, by its design, it cannot guarantee to avoid retracing of the path.

In this paper, we propose a *cell permeability-based coverage* (CPC) algorithm which will make the robot cover even the partially occupied cells, while reducing the amount of coverage overlap in comparison with algorithms such as full-STC[12] and completely avoiding retraces/coverage overlaps that arise due to necessity for retracting and restarting in algorithms such as those proposed in refs. [15,16] or online versions of BDC[1] algorithm.[9] These advantages are demonstrated in the paper by a comparison of the coverage path generated by the proposed CPC algorithm with that generated by the STC family of algorithms. We provide a formal analysis on the completeness of coverage with the proposed CPC algorithm with a discussion on its competitiveness. Basic ideas were presented in ref. [25]. This paper builds on the previous work and provides a detailed account of the CPC algorithm along with formal analysis.

2. The Problem Setting

In this section, we discuss the problem setting in detail. We consider a robot equipped with a coverage tool having a square footprint of sides D. The region of interest Q is a topologically connected bounded subset of $\mathbb{R}^2$, having a finite number of obstacles $O_1, O_2, \ldots, O_N$, with $O = \{O_1, O_2, \ldots, O_N\}$. The region of interest Q is enclosed within a (minimal) rectangular area Q_R

[1] Recent versions of these algorithms such as in ref. [20] attempt to avoid such scenarios when the region to be covered is known a priori.

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

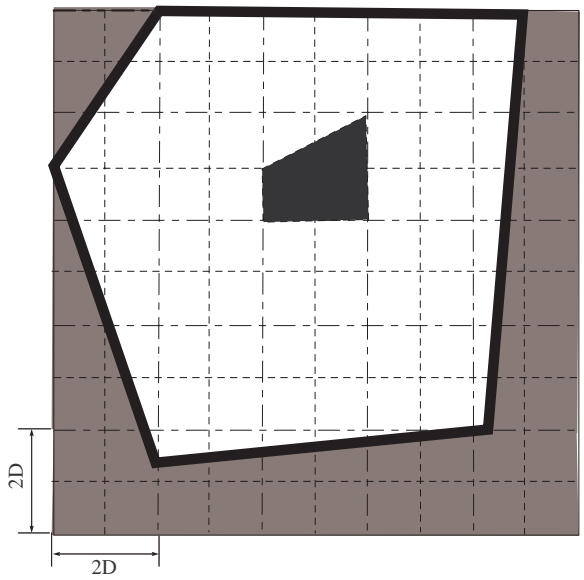


Fig. 1. The region Q to be covered is enclosed within a rectangular area of size $2mD \times 2nD$.

of size $2mD \times 2nD$, where m, n are integers. The enclosing rectangle is divided into $m \times n$ square cells, called major cells, of sides $2D$. Further, major cells are divided into four $D \times D$ sized sub cells. The space beyond the region to be covered, that is, $Q_R \setminus Q$ is considered to be a virtual obstacle. This is illustrated in Fig. 1 for $m = 5$ and $n = 5$. Major cells are shown as grids with long–short dashed lines and the subcells are shown as grids with dashed lines. An obstacle (black) is located at the center of the region. A subcell may be completely free (white cells), completely occupied by obstacles (black – if occupied by obstacle, dark grey – if completely outside Q), or partially occupied by obstacles/virtual obstacle (partly white and partly black/dark grey).

If a free space is embedded within an obstacle, it is considered as part of the obstacle, as this region is not accessible from any point within $Q \setminus O$. Similarly, two obstacles which have a gap through which a disc of diameter D cannot pass through are considered as a single obstacle. Now let us look at definitions of a few terms used in this paper.

Definition 1. If the region Q is divided into cells C_i, i$\in \{1, 2, \ldots, M\}$, and a robot path passes through all completely free cells and not through partially free cells, then such a coverage path is said to be *resolution complete*.

Definition 2. If the region Q is divided into cells C_i, i $\in \{1, 2, \ldots, M\}$, and a robot path passes through all cells, completely or partially free, such that the coverage tool covers entire $Q \setminus O$, then such a coverage is said to be *exact coverage*.

Resolution complete coverage converges to exact coverage in the limit as the size of cell size tends to zero. However, in approximate[2] cellular decomposition-based coverage algorithms, the size of the cells is the same as that of the robot (or coverage tool) footprint.

Definition 3. A region Q possibly containing a set of obstacles O is said to be *coverage conducive* by a robot with coverage footprint of $D \times D$ square, if $Q \setminus O$ is a contiguous region made up of union of $2D \times 2D$ sized square cells/grids.

Note that with a D sized coverage tool, it is not possible for any CPP algorithm to generate a complete coverage path with zero coverage overlap if the region is not "coverage conducive." Algorithms such as in ref. [16] or [18] can provide complete coverage of even non-coverage conducive regions, at the cost of non-zero coverage overlap. However, these algorithms may result in coverage overlap even for a coverage conducive region due to (i) need for retracting and restarting coverage (in refs. [16, 18]), and (ii) the fact that the decomposed cells can be non-conducive regions (in ref. [18]). But by design, the STC algorithm[11] might result in coverage gaps but ensures a non-overlapping coverage path.

[2] They are termed approximate cellular decomposition as Q is approximated by the union of completely free cells.

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

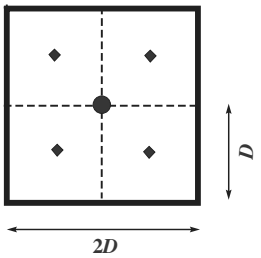


Fig. 2. A $2D \times 2D$ major cell with four $D \times D$ subcells.

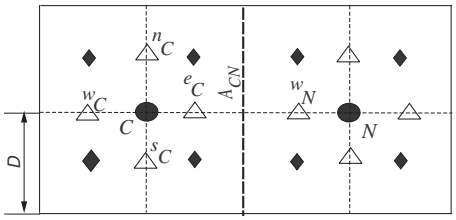


Fig. 3. A pair of major cells (labeled C and N) with the corresponding major nodes, subcells/nodes, and minor nodes.

The problem addressed in this work is to devise a coverage algorithm to achieve exact coverage of an arbitrary initially unknown region which may not be "coverage conducive." Since for such regions, no CPP algorithms can provide non-overlapping coverage, we try to achieve the complete coverage, and at the same time avoid (or at least reduce) the coverage overlap whenever possible. Also, the coverage algorithm should provide complete and non-overlapping coverage of regions that are "coverage conducive."

3. Adjacency Graph

Now we describe the adjacency graph which forms basis on which the coverage path is generated.

3.1. Major/subcells and nodes

Figure 2 shows a $2D \times 2D$ major cell with the $D \times D$ subcells embedded in it after the decomposition of the area as described in the previous section. The disc at the center of the major cell is the major node corresponding to the major cell shown. The subcells are indicated by $D \times D$ cells bounded by dashed lines. Subnodes corresponding to the subcells are shown by diamonds at the center of respective subcells. Note that the major and subnodes have physical existence in the sense that they correspond to the physical space $\mathbb{R}^2$. Further, the cells correspond to the physical (square) areas in $\mathbb{R}^2$ and nodes are primarily used for construction of the adjacency graph used for planning the coverage path.

3.2. Minor nodes

The concept of major and subcells and the corresponding nodes were introduced in ref. [11]. Now we introduce a new concept called minor node. Consider two adjacent major cells "C" and "N" as illustrated in Fig. 3 (by a slight abuse of notation, we use the same label to refer to both a major cell and the corresponding node). The common boundary A_{CN} between them is shown by a thick long dashed line. Corresponding to A_{CN} on either side, we introduce two minor nodes ${}^eC$ and ${}^wN$ shown as triangles, each associated with major cells (nodes) "C" and "N," respectively. As each major cell has four such boundaries on *north, east, west*, and *south* sides, we associate four minor nodes (shown as triangles) to each major cell/node. Here, ${}^eC$ denotes a minor node corresponding to the major cell (node) C and its boundary (A_{CN} in this case) on the **e**ast side. Further, each major cell (node) has four neighboring major cells (nodes) on its *north, east, west*, and *south* sides. We denote the major cell (node) of current interest as C and its neighbors on *north, east, west*, and *south* sides as N_n, N_e, N_w, and N_s, respectively. We denote a major cell as N without any subscript in situations when it is the only neighbor in question, for brevity (as in Fig. 3).

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

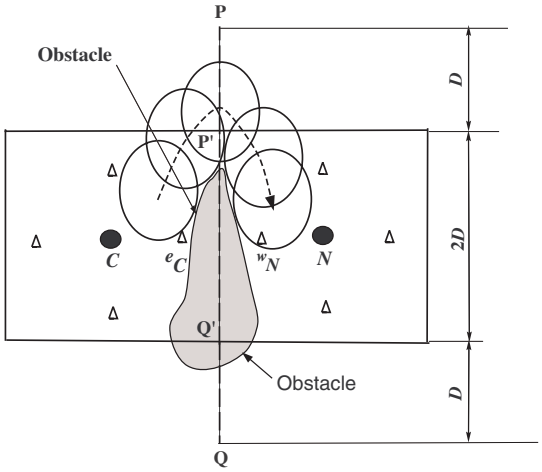


Fig. 4. Illustration of connectivity between two minor nodes.

Note that, unlike the major and subnodes, minor nodes do not have physical/spatial existence. They are associated with a major cell boundary. The minor nodes play a prominent role in the proposed algorithm. Unlike in the STC algorithm,[11] which considers the major nodes for the construction of a spanning tree, the proposed algorithm constructs a spanning tree over the minor nodes.

3.3. Minor node connectivity

Two minor nodes in question may correspond either to adjacent major cells or to a single major cell. Two minor nodes are adjacent if either they correspond to the same major cell boundary or they belong to (or associated with) the same major cell. First situation gives an inter-cellular connection, and the second situation gives an intra-cellular connection between the minor nodes. In the following, we discuss the inter-cellular and intra-cellular node connectivity.

Inter-cellular minor node connectivity: Two adjacent inter-cellular minor nodes corresponding to two adjacent major cells, say "C" and "N," are connected by an edge in the adjacency graph, if a robot can move from "C" to "N", through the corresponding common cell boundary A_{CN}. If a robot can pass through a major cell boundary, then we say that it is *permeable*. The major cell boundary may be free or occupied by an obstacle, partially or fully. In case it is completely free, it is permeable, and it is completely occupied, then it is not permeable. However, depending on the path planning algorithm, we may consider a partially occupied major cell boundary as permeable. In this work, we consider a major cell boundary as permeable, if the robot can pass through at least a portion of it. We define permeability as follows:

Definition 4. Let $\tilde{A}_{CN}$ be a line segment obtained by extending A_{CN}, the common cell boundary between major cells C and N, at both the ends by a length D. Now, if there is a free segment of length D in $\tilde{A}_{CN}$ (so that a robot of size D can freely pass), then we say A_{CN} is *permeable*.

If A_{CN} is permeable, then the corresponding inter-cellular minor nodes are connected by an edge in the adjacency graph formed by the minor nodes. Consider an example illustrated in Fig. 4. Here, an inter-cellular minor node adjacency graph edge is created between adjacent minor nodes ^{e}C and ^{w}N, corresponding to C and N, as the robot can go from C to N passing through at least a portion of A_{CN}. This is possible as there is a segment of length more than D in $\tilde{A}_{CN}$ (PQ), free of obstacles.

Minor node multiplication: As observed before, the minor nodes are related to possible robot paths between two major cells through the corresponding common boundary. There are three possibilities: First, there is no path leading to corresponding minor nodes being not connected (or equivalently, the corresponding minor nodes are marked "blocked"). Second scenario is that there is exactly one path connecting two major cells. In this situation, an edge is said to connect the corresponding minor nodes. Third situation is there could be multiple paths connecting two adjacent major cells through the common boundary because of the presence of obstacles. One such situation

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

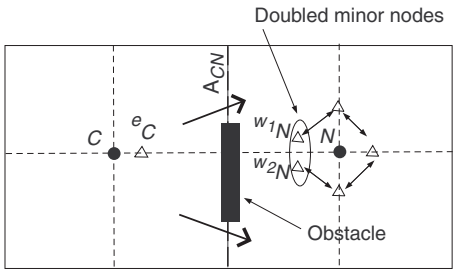


Fig. 5. Illustration of node doubling in the presence of an obstacle.

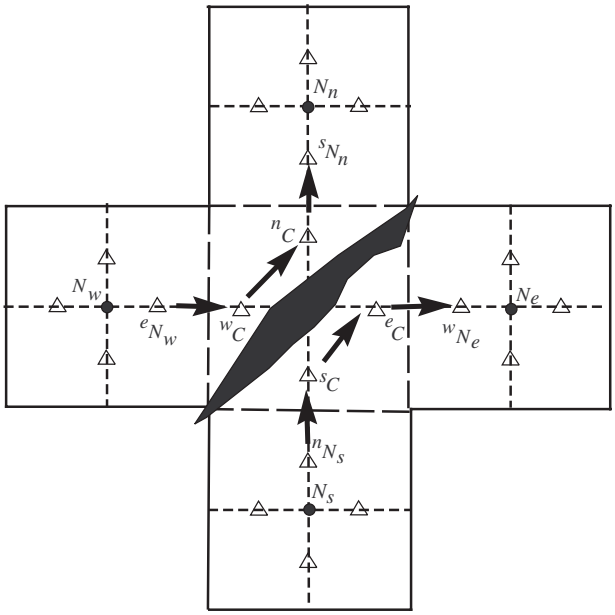


Fig. 6. Illustration of intra-cellular minor node connectivity. The black object at the center is an obstacle.

is illustrated in Fig. 5. Here, the minor node ^{w}N in the major cell N gets doubled (or split) as ^{w_1}N and ^{w_2}N. Similarly, the minor node ^{e}C too gets doubled. These situations occur when a major cell boundary is free, but has more than one connected component (we can show that we have at most three of them), and hence, there are multiple paths connecting two major cells. As minor nodes correspond to the possible connections/paths across a major cell boundary, such situations force us to duplicate a minor node. We call such situations as *node doubling (or tripling)*. Though these situations can be handled as special cases, in this paper, it is assumed that such situations do not arise.

Intra-cellular minor node connectivity: Similar to inter-cellular connectivity, minor nodes within a major cell can have different forms of adjacency graphs based on connectivity between the minor nodes, depending on the obstacle scenario. All four minor nodes within a major cell are considered adjacent. Two intra-cellular minor nodes ^{i}C and ^{j}C, $i, j \in \{n, e, w, s\}$, are connected by an edge in the intra-cellular minor node adjacency graph, if and only if both of them are connected to the corresponding minor nodes in the adjacent major cells, say N_l and N_m, $l, m \in \{n, e, w, s\}$, and the robot entering C from N_m can move to N_l. Figure 6 shows one scenario to illustrate the intra-cellular minor-cell connectivity. Consider two situations. First, let the robot enter the current cell C from N_s through A_{N_sC} using the graph edge between minor nodes $^{n}N_s$ and ^{s}C. Now the minor node ^{e}C is connected to ^{s}C as the robot can move from current position within C to cell N_e through A_{CN_e}. In this situation, minor nodes ^{n}C and ^{w}C are not connected to ^{s}C. Consider a second situation where the robot enters C from N_w through the edge $^{e}N_w \rightarrow {}^{w}C$. Now ^{w}C is connected to ^{n}C, but not to ^{s}C and ^{e}C.

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

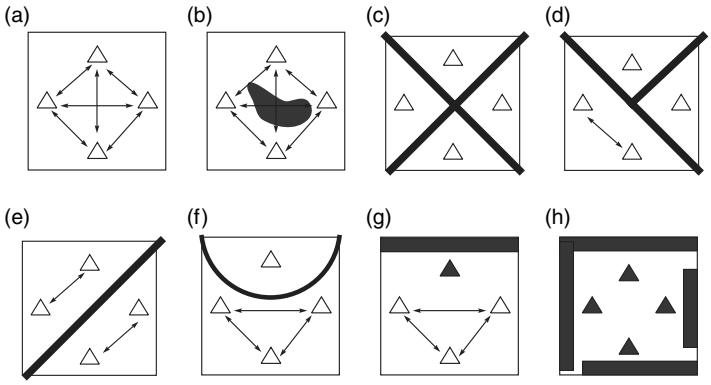


Fig. 7. Illustration of clustered minor nodes. Free minor nodes are shown with white triangles and blocked minor nodes are shown with black triangles.

Clustered minor nodes: Each major cell has four minor nodes associated with it. The intra-cellular minor node adjacency graph clusters the minor nodes into completely connected components. We call each maximal subset of the minor nodes (within a major cell), which is completely connected, as a *clustered minor node*. If a major cell is free, or, the free region within it is a single topologically connected set and all its boundaries are permeable, then all the four minor nodes form a single clustered minor node. We refer to a major cell with a single clustered minor node, as *effectively free* major cell. In general, there are at most four clustered minor nodes (as we do not consider minor node doubling or tripling here). Figure 7 illustrates a few possible scenarios. When all the four minor nodes associated with a major cell form a connected intra-cellular minor node graph as shown in Fig. 7(a) (a free major cell) and (b) (an effectively free major cell), a single clustered minor node is created. A similar situation but a cluster of only three minor nodes leading to a single clustered minor node occurs in a scenario illustrated in Fig. 7(g). In fact, whenever an obstacle free region within a major cell is a single contiguous region, a single clustered minor node is created. When the free region within a major cell is made up of several topologically disconnected regions, an equal number of clustered minor nodes are formed. In scenarios illustrated in Fig. 7(c) four clustered minor nodes (with each cluster containing exactly one minor node), (d) three clustered minor nodes $(2 + 1 + 1)$, (e) two clustered minor nodes $(2 + 2)$, and (f) two clustered minor nodes $(1 + 3)$ are formed. If any of the major cell boundaries is not permeable, corresponding minor node is not connected to the adjacent minor node in the neighboring major cell. Equivalently, the minor nodes (on both major cells) corresponding to an impermeable major cell boundary are considered "blocked." A blocked minor node is illustrated in Fig. 7(g). More than one minor node may be blocked. Note that if all minor nodes corresponding to a major cell are blocked, it implies that the robot cannot enter this major cell and hence it is considered completely occupied as illustrated in Fig. 7(h). Here, even if the major cell is not occupied completely, as the robot of size $D \times D$ cannot enter the cell, it is equivalently blocked/completely occupied. Unlike in ref. [11], where a partially occupied major cell is considered blocked even when the robot can enter the cell and cover the free region in it, in the proposed work, we consider the major cell blocked only when all the four minor cells are blocked, and hence, a robot cannot enter the cell.

4. Details of the CPC Algorithm

The robot incrementally constructs a spanning tree on the adjacency graph corresponding to minor nodes. A minor node is set as "new" by default. Whenever a spanning tree edge is created to a "new" minor node, its state is set as "old." Further, when a minor node's state is changed to "old," all the minor nodes within the corresponding major cell, which form a cluster, are set as "old" minor nodes. This ensures that if a minor node has status "old" the status of the corresponding "clustered minor node" is set as "old." A "new" minor node which is connected to the current minor node in question, in the minor node adjacency graph, is called a "prospective" minor node. A scan in anticlockwise direction as illustrated in Fig. 8 is used to find these "prospective" minor nodes. As the spanning tree edges are constructed, a graph-level path is created along the corresponding subnodes, on the right side of the spanning tree edge. The robot moves along this path, avoiding obstacles, if any, using a

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

Algorithm 1: CPC algorithm.

Input: Starting cell S.
Recursive function: A function CPC(P,C), where "P" is the previous cell and "C" is the current cell.
Initialization:"P" is "null" and "C" is "S", the starting cell.
CPC(P,C):
Mark the current minor node (${}^{l}C$ ($l \in \{n, e, w, s\}$) and other minor nodes in the cluster as "old".
while (${}^{l}C$ has "prospective" minor node) **do**
1. Scan neighboring major cells in anticlockwise direction for the *prospective* minor node.
2. Construct the spanning tree edge from current minor node to the first available prospective minor node (say, ${}^{i}N_j$)
3. Mark ${}^{i}N_j$ and all minor nodes in the cluster as *old*.
4. Create path through corresponding sub nodes on the right side of spanning tree edge.
5. Move the robot along this path to cell N_j (avoiding obstacles if any using wall following algorithm).
6. Mark $C \rightarrow P$ and $N_j \rightarrow C$.
7. Execute CPC(P,C)

end while
if (End of the spanning tree edge is reached) **then**
1. Circumnavigate along the constructed spanning tree to reach cell along sub nodes, say "N".
2. Move the robot along this path to cell "N" (avoiding obstacles if any).
3. Mark $C \rightarrow P$ and $N \rightarrow C$
4. Execute CPC(P,C).

end if
if $P \neq S$ **then**
CPC(P,C)
end if
Return(End of CPC(P,C))

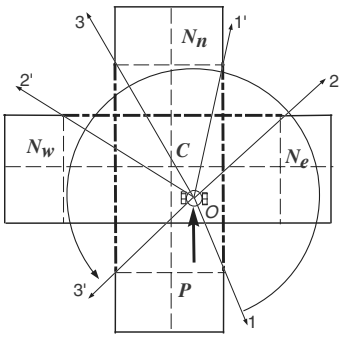


Fig. 8. The robot located in a subcell scans extended major cell boundaries to check permeability. A scan from $O1$ to $O3'$ scans all the three major cell boundaries for permeability. Scan between $O1$ and $O1'$ scans A_{CN_e}, $O2$ and $O2'$ scans A_{CN_n}, and $O3$ and $O3'$ scans A_{CN_w}.

wall following algorithm, covering subcells, including partially occupied subcells, to ensure exact coverage. Once the end of a branch of the spanning tree is reached, the robot comes back through the subcells on the other side of the spanning tree edges circumnavigating them, thus avoiding retracing the path. The pseudocode of the algorithm is shown in Algorithm 1.

The construction of a spanning tree edge is illustrated in Fig. 9. The robot enters the current major cell C from previous major cell P when a spanning tree edge is created between the minor node pair (${}^{n}P$, ${}^{s}C$) corresponding to the common cell boundary A_{CP}. Minor-node ${}^{s}C$ is marked *old*. Now the robot will check if the other three boundaries of C are permeable, in anticlockwise order, that is, cells N_e, N_n, and N_w, in order. The *prospective* minor nodes are ${}^{w}N_e$, ${}^{s}N_n$, and ${}^{e}N_w$, provided they are *new* and connected to ${}^{s}C$ through corresponding minor nodes in C. In this process, the algorithm identifies ${}^{s}C$ and ${}^{e}C$ form a clustered minor node and ${}^{e}C$ (and hence the corresponding clustered minor node)

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

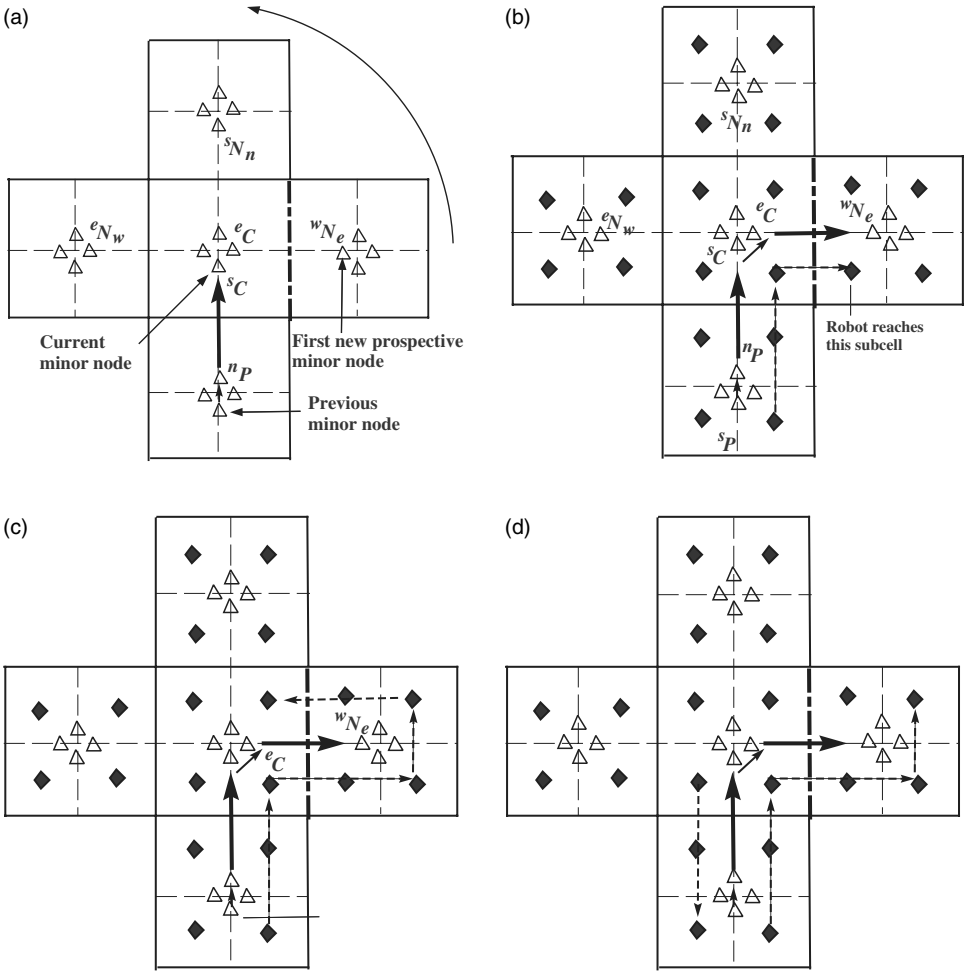


Fig. 9. Illustration of the proposed CPC algorithm. (a) Anticlockwise scanning of three neighbors (subnodes are not shown for clarity). (b) A move from "C" to a new cell N_e. (c) A return to "C" on reaching end of spanning tree branch, and (d) robot comes back to a previously visited cell on its return path.

is marked "old." If ${}^{w}N_e$ is a *prospective* minor node, a spanning tree edge ${}^{s}C \rightarrow {}^{e}C \rightarrow {}^{w}N_e$ is created and a robot path is created through corresponding subnodes as illustrated in Fig. 9(b). The robot now moves to the major cell N_e on this planned path avoiding obstacles if any. This procedure continues until from a current cell, no more *prospective* minor nodes are found, indicating the end of a spanning tree branch. Now a robot path is created along subnodes circumnavigating the spanning tree edge in counterclockwise direction to reach the current parent cell. In the example illustrated in Fig. 9(c), the robot returns from the major cell N_e to C, assuming no *prospective* minor nodes exist from N_e.[3] As the procedure continues, at some point of time, the robot returns to a previously not covered subcell of a previously covered major cell on the other side of the corresponding spanning tree edge. Figure 9(d) illustrates the graph-level return path to the major cell P through previously not covered subcells. The algorithm would terminate when the robot reaches a subcell it started from.

4.1. Illustrative example

Now we illustrate the proposed CPC algorithm with the help of an example. Consider a simple scenario as shown in Fig. 10. Here, the workspace contains a single obstacle at the center, shown as a black object at the center in Fig. 10(b). The spanning tree and graph-level path are constructed as discussed earlier. The graph-level coverage path is shown with dashed lines with arrows and spanning tree edges are shown with solid lines in Fig. 10(a). Here, the minor nodes are shown as small circles and circles encircling all minor nodes in a major cell depict a clustered minor node. Figure 10(b)

[3] Even if ${}^{e}C \leftrightarrow {}^{w}N$ is not a leaf edge, the robot eventually comes back to the major cell C and enters this subcell.

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

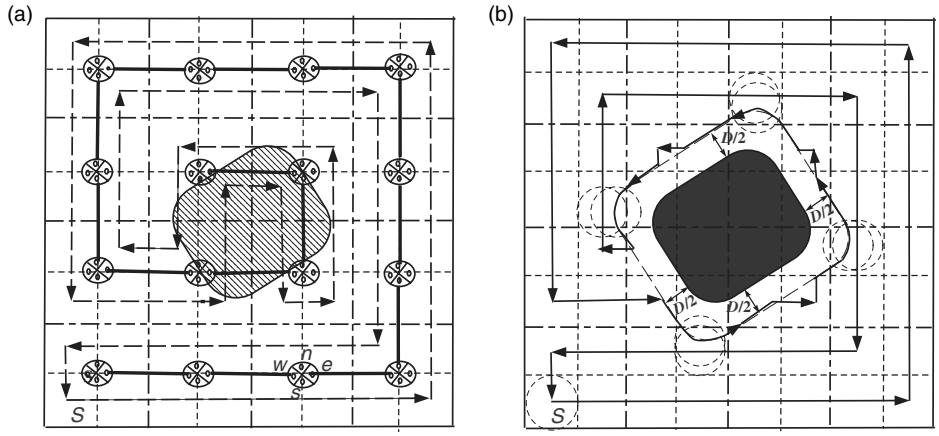


Fig. 10. Illustration of construction of (a) the spanning tree, the graph-level path, and (b) the actual robot path using the proposed CPC algorithm.

shows the robot coverage path, avoiding the obstacles, with solid arrowed lines. Except around the obstacle, the actual robot path is identical to the graph-level path through the subnodes. The robot path around the obstacle coincides with the obstacle boundary in the configuration space. Though there are no path retraces here, whenever two robot path segments are closer than D, coverage overlap occurs as seen at the corners of the obstacle shown with overlapping circles (dashed) of diameter D representing the size of the robot/coverage tool. In this specific example, the obstacle scenario is such that all the major cell edges are permeable and hence no minor node is blocked. Also, free space in all the major cells has a single connected patch, leading to a single clustered minor node (each with all four minor nodes) in every major cell.

5. Analysis of the Proposed Algorithm

In this section, we discuss the properties of the proposed CPC algorithm. First we make certain observations highlighting the similarities and differences between the proposed algorithm and the STC algorithm[11] in order to obtain the properties of the proposed algorithm.

P1 The adjacency graph over the free minor nodes is equivalent to the adjacency graph over the clustered minor nodes.

P2 Though the algorithm constructs an incremental graph edge between minor nodes, all the intra-cellular minor nodes connected to it are considered connected as once an incoming graph edge is created to a minor node, all the intra cellular minor nodes connected to it are considered as connected. Thus, it is equivalent to an incoming edge being created to a "clustered minor node" rather than just the minor node in question. Further, the algorithm uses the depth first search algorithm to construct a spanning tree incrementally. Thus, we may observe that the proposed algorithm incrementally constructs a spanning tree over the adjacency graph of clustered minor nodes, unlike in ref. [11] where it constructs spanning tree over the adjacency graph formed by the major nodes.

P3 The graph-level robot path through the subnodes is created on the right side of the spanning tree edges, in a similar manner as in ref. [11].

P4 The graph-level robot path through the subcells/nodes circumnavigates the spanning tree edge on reaching the end of branch/leaf as in ref. [11].

P5 If a major cell is free (all minor nodes connected), then:

(a) The graph-level robot path covers the subcells in counterclockwise order, not necessarily contiguously as stated in Proposition 3.2 in ref. [11], which is also applicable to the CPC algorithm in this situation. A possible order in which the subcells are covered along with the spanning tree edges is illustrated in Fig. 11. Here, thick solid lines with arrows indicate spanning tree edges and dashed arrowed lines show the graph-level path through the subnodes. Note that the number of spanning tree edges may differ. For example, in the scenarios shown in Fig. 11(a) and (b), one incoming and one outgoing spanning tree edges are created, in the

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

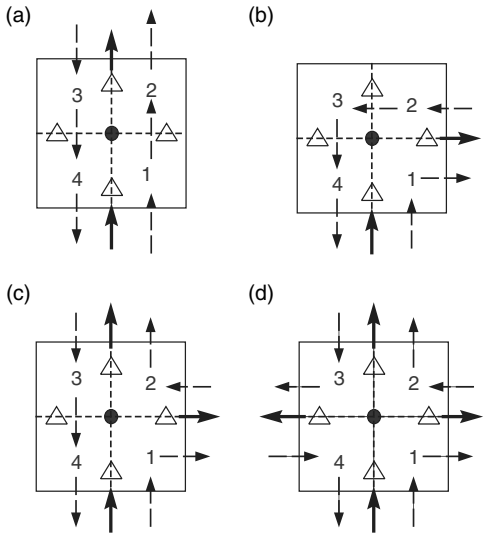


Fig. 11. Coverage of subcells when the major cell is free or effectively free.

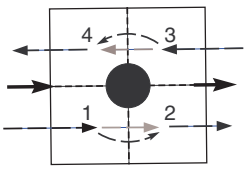


Fig. 12. Actual robot path deviates from the graph-level path when an obstacle is present.

scenario shown in Fig. 11(c) one incoming and two outgoing spanning tree edges are created, and in the scenario shown in Fig. 11(d) two incoming and two outgoing spanning tree edges are created. The order in which the subcells are covered is always counterclockwise $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ (cyclic order).

(b) All subcells (nodes) of free major cells are visited only once by the robot. This follows from Lemma 3.1 of ref. [11].

P6 Observation made in P5 is applicable even for effectively free but physically occupied major cells. Note that in the case of the STC algorithm[11] no spanning tree edge is created to such a partially occupied cell and hence no graph-level coverage path is created through the corresponding subnodes. However, in the proposed algorithm, such a cell is equivalent to a free major cell for the purpose of spanning tree edge creation and graph-level path generation through the subnodes. Actual robot path, avoiding obstacles, may result in coverage overlap as the region is no more coverage conducive as illustrated in Fig. 12. Thus, unlike the STC algorithm[11] with the proposed algorithm,

(a) all the subcells are covered.
(b) each subnode is visited exactly once at the graph-level.

P7 If a major cell is partially occupied and is not effectively free and has at least one of its boundary permeable, a spanning tree edge is created to the corresponding minor node, and hence, to the corresponding clustered minor node. Hence, each subnode is part of a graph-level path on the right side of the spanning tree edge or circumnavigating it.

(a) If a clustered minor node has only one minor node, then exactly one spanning tree edge is created to it.
(b) If a clustered minor node has two minor nodes, then exactly two (one incoming and one outgoing) spanning tree edges are created.
(c) If a clustered minor node has three minor nodes, then two (one incoming and one outgoing) or three spanning tree edges may be created.

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

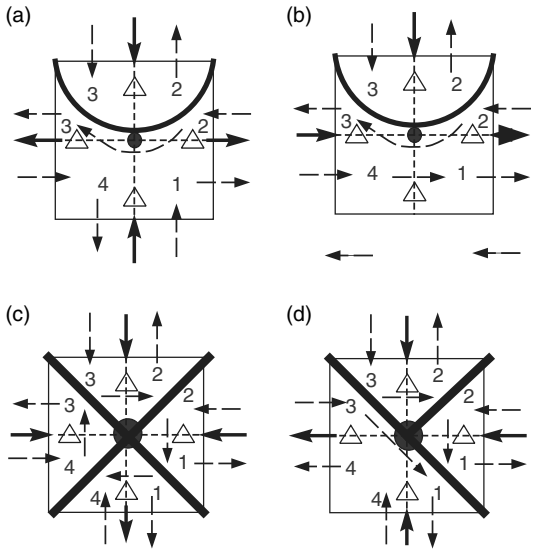


Fig. 13. Coverage of partially occupied major cell with all permeable edges.

(d) If a clustered minor node has four minor nodes, it is effectively free. See P5 and P6 for a detailed description.

In all the cases, the graph-level path will include all the four subnodes. Thus, as illustrated in Figs. 11 and 13:

(e) In the case of an effectively free major cell, all the subnodes are encountered exactly once.
(f) In the case of two clustered nodes, each subnode is encountered twice.
(g) In the case of three clustered nodes, each subnode is encountered thrice.
(h) In the case of four clustered nodes, each subnode is encountered four times.

Thus, at the graph-level, the number of repetitive coverage of a subnode is equal to the number of clustered nodes, which is at most four. However, note that the actual robot path which may use wall following technique results in a substantially lower coverage overlap compared to the graph-level coverage path. This is illustrated in Fig. 13. Here, the thick solid lines with arrows indicate spanning tree edges and the arrowhead shows the direction it was created. Dashed arrowed lines show the graph-level path through the subnodes with arrowhead indicating the direction of motion. Thus, the upper bound on coverage overlap is four times the number of non-effectively free cells. Observe that the non-effectively free cells are the ones on the obstacle or region boundary. These cells cannot be covered without overlap, as these situations make the region not coverage conducive.

P8 It is obvious that no spanning tree is created to a blocked major cell.

With these observations, we state the following results.

Theorem 1. *The CPC algorithm makes a robot cover a coverage conducive region completely without any overlap.*

In a coverage conducive region, we may note that the coverage path generated by the proposed algorithm is exactly the same as that provided by the STC algorithm,[11] as all the major cells are either completely free or completely occupied.

The following results are valid only when obstacle scenario is such that minor node doubling or tripling does not occur.

Lemma 1. *With the CPC algorithm, the graph-level path passes through every subnode corresponding to every free or partially free major cell, at least once.*

Lemma 1 guarantees a graph-level completeness.

Lemma 2. *With the CPC algorithm, every subnode corresponding to a free or effectively free major node appears exactly once in the graph-level coverage path.*

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

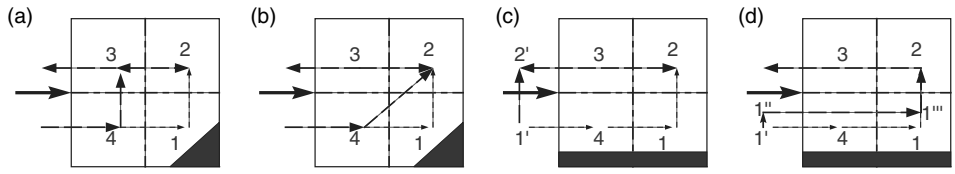


Fig. 14. Illustration of graph-level planned path versus deformed robot path in subcells partially occupied by obstacles. Scenario 1 with (a) full-STC and (b) CPC algorithms; Scenario 2 with (c) full-STC and (d) CPC algorithms.

Lemma 2 guarantees that there is no repetitive path, at least at the graph-level, through the subnodes corresponding to a free or effectively free major node. In fact, the robot path through all physically free major cells is non-overlapping.

Theorem 2. *With the CPC algorithm, a robot using suitable obstacle avoidance strategy, such as a wall following algorithm, covers all subcells corresponding to free or partially occupied major cells completely.*

Proofs of Theorem 1, Lemmas 1 and 2, and Theorem 2 follow from the observations P1–P8.

Now we make some observations comparing the full-STC algorithm[12] with the proposed algorithm.

1. The full-STC algorithm[12] covers only completely free subcells within a partially occupied major cell, while the proposed algorithm covers even partially free subcells. Thus, the proposed algorithm is complete in the exact sense, while STC algorithm[12] is complete at $2D \times 2D$ resolution and full-STC algorithm[12] is complete at $D \times D$ resolution.
2. In the full-STC algorithm,[12] the coverage path created through subnodes in a partially occupied major cell is deformed to pass through only those subnodes which correspond to completely free subcells. This leads to complete retrace of one or two coverage path segments, each of length D. Two such scenarios are illustrated in Fig. 14(a) and (c). In both the cases, the original path is into subnode 4, then, $4 \rightarrow 1 \rightarrow 2 \rightarrow 3$, and finally out of subnode 3. The deformed path in the scenario illustrated in Fig. 14(a) is $4 \rightarrow 3 \rightarrow 2 \rightarrow 3$, with a retrace of path segment $2 \leftrightarrow 3$ of length D. In the case of the scenario illustrated in Fig. 14(c), the deformed path is $1' \rightarrow 2' \rightarrow 3 \rightarrow 2 \rightarrow 3 \rightarrow 2'$, with a complete retrace of two D segments $2' \rightarrow 3 \rightarrow 2 \rightarrow 3 \rightarrow 2'$. Here, $1'$ and $2'$ are the subnodes in the preceding major cell. In both the scenarios, the full-STC algorithm treats partially occupied subcells as completely occupied.

   In contrast, in similar situations, the proposed CPC algorithm ensures complete coverage of these partially occupied subcells, without path retraces (which is avoidable in both these scenarios) as illustrated in Fig. 14(b) and (d). However, as the scenario warrants, at the coverage footprint level the overlap cannot be avoided completely, though the amount of overlap is substantially lower with the CPC algorithm as compared to that with the full-STC algorithm. In fact, in these scenarios, the overlap with the proposed CPC algorithm is equal to the area of the (partially occupied) major cell occupied by the obstacles.

Thus, the proposed CPC algorithm apart from providing complete coverage in an exact sense results in lower coverage overlap as compared to the full-STC algorithm. The competitiveness of the full-STC algorithm has been established in ref. [12]. Thus, we may observe that the proposed CPC algorithm too is competitive.

Note also that algorithms such as BSA[16] and BDC,[18] though provide exact coverage, do not guarantee non-overlapping coverage of even free cells/regions. Even when the region is coverage conducive, with these algorithms, the robot may have to move through already covered regions (or cells) in order to reach a pocket not covered (this is referred to as retracting). Further, as illustrated in Fig. 15(a), in the case of BDC algorithm[18] coverage overlap (shown as shaded areas) occurs when the width of a cell is not an exact multiple of D, due to the presence of an obstacle (black region).[4] Here,

[4] In fact, a similar situation has been used as motivation for use of Boustrophedon decomposition in the place of trapezoidal decomposition in ref. [18], as the Boustrophedon decomposition generates lesser cells compared to that with the Trapezoidal decomposition.

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

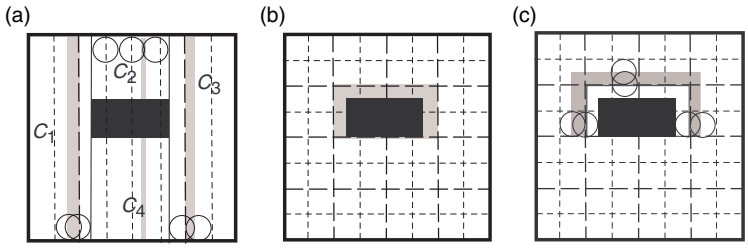


Fig. 15. Effect of an obstacle on (a) BDC, (b) STC, and (c) the proposed CPC algorithms.

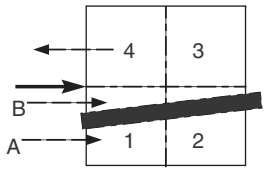


Fig. 16. A node doubling situation due to the presence of an obstacle (black object) leads to two possible paths marked "A" and "B."

short dashed vertical lines are D distance apart; C_1, C_2, C_3, and C_4 are decomposed cells. Observe that as illustrated in Fig. 15(b) in a similar scenario, with the STC algorithm, the region affected by the obstacle, shown as shaded region around it, is restricted to $2D \times 2D$ cells containing the obstacle, which is typically much smaller than the affected (in terms of coverage overlap) area with the BDC algorithm. The shaded region in Fig. 15(b) is not covered by the STC and full-STC algorithms (if there are completely free subcells within the shaded/affected region, then they are covered by the full-STC algorithm, however with overlap). As shown in Fig. 15(c), with the proposed CPC algorithm, this region is completely covered, with overlap (shown as shaded region), which cannot be avoided anyway. The proposed CPC algorithm combines the advantages of both the STC (or similar) algorithm[11] in terms of non-overlapping coverage, for coverage conducive region, and the BDC (or similar) algorithm[18] in terms of exact coverage.

As observed in the above discussion, if a subcell is partially occupied, then the full-STC algorithm results in path retrace, while unless the subcell is completely occupied, the proposed CPC algorithm results in non-retracing robot path. In fact, in both the scenarios illustrated in Figs. 14 and 15, the proposed CPC algorithm results in an optimal path, in the sense that it minimizes coverage overlap, while maximizing the coverage. Note that the coverage with the CPC algorithm is 100% of the free region within each major cell and the percentage overlap is equal to the percentage of the major cell occupied by the obstacle. In contrast, the full-STC algorithm does not provide 100% coverage of the free region within a partially occupied major cell, and at the same time the coverage overlap is higher than that with the proposed CPC algorithm. Though we do not provide an empirical or analytical measure, such as bound on ratio of coverage overlaps with the full-STC and CPC algorithms here, we present a comparative quantitative analysis using an illustrative example in a later section.

Implication of node multiplication on coverage: As mentioned earlier, we assume that the obstacle scenario does not lead to minor node multiplication (doubling or tripling). All the results we discussed in this section are applicable only when this assumption is valid. Theorem 1 is still applicable, as in the case of a coverage conducive region all the major cells are completely free, and hence, node multiplication situations do not arise. Though we do not address the issue of minor node multiplication and suitable modification to the proposed algorithm to successfully handle these situations, for the sake of completion, we provide a brief discussion on a simple modification to the proposed algorithm to handle these situations and the expected results through a few illustrative examples. When node multiplication is encountered, the algorithm may run into difficulty while generating the robot coverage path as there are two (or three) possible paths. Consider a situation as illustrated in Fig. 16. Here, a spanning tree edge created from the left side neighbor results in two possible paths on its right side, marked "A" and "B.". The algorithm cannot decide which path the robot should use. In this case, we can observe that if the robot takes the path marked "A,", then all the subcells

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

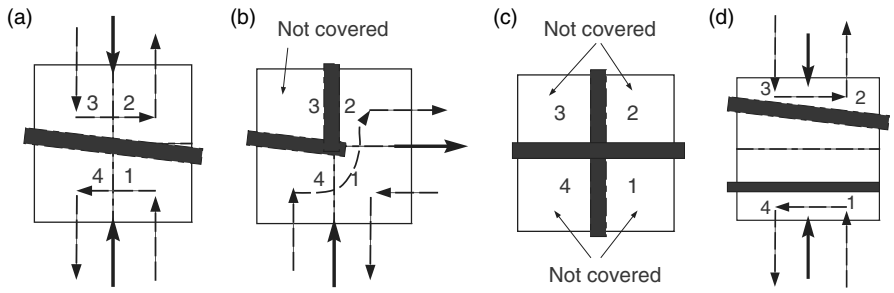


Fig. 17. Coverage with modified CPC algorithm in scenarios leading to minor node doubling or tripling.

are covered. The bottom portions of the subcells "1" and "2" will be left not covered with the path marked "B."

The algorithm may be modified by incorporating an additional condition while scanning for the prospective next minor node, so as to handle scenarios leading to node doubling/tripling. The algorithm skips a connected minor node if it finds that the corresponding major cell boundary is made up of more than one connected path, that is, the robot identifies a situation leading to node doubling/tripling. With such a modification, the algorithm still has all the properties of completeness and competitiveness whenever no situation leading to minor node multiplication is present. When such a situation occurs, the robot may not be able to provide complete coverage of only such major cells. A few illustrative scenarios are shown in Fig. 17. In the scenario shown in Fig. 17(a), no spanning tree edges are created to the left and right boundaries as the corresponding minor nodes are doubled. However, the algorithm still achieves complete coverage as top and bottom cell boundaries are free. A scenario is shown in Fig. 17(b), where a portion of subcell 3 is not covered, while the rest of the region within this cell is covered. In another scenario as illustrated in Fig. 17(c), none of the subcells are covered as all major cell boundaries have doubled minor nodes. Finally, Fig. 17(d) shows a situation where minor node tripling occurs leading to incomplete coverage of the corresponding major cell. These scenarios are presented only to illustrate how node doubling/tripling can be handled by suitable modification to the CPC algorithm. As mentioned earlier, this paper assumes that scenarios leading to node doubling/tripling do not arise.

The STC and full-STC algorithms handle the scenarios illustrated in Fig. 17 differently. In the case of the STC algorithm, no spanning tree is created to any of these major cells, and hence none of the subcells are covered. In the case of the full-STC algorithm, if any of the subcell is completely free, it is covered irrespective of node doubling/tripling.

6. Implementation Issues

In this section, we present a brief discussion on some issues that affect the implementation of the proposed CPC algorithm on a physical robot.

6.1. Obstacle detection

The proposed CPC algorithm requires to detect the obstacles in order to construct the spanning tree over the minor nodes and to implement a wall following algorithm. Note that in both the scenarios, proximity sensors can be used which are capable of detecting obstacles with reasonable accuracy. Further, the algorithm is not very sensitive to reasonable inaccuracies in obstacle detection in terms of its relative location from the robot's current position. Hence, algorithm's performance in terms of completeness and percentage overlap is expected to degrade only marginally in the presence of nominal obstacle detection sensor inaccuracies.

Consider a scenario as illustrated in Fig. 18(a). Here, the cells bounded by long dashed lines are major cells and the dark discs at the center of these cells show the corresponding major nodes, while the subcells are shown with short dashed lines with smaller discs representing the corresponding subnodes. The robot enters the major cell marked "C" from the previous cell marked "P.'. Due to obstacle localization error because of the sensory errors, the obstacle present in the major cell marked "F" is identified as present in cell "E." In this situation, with the STC algorithm, the robot leaves the entire major cell "E" not covered even when it is completely free. Thus, a small error in obstacle detection, say $d \ll D$, can cause the STC algorithm to leave a $2D \times 2D$ region not covered. However,

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

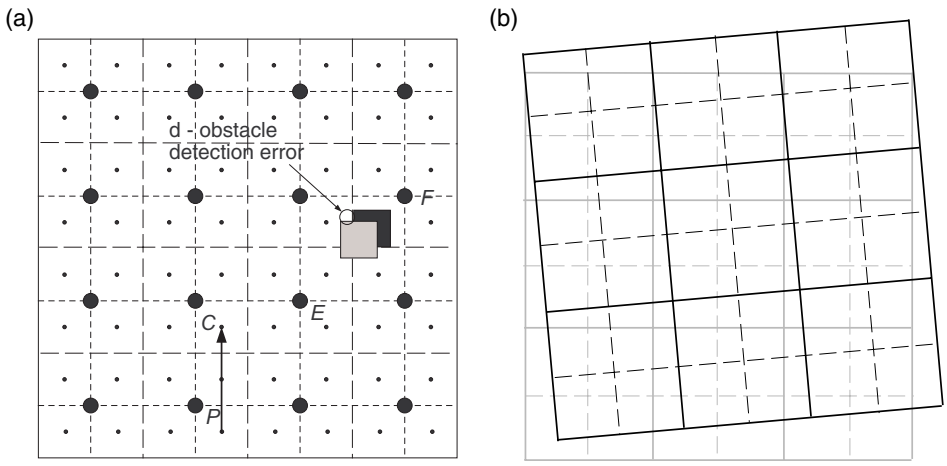


Fig. 18. Illustration of (a) the effect of obstacle detection sensory error on coverage performance and (b) shift in grids due to robot drifting from planned path and incorrect localization.

with the proposed CPC algorithm, the major cell "E" is covered except for a small region wrongly "assumed" to be occupied by the obstacle. Thus, a small obstacle detection error results in only a small region left not covered.

6.2. Localization

Localization is one of the fundamental requirements in any path planning algorithm or even execution of a planned path on a mobile robot. In the case of approximate cellular decomposition-based coverage algorithms, the grid structure forms the basis on which the path is planned. As the robot veers off the gridded path it has to move, the grid it constructs to plan its path from the current major cell may not coincide with the initial grid. This is illustrated in Fig. 18(b). Algorithms such as STC may fail when the current major cell which the robot initially assumed to be free now has an obstacle, in its return path on the other side of the spanning tree edge. However, such a problem is not encountered with the proposed CPC algorithm as it uses obstacle avoidance by wall following. In any case, with nominal localization error, the performance of the algorithm in terms of deviation from the ideal path may be assumed to degrade gracefully. However, when the localization error is comparable to the size of the robot, which forms the basis for the size of gridding, the algorithm may not perform satisfactorily. The effect of localization error is less prominent when off-line CPP algorithms are used, including the off-line implementation of the proposed CPC algorithm.

Note that the implication of obstacle detection sensor inaccuracies and localization errors are common to most CPP algorithms and more prominent in the case of approximate cellular decomposition-based CPP algorithms such as STC, full-STC, and BSA. In spite of this, approximate cellular decomposition-based strategies are extensively used owing to their suitability for planning. A few simulation and experimental results are reported in the literature demonstrating that the coverage performance degrades only nominally in the presence of nominal sensory noise.[24] A detailed formal analysis on the effect of sensory noise including localization errors on the coverage path along with possible techniques that reduce such an effect will be very useful. However, such an analysis is beyond the scope of this paper.

7. Results and Discussion

In this section, we first compare the coverage paths generated by the proposed CPC algorithm with that generated by the STC and full-STC algorithms. Then, we provide a sample result from implementation of the proposed algorithm in the robot operating system (ROS)/Gazebo environment. Finally, we provide results of a demonstrative experiment with the Fire Bird V mobile robot.

7.1. Coverage path comparison

First we present results comparing the coverage performance using the proposed CPC algorithm with that using the STC and full-STC algorithms in terms of percentage of coverage and coverage

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

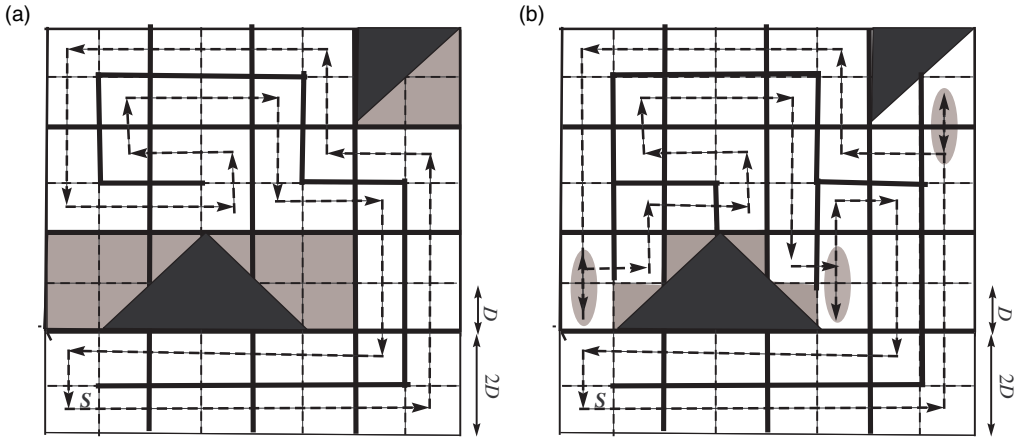


Fig. 19. Comparison of coverage path generated using (a) STC (b) full-STC.

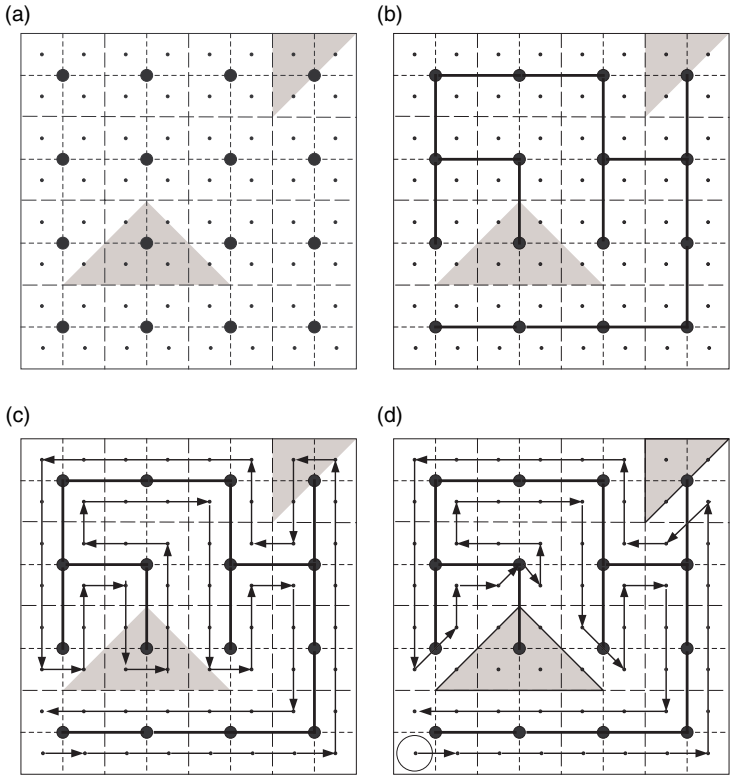


Fig. 20. Process of coverage path generation in a given scene.

overlap. Figures 19(a), (b), and 20(d) show the robot path created using the STC, full-STC, and the proposed CPC algorithms, respectively. The area considered has 9.375% of obstacles (shown as black triangles). In Fig. 19(a) and (b), black triangles are the obstacles in the region, shaded regions are those that are not covered, and double arrowed lines (circled and shaded in Fig. 19(b)) show the retraced path. It can be observed from Fig. 19(a) that the STC algorithm completely skips the coverage of the entire partially occupied major cells (shaded region), that is, an area of 25% of the total area. The STC algorithm covers 82.75% of the free area. As shown in Fig. 19(b), the full-STC algorithm skips all the partially occupied subcells (shaded region) which account for 14% of the total area, and it covers about 96% of the free space, retracing (double arrowed lines) 5% of the total path length. Observe also that each retrace results in complete overlap, that is, there is 5% coverage area overlap. Figure 20(a)–(d) shows the process of coverage path generation using the proposed CPC algorithm for the same environment. Figure 20(a) shows the area to be covered, which is divided

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

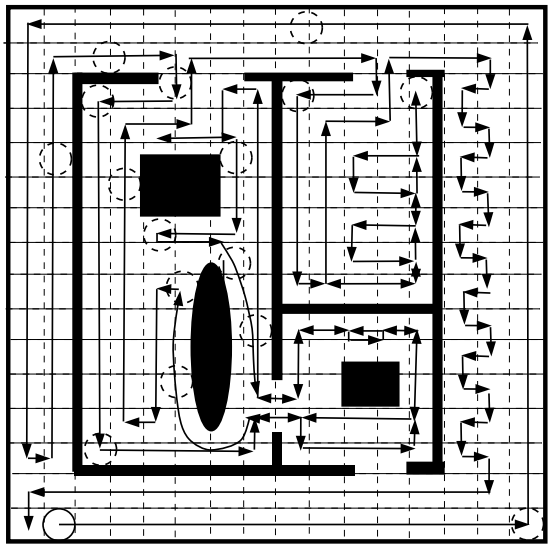


Fig. 21. Coverage path generated by the proposed CPC algorithm in an office-like environment.

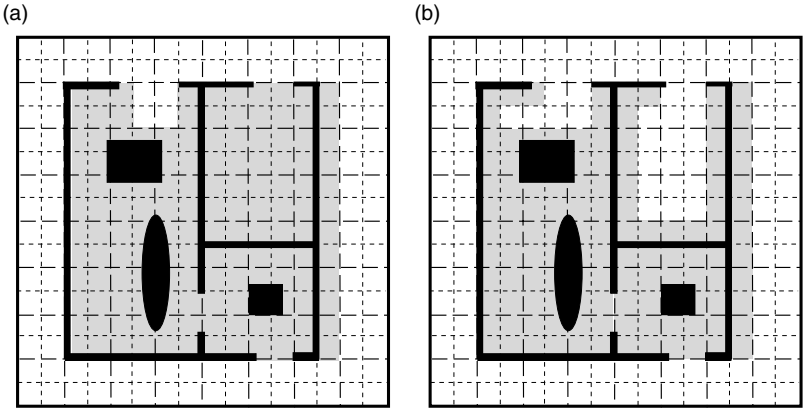


Fig. 22. Coverage of an office-like environment using (a) the STC and (b) the full-STC algorithms. Shaded regions show the space that is not covered by the robot.

into major (long dashed lines) and subcells (dashed lines). Corresponding major nodes (discs) and subnodes (dots) are also shown. Figure 20(b) shows how the spanning tree is created through the clustered minor nodes (which coincide with the major nodes here). Graph-level planned path (lines with arrow) based on the ST is shown in Fig. 20(c) and the actual robot path (lines with arrow) avoiding obstacles is shown in Fig. 20(d). It can be observed from Fig. 20(d) that the robot covers the whole of the free space achieving exact coverage. Further, there are no retraced paths. However, coverage overlap occurs at some areas around the boundaries of obstacles, which cannot be avoided completely by any algorithm if exact coverage is required.

Now we consider an office-like scenario and compare the performance of the proposed CPC algorithm with those of the STC and full-STC algorithms. Figures 21, 22(a), and (b) show the coverage by the proposed CPC, STC, and full-STC algorithms, respectively. Black regions represent obstacles, grey regions represent the free region that are not covered, and white regions show the free region that are covered by the robot. Figure 21 shows the complete robot path (shown as solid arrowed lines) with the proposed CPC algorithm. Observe that coverage overlap occurs wherever the distance between two parallel paths is less than D (shown at several places with a circle of diameter D). A complete retrace of path is shown with double arrowed lines. It may be observed that coverage is complete in exact sense with the proposed CPC algorithm, while most of the interior region is not covered by the robot using the STC algorithm as shown in Fig. 22(a). The area covered with the

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

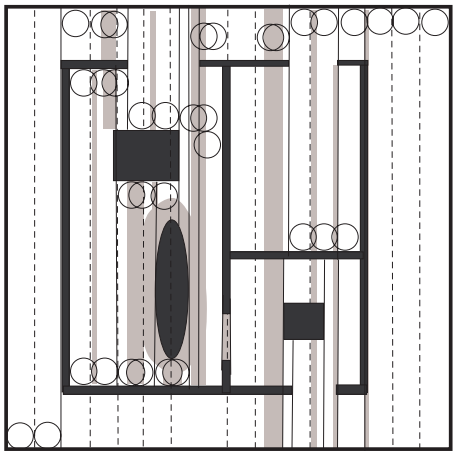


Fig. 23. Boustrophedon decomposition of the office-like environment and the coverage overlap (shown as shaded strips) due to "non-coverage conducive" cells. Small circles show the robot footprint of diameter D.

full-STC algorithm, as shown in Fig. 22(b), though is better than that with the STC algorithm, the robot still does not cover most interior regions.

Figure 23 shows the coverage overlap using the BDC algorithm for the same environment as shown in Fig. 22. Observe that due to presence of several cells with width less than D (or not exact multiple of D), using the BDC algorithm results in a higher coverage overlap, though by its property, exact coverage is achieved. The overlaps shown here do not include those due to retracting and restarting while traversing over the Reeb graph formed by the decomposed cells.

7.2. Simulation using TurtleBot in ROS environment

Now we present results of realistic simulation of the proposed CPC algorithm on a TurtleBot[5] mobile robot in the ROS/Gazebo environment. The results demonstrate the proposed algorithm and also compare its performance with that of the STC algorithm in terms of actual robot path. Figure 24(a) shows the environment modeled within the Gazebo simulator. This environment is mapped to visualize the path generated by the algorithm in RVIZ in ROS which is shown in Fig. 24(b). Figure 24(c) and (d) shows the coverage path generated using the STC and CPC algorithms on a TurtleBot robot. All the partially occupied cells that are not covered by STC algorithm (Fig. 24(c)) are covered by the proposed CPC algorithm (Fig. 24(d)), ensuring exact coverage.

We have used Hokuyo URG-04LX-UG01 laser-based sensor for obstacle detection, which has an accuracy of ± 30 mm. The robot footprint size $D = 360$ mm. As we have discussed in Section 6.1, in spite of this error, it is evident from Fig. 24(d), that the robot covers the entire obstacle-free region. Note that, because of these sensory errors, there are coverage gaps wherever the distance between the adjacent robot paths is more than D and coverage overlap whenever it is less than D. This is exactly the performance degradation in the presence of nominal sensorial errors that we discussed in Section 6.

7.3. Experiments using Fire Bird V robots

Finally, we present results of experiments conducted using Fire Bird V robots. Fire Bird V is Atmega2560-based robotic research platform designed by ERTS Lab, CSE, IIT Bombay and manufactured by Nex Robotics Pvt Ltd. The differential wheeled robot is equipped with three white line sensors, five Sharp GP2D12 IR range sensors, eight analog IR proximity sensors, eight analog directional light intensity sensors, and two position encoders. The robot has wireless ZigBee communication, USB communication, Wired RS232 communication, and simplex infrared communication capabilities. The robot has the following dimensions: diameter: 16 cm, height: 10 cm, and weight: 1300 g.

[5] TurtleBot is a low-cost, personal robot kit with open-source software. It was created at Willow Garage by Melonee Wise and Tully Foote in November 2010.

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

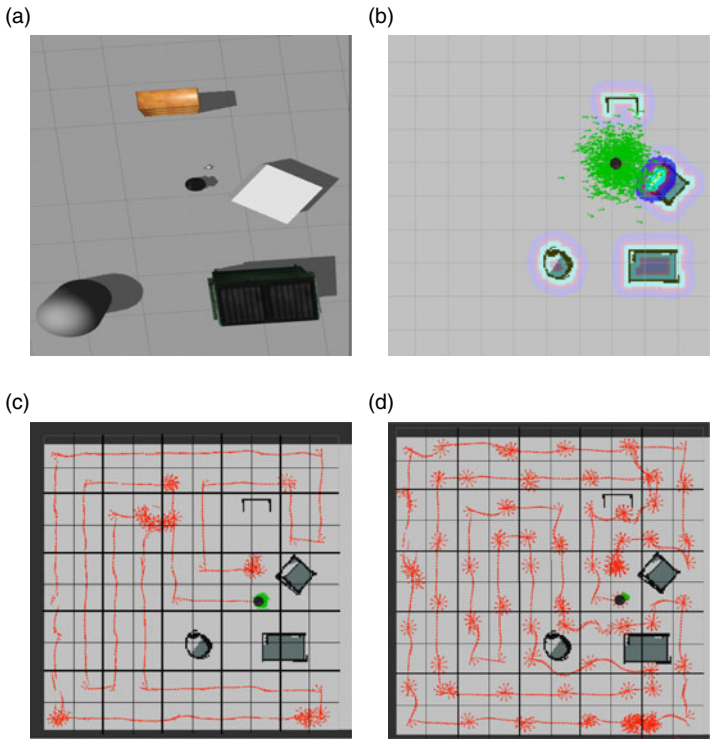


Fig. 24. (a) TurtleBot in an area with static obstacles modeled in ROS and (b) the map of the region in RVIZ. TurtleBot coverage path in the ROS/Gazebo environment (c) with the STC algorithm and (d) with the proposed CPC algorithm.

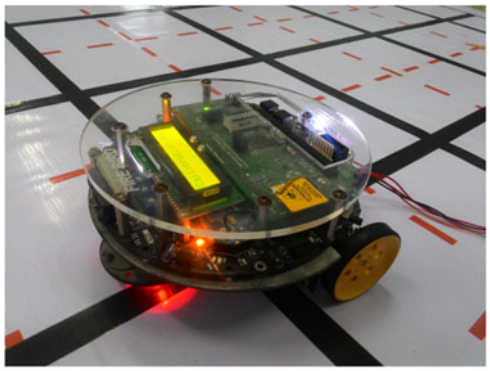


Fig. 25. Photograph of a Fire Bird V robot in the gridded environment. Dashed (red) lines show the $D \times D$ subcells, dark black gridded lines pass through the subnodes along which the robot needs to move.

As discussed in the previous section, localization plays an important role in a path planning algorithm such as the proposed CPC algorithm. In lab environments, localization may be achieved by the use of odometric sensors/dead reckoning, use of overhead cameras, or motion capture systems. Each of these methods has its own disadvantages. While the motion capture systems are typically expensive the odometry and overhead camera systems are economical options. Dead reckoning method is prone to errors and accuracy also depends on the type of surface on which the robot moves. With both an overhead camera and a motion capture system, an external computer is required and the localization problem is solved outside the robot itself. In this work, as the algorithm is based on $2D \times 2D$ grids and the robots move through the subnodes (which form a $D \times D$ gridded environment), except while avoiding obstacles using wall following algorithm, we printed $D \times D$ grids with solid black lines. The robot uses its line following sensors to follow the grid. These guide lines serve as directional guides replacing the orientational localization, and the grid points where two perpendicular guide lines meet serve as relative positional guides replacing the absolute positional localization. Thus, by using only printed grid lines as guides, the robot can execute a planned path, that is, move

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

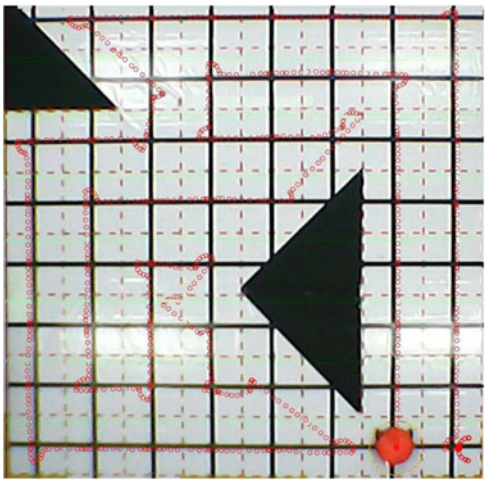


Fig. 26. Coverage by a Fire Bird V robot using the proposed CPC algorithm.

along the planned path in the gridded space. The robot uses onboard wheel encoders when it moves off the grid around the obstacles.

Figure 25 shows photograph of a Fire Bird robot in a printed gridded environment. We provided the planned path as shown in Fig. 20(d) to the robot. The robot moved along the planned path using a line following algorithm along the grids and using wheel encoders while moving off the grid (diagonal path) around the obstacles, as shown in Fig. 26. It can be observed that the generated path provides exact coverage and the robot successfully moves along the coverage path generated.

The experimental result serves only to demonstrate the proposed CPC algorithm. Several practical considerations such as localization, detection of cell permeability using onboard sensors, and implementation of suitable wall following algorithm to translate the planned graph level path to actual robot path need to be addressed before the proposed CPC algorithm is implemented and tested on a mobile robot.

8. Conclusions

We proposed a new approximate cellular decomposition-based online CPP algorithm for a mobile robot, which provides exact coverage with reduced coverage overlap compared to that with the existing coverage algorithms. A new concept of minor node is introduced and a spanning tree is constructed incrementally over the adjacency graph induced by them. We established that the proposed CPC algorithm provides competitive and complete coverage when situations leading to node doubling/tripling do not occur. The performance of the proposed algorithm is compared with that of the STC, full-STC, and BDC algorithms, in terms of completeness of coverage and coverage overlap. The algorithm was implemented on a TurtleBot mobile robot within the ROS/gazebo environment and on a Fire Bird V robot, in off-line mode, to demonstrate its performance in a realistic scenario.

We have identified coverage performance issues when the situations of minor node doubling/tripling occur, though we do not address these issues in detail. Suitable refinement to the algorithm to provide an exact coverage, if possible, even when situations of minor node doubling/tripling arise; suitable modification to the algorithm to generate a back and forth motion instead of a spiralling motion; a true implementation of the proposed CPC algorithm on a mobile robot after taking into account all practically relevant issues are some of the directions for future work. A formal and detailed quantitative analysis of how the coverage performance varies with the sensory errors is also a very useful direction for the future work.

Acknowledgments

The authors acknowledge Mr. Arjun Sadananda and Mr. Nagarakshith M. S. for their help in experiments with the Fire Bird V Robots.

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press

References

1. E. U. Acar, H. Choset, Y. Zhang and M. Schervish, "Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods," *Int. J. Robot. Res.* **22**(7–8), 441–466 (2003).
2. P. Dasgupta, A. Muñoz-Meléndez and K. R. Guruprasad, "Multi-robot Terrain Coverage and Task Allocation for Autonomous Detection of Landmines," *Proceedings of SPIE 8359, Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense XI, 83590H* (2012).
3. W. Sheng, N. Xi, M. Song, Y. Chen and P. MacNeille, "Automated CAD-Guided Robot Path Planning for Spray Painting of Compound Surfaces," *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, vol. 3 (2000) pp. 1918–1923.
4. K. L. Doty and R. R. Harrison, "Sweep Strategies for a Sensory-Driven, Behavior-Based Vacuum Cleaning Agent," *Proceedings of AAAI 1993 Fall Symposium Series* (1993) pp. 1–6.
5. M. Waanders, "Coverage Path Planning for Mobile Cleaning Robots," *Proceedings of 15th Twente Student Conference on IT, The Netherlands. Copyright* (2011).
6. M. Weiss-Cohen, I. Sirotin and E. Rave, "Lawn Mowing System for Known Areas," *Proceedings of Computational Intelligence for Modelling Control & Automation, 2008 International Conference on* (IEEE, 2008) pp. 539–544.
7. E. M. Arkin, S. P. Fekete and J. S. B. Mitchell, "Approximation algorithms for lawn mowing and milling," *Comput. Geom.* **17**(1), 25–50 (2000).
8. C. N. Macleod, G. Dobie, S. G. Pierce, R. Summan and M. Morozov, "Machining-based coverage path planning for automated structural inspection," *IEEE Trans. Robot.* **15**(1), 202–213 (2018).
9. H. Choset, "Coverage for robotics–a survey of recent results," *Ann. Math. Artif. Intell.* **31**(1–4), 113–126 (2001).
10. E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auto. Syst.* **61**(12), 1258–1276 (2013).
11. Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Ann. Math. Artif. Intell.* **31**(1–4), 77–98 (2001).
12. Y. Gabriely and E. Rimon, "Competitive on-line coverage of grid environments by a mobile robot," *Comput. Geom.* **24**(3), 197–224 (2003).
13. N. Agmon, N. Hazon and G. A. Kaminka, "Constructing Spanning Trees for Efficient Multi-robot Coverage," *Proceedings of IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*, May 2006, pp. 1698–1703.
14. X. Zheng, S. Jain, S. Koenig and D. Kempe, "Multi-robot Forest Coverage," *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, August 2005, pp. 3852–3857.
15. E. Gonzalez, M. Alarcon, P. Aristizabal and C. Parra, "BSA: A Coverage Algorithm," *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2 (IEEE, 2003) pp. 1679–1684.
16. E. Gonzalez, O. Alvarez, Y. Diaz, C. Parra and C. Bustacara, "BSA: A Complete Coverage Algorithm," *Proceedings of IEEE International Conference on Robotics and Automation* (IEEE, 2005) pp. 2040–2044.
17. V. R. Jisha and D. Ghose, "Frontier based goal seeking for robots in unknown environments," *J. Intell. Robot. Syst.* **67**, 229–254 (2012).
18. H. Choset, "Coverage of known spaces: The boustrophedon cellular decomposition," *Auto. Robots* **9**(3), 247–253 (2000).
19. S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **34**(1), 718–725 (2004).
20. A. Xu, C. Viriyasuthee and I. Rekleitis, "Efficient complete coverage of a known arbitrary environment with applications to aerial operations," *Auto. Robots* **36**, 365–381 (2014).
21. R. Mannadiar and I. Rekleitis, "Optimal Coverage of a Known Arbitrary Environment," *Proceedings of IEEE International Conference on Robotics and Automation* (IEEE, 2010) pp. 5525–5530.
22. G. P. Strimel and M. M. Veloso, "Coverage Planning with Finite Resources," *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2014, pp. 2950–2956.
23. I. Shnaps and E. Rimon, "Online coverage of planar environments by a battery powered autonomous mobile robot," *IEEE Trans. Autom. Sci. Eng.* **13**(2), 425–436 (2016).
24. J. Song and S. Gupta, "$\varepsilon^{\star}$: An online coverage path planning algorithm," *IEEE Trans. Robot.* **34**(2), 526–533 (2018).
25. K. R. Guruprasad and T. D. Ranjitha, "ST-CTC: A Spanning Tree-Based Competitive and Truly Complete Coverage Algorithm for Mobile Robots," *Proceedings of Advances in Robotics, 2nd International Conference of Robotics Society of India*, July 2015.

https://doi.org/10.1017/S026357472000096X Published online by Cambridge University Press