Peer-reviewed version available at *Robotics* **2019**, *8*, 44; doi:10.3390/robotics8020044

Article

Proposed Smooth-STC Algorithm for Enhanced Coverage Path Planing Performance in Mobile Robot Applications

Hai Van Pham [1,*], Philip Moore [2] and Dinh Xuan Truong [3]

[1, 3] School of Information Technology and Communication, Hanoi University of Science and Technology, 1 Dai Co Viet, Le Dai Hanh, Hai Ba Trung, Hanoi, Vietnam; haipv@soict.hust.edu.vn, truongdinh4w@gmail.com

[2] School of Information Science and Engineering, Lanzhou University, Feiyun Building, 222 Tianshui S Rd, Chengguan Qu, Lanzhou Shi, Lanzhou 730030, Gansu Sheng, China; ptmbcu@gmail.com

* Correspondence: haipv@soict.hust.edu.vn; Tel.: (84-4) 823 727555

Abstract: Robotic path planning is a field of research which is gaining traction given the broad domains of interest to which path planning is an important systemic requirement; the aim being to optimise the efficacy of robotic movements in a defined operational environment. For example, robots have been employed in many domains including: cleaning robots (such as vacuum cleaners), automated paint spraying robots, window cleaning robots, forest monitoring robots, agricultural robots (often driven using satellite and geostationary positional satellite data). Additionally, mobile robotic systems have been utilised in disaster areas and locations hazardous to humans (such as war zones in mine clearance). The *coverage path planning* problem describes an approach which is designed to determine the path that traverses all points in a defined operational environment while avoiding static and dynamic (moving) obstacles. In this paper we present our proposed Smooth-STC model, the aim of the model being to identify an optimal path, avoid all obstacles, prevent (or at least minimise) backtracking, and maximise the coverage in any defined operational environment. The experimental results in a simulation show that, in uncertain environments, our proposed smooth STC method achieves an almost absolute coverage rate and demonstrates improvement when measured against alternative conventional algorithms.

Keywords: Mobile robotic systems: Coverage Path planning; Smooth-STC algorithm; Robotic C-Space path planning

1. Introduction

Robots have been employed in a diverse range of domains and systems; for example robotic systems have addressed the demands of: cleaning robots (such as vacuum cleaners), automated paint spraying robots, window cleaning robots, forest monitoring robots, agricultural robots (often driven using geostationary positional satellite data), and in disaster areas and locations hazardous to humans (such as war zones). The broad range of robot applications has driven interest in optimising the efficacy of robots in operation, to address this challenge *Coverage Path Planning* (CPP) problem has gained traction, CPP describes an approach which is designed to determine the path that traverses all points in a *defined operational environment* (DOE) while avoiding static and dynamic (moving) obstacles within a defined operational area [8].

An example of CPP is a automatic robotic domestic vacuum cleaner which, when activated using CPP, can manoeuvre around a DOE (typically a room or on one level) to clean the area while avoiding static (moving) obstacles (such as furniture) or dynamic (moving) objects (such as a domestic animal) when traversing the DOE [20]. To solve this problem, it is necessary to build an algorithm to find optimal coverage path for the DOE; in studies addressing CPP (for example see Cao et al. [3]) basic standards for the robotic path coverage problem have been set (see Section 2).

Peer-reviewed version available at Robotics 2019, 8, 44; doi:10.3390/robotics8020044

In this paper, to overcome the outstanding limitations in current coverage path planning applications [and obtain maximum coverage of the DOE] we propose a new Smooth-STC (SmSTC) algorithm based on the use of a spanning tree. Our proposed SmSTC model targets the identification of optimal paths while avoiding all obstacles, preventing (or at least minimise) backtracking, and maximising the coverage in any defined operational environment. The experimental results in a simulation show that, in uncertain environments, our proposed smooth STC method achieves an almost absolute coverage rate and demonstrates improvement when measured against alternative conventional algorithms.

The remainder of this paper is structured as follows: Section 2 considers related research addressing the CPP problem. CPP in unknown environments is addressed in Section 3 with CPP environments considered in Section 4. Section 5 sets out the proposed SmSTC algorithm with the results and a discussion set out in Section 6 with an overview of potential future work. The paper closes with concluding observations.

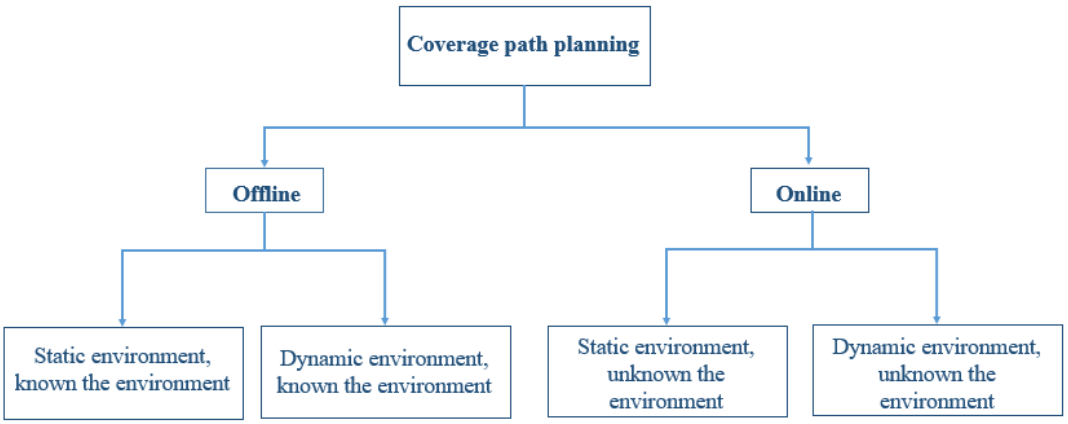


Figure 1. Composite classification showing the main approaches to the CPP problem. (source: [4] [8])

2. Related Research

In considering the CPP problem there are a number of basic standards for the robotic path coverage problem have been identified, for example see Cao *et al* [3]); in practice it is not always possible to satisfy all the basic standards and there is frequently a 'trade-off' to reach an optimal domain-specific solution. The 'trade-off' is reflected in the current research where methods focus on certain 'specific' constraints.

Over the past three decades, robotics research has mainly focused on solving the CPP and coverage path optimisation problems in both static environments (environments where obstacles do not move) and dynamic environments (environments in which there are both static and dynamic (moving) obstacles). The CPP problem may be approached in two ways: *classical* and *heuristic* methods (also identified as artificial intelligence (AI)); each of the approaches demonstrates specific strengths and weaknesses [8,17]. In Figure 1 we present our graphical composite summary of the mainstream approaches to CPP, namely *off-line* and *on-line* methods; the classification tree is drawn from related research sources reviewed, principally [4] [8].

CPP algorithms are categorised as *on-line* and *off-line*, see Choset, 2001) [4]. Off-line algorithms assume that the working environment (the DOE) is known, therefore the path of the robot can be optimised. In contrast, on-line algorithms do not need to know the advance the DOE information, the operation of a robot is based on information collected from 'real-time' sensors which read and monitor the DOE and the location, shape, and size of the obstacles [6,21] thus providing a suitable path

Peer-reviewed version available at Robotics 2019, 8, 44; doi:10.3390/robotics8020044

to provide complete coverage of the DOE. On-line algorithms are generally known as sensor-based coverage algorithms [1,8].

Moreover, dependent on the conditions of the DOE, it is possible to classify more detailed problems based on static environments or dynamic environments; in this paper, the authors consider static environments. Figure 1 1 shows a graphical representation of the coverage path planing problem.

A simple approach to solving the CPP problem is a random algorithm. For example, consider a cleaning robot where the cleaning is randomly executed without time restrictions, while the cleaning may be effective the algorithmic approach is not efficient or reliable [3,7,8]. For example, some commercial robots (such as Roomba by iRobot [20]) do not need to be equipped with complex sensors to enable location determination and calculation (of the coverage path) and are therefore more simple. Nonetheless, in a large area or a three-dimensional space [such as underwater or aerial DOE's] the operational costs of the robot (in energy and time) are very expensive, not feasible in reality, so need other approaches [7,18]. First, it is necessary to divide the environment into small continuous areas where the robot can completely move without colliding with obstacles. In this way, there are algorithms such as *Trapezoidal* Decomposition, *Boustrophedon* Decomposition, and BSA, BA*, just to name a few [9,11]. All of them need to have a mechanism for linking domains to achieve the best coverage. It marks "backtracking" points and uses different algorithms to find the shortest path from the current location to those backtracking points. The greater the number of backtracking points, the larger the overlap area and the longer execution time for the robot. Moreover, the robot also requires a large enough memory to store all of the backtracking points.

An alternative approach is to divide the environment into a grid equal size cells, the cells being equal to the size of the robot. The Spanning Tree Covering (STC) algorithm (see Section 3) is a significant example of this approach, it defines two types of cell: mega-cells and sub-cells that create a spanning tree between mega-cells and a path covered [by a robot] over the sub-cells by moving along the spanning tree. It has been proven to enable the coverage of the DOE with limited of no overlapping. However, the major limitation [of the STC algorithm] lies in the coverage area not being optimal. Figure 2 (see 2a) demonstrates the limitations of the STC Algorithm; if a mega-cell is occupied by a part of an obstacle then the entire mega-cell is also seen as an obstruction occupied entirely by an obstacle, as such a robot will consider the entire cell an obstruction and fail to cover the cell area. Therefore, the DOE will not be entirely covered.

A proposed improvement in the STC algorithm is a Full-STC algorithm where a robot ignores only sub-cells occupied by obstacles. In this case, an overlap in DOE coverage is accepted in exchange for a large coverage area. The limitation of this algorithm is shown in Figure 2 (see 2b). Depending on the shape and size of the obstruction, the cell coverage area also changes.

Research in [13] by Luo & Yang proposes a neural network based approach to CPP predicated on grid maps along with other recent improvements in 2008 and 2015 [15,16]. The DOE is into grid cells (similar to sub-cells in STC) called neurons. The possibility nodes can carry activate values (free cell) or restrain values (fully occupied cell or partially occupied by obstacles). This approach is based on the neural propagation model of Hodgkin-Huxley (1952) [12] in order to ensure high-positive neurons will be propagated to the entire environment while restrained neurons will only spread within a narrow range of its neighbours [14,15]. The robot determines the next position by selecting the proximity of the highest positive neuron. The drawback of STC algorithms is that all cells that occupied partly by obstacles are bypassed.

Our SmSTC method presented in this paper aims to address the limitations of the related research considered by targeting the identification of optimal paths while avoiding all obstacles, preventing (or at least minimise) backtracking, and maximising the coverage in any DOE. The SmSTC algorithm aims to address:

Peer-reviewed version available at Robotics 2019, 8, 44; doi:10.3390/robotics8020044

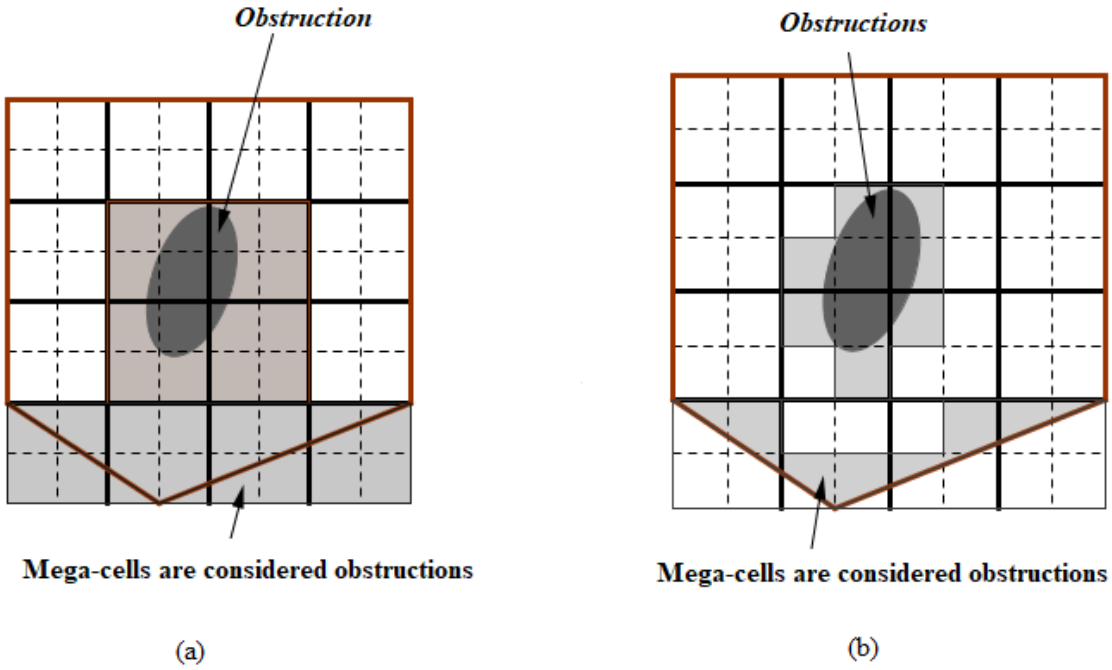


Figure 2. The limitations of STC and Full-STC algorithm when robots ignored many areas.

3. Robot Coverage Path Planning in Unknown Environments

In a static known environment, obstacles are fixed and the robot will anticipate environmental information before implementing coverage [23]. One of the classic methods to solve this problem is divide the DOE into non-overlapping sub-domains (cells) that contain no obstacles which called cell. These cells are easily covered with simple movements of robot like a zigzag line. Two cells are called adjacent if they share the same edge. An adjacent graph is used to represent the relationship between cells. The nodes represent the cells and the edges represent the links between the cells. The problem now is finding a path through all the nodes of the graph. (Figure 3 shows an adjacent graph for the algorithm).

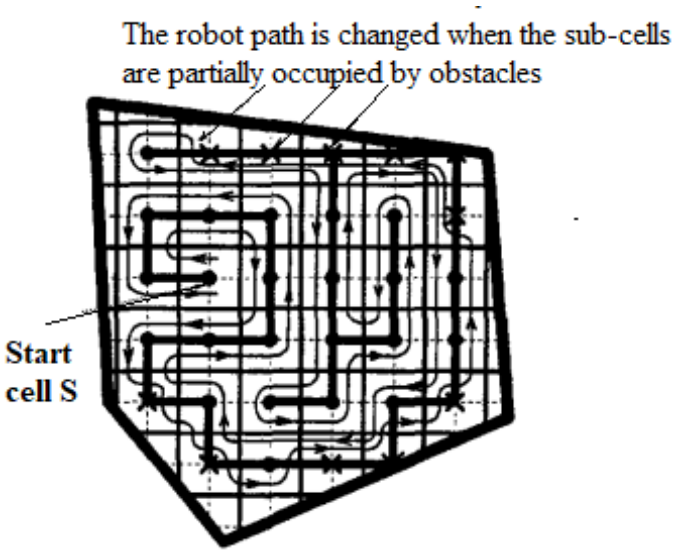


Figure 3. The adjacent graph for Full-SmSTC algorithm.

Gabriely and Rimon (2002) [6] proposed the Spiral Spanning Tree Coverage (SSTC) algorithm which is an on-line algorithm in which the path of the robot was built in a spiral; two SSTC algorithms

Peer-reviewed version available at Robotics 2019, 8, 44; doi:10.3390/robotics8020044

are introduced: SSTC and Full-STC. As with STC off-line, the entire grid is divided into mega-cells, each of them contains four robot size smaller cells. However, the robot is equipped with two additional sensors: a position-direction sensor and a sensor to detect obstacles [6,18]. Whenever the robot reaches a certain mega-cell that will be marked "visited" and against the new cells marked "unvisited". The robot will scan the clockwise direction to find the first neighbourhood marked "unvisited" in order to create a spanning tree. Then the robot moves to the right edge from the current cell to the selected neighbouring cell. The algorithm ends when the robot returns to its original position.

As with the off-line STC algorithm, SSTC will ignore the mega-cells that are partially or wholly occupied by obstacles; therefore, it still carries the limitations of STC. Along with the Full-STC algorithm, the SSTC algorithm ignores only the sub-cells occupied by obstacles and creates a pathway through the remaining free sub-cells. Figure 3 shows the path resulting from the use of the Full-STC algorithm.

4. Robot Coverage Path Planning Environments

4.1. Conceptual cells of robot

Similar to the STC algorithms, the grid division uses two cell types which are DxD-sized sub-cells and $2Dx2D$-sized mega-cell (D is the size of the robot). Each mega-cell will contain 4 sub-cells inside and Figure 4 shows two types of cell.

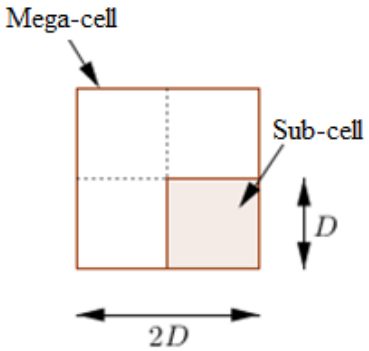


Figure 4. Figures showing two types of cell in STC.

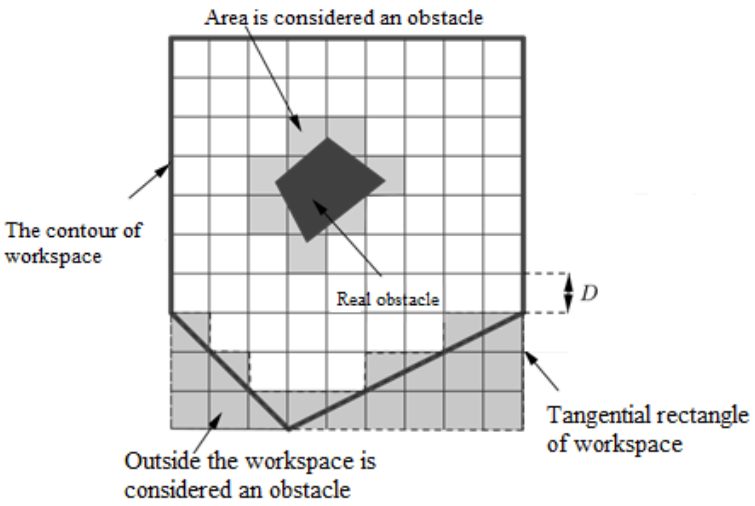


Figure 5. Divide the workspace in Smooth-STC.

Peer-reviewed version available at Robotics 2019, 8, 44; doi:10.3390/robotics8020044

Assume Q is the workspace of robot. Consider Q performing in 2-dimensional space ($Q \subset \mathbb{R}^2$) and Q_R is the smallest size rectangle that surrounds the Q. Assume Q_R is the $2mDx2nD$-sized ($m, n \in \mathbb{N}^*$) in other words Q_R is divided into $(m\ x\ n)$ mxn mega-cells of $(2D\ x\ 2D)$ size.

Suppose there are N obstructions in Q which are $O = \{O_1, O_2, ..., O_N\}$. Each obstruction O_i can occupy whole or part of the sub-cell. Space outside the work area of Q_R will be considered an obstacle. An illustration of this division is shown in Figure 5.

4.2. The Nodes in a Mega Cell and Sub Cells

Each mega-cell will be characterised by a node located centrally in the cell, the same configuration will apply to each sub-cell. Figure 6 shows the location of the nodes in the centre of the cells. These nodes play an important role in creating the spanning tree of the SmSTC algorithm.

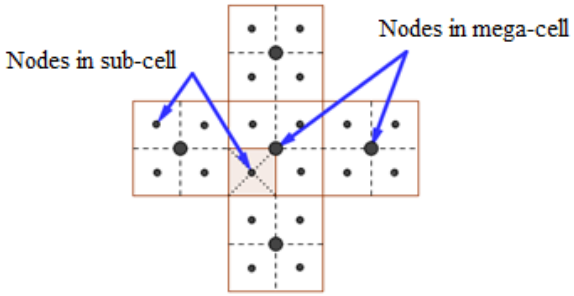


Figure 6. Nodes in cells.

4.3. The C-Space in the World of the Mobile Robot

The C-Space simply considers an object of any shape in this space (the DOE) as a point. Assume C is a C-Space where $q = (x, y)$ are points in the DOE occupied by the robot. $R(q)$ is a set of the q points. Obstructions O_i in C-Space is a form in which robots intersect with a real obstacle WO_i in the workspace ie: $O_i = \{q \in C | R(q) \cap WO_i \neq \emptyset\}$. The free space also known as the C_o free space which is a set of areas in which robots do not intersect with any obstacles WO_i.

Considering a circular robot represented by a centre (x, y) in space. Knowing the r radius of the robot can completely identify the set of points $q = (x, y)$ in the space occupied by the robot. $R_q = \{(x', y') | (x - x')^2 + (y - y')^2 \leq r^2\}$. Suppose the workspace of robot has only one obstruction (see Figure 7). Figure 7b showing the robot moves around an obstacle to determine the shape of an obstacle in the C-Space. At that time, the robot can be viewed as a point in C-Space and it can move arbitrarily without affecting obstacles (Figure 6c). Depending on the size of each robot that the shape of the C-Space is different (see Figure 7)

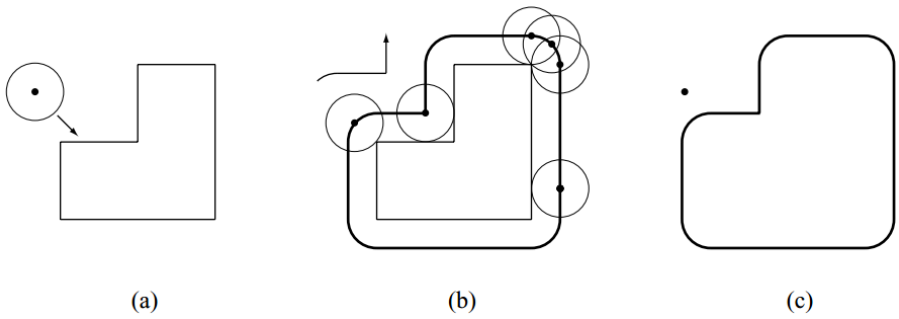


Figure 7. The process of creating C-Space

Peer-reviewed version available at Robotics 2019, 8, 44; doi:10.3390/robotics8020044

4.4. The adjacent graph

An adjacent graph $G(V, E)$ has a set of vertices V which is the set of nodes in the mega-cells and E is the connections between the nodes in two adjacent mega-cells. Smooth-STC uses this graph to create a robot cover path.

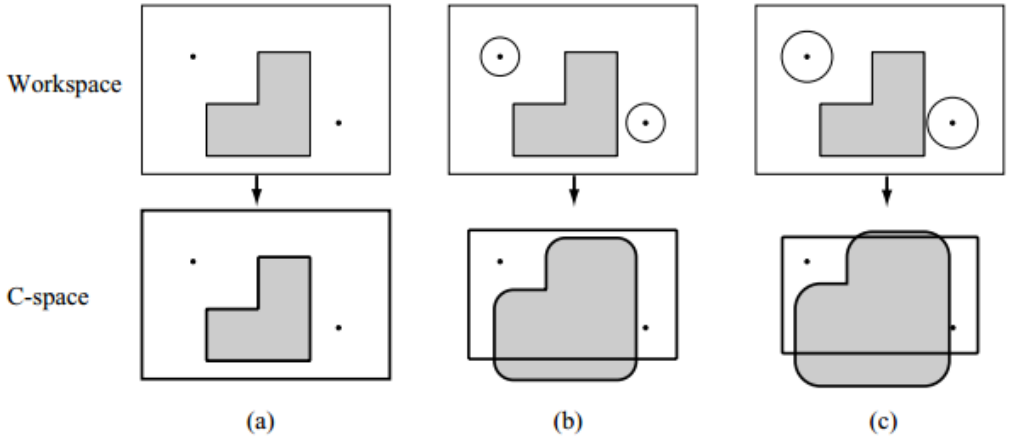


Figure 8. Different forms of space correspond to sizes of robot

4.4.1. Connection Edge Between Two Nodes of Two Adjacent Mega-Cells

Where the edge between two adjacent mega-cells is completely free, a connected edge between the corresponding nodes on the graph can be created and vice versa, then two nodes cannot be connected through this edge. Figure 8 shows the connections of adjacent mega-cells. Node N_c can create the connect with nodes N_e, N_s and N_w because the common edges between them (B_e, B_s and B_w) are free. Node N_c is not possible to create a connection to N_n since the common edge B_n is partially occupied by the obstacle. Figure 9 models this condition.

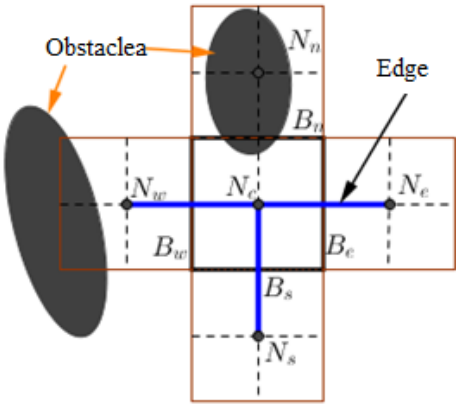


Figure 9. Connection edge between adjacent mega-cells

4.4.2. Input and Output Edges

If node N_i can create a connected edge E_{ij} to an adjacent node N_j, that edge E_{ij} is termed the input edge of node N_j and output edge of node N_i. In our proposed SmSTC algorithm, completely free mega-cells may have either an input or an output edge. However, with mega-cells partly occupied by an obstacle, they only have an output edge (i.e., there is no input edge). Mega-cells completely occupied by obstacles of course there will be no connection edges. Figure 10 shows an example of the input and output edge.

Peer-reviewed version available at Robotics 2019, 8, 44; doi:10.3390/robotics8020044

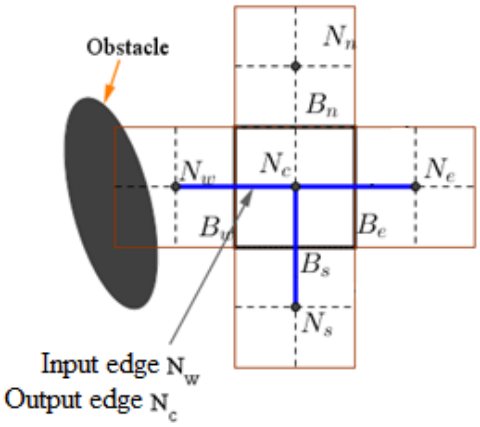


Figure 10. Input edge and output edge in Smooth-STC

4.5. The Proposed Smooth-STC algorithm

In the proposed algorithm to smooth the DOE and maximise the coverage area mega-cells partially occupied by obstacles are converted into a C-Space. The authors propose two types of SmSTC algorithms: *off-line* and *on-line*.

4.5.1. Smooth-STC in a Known Environment

Input : Environmental information is divided into mega-cells and the starting cell position *S* is the starting point for the robot to traverse the DOE.

Output: the coverage path of robot.

Algorithm 1: SmSTC off-line.

1. Start from create spanning tree using DFS.
2. Implementation coverage: Starting from a sub-cell of *S*.

    2.1. If the current mega-cell has both input and output edges: Move to adjacent sub-cells along the circumference of spanning tree in a anticlockwise direction.
    2.2. If the current mega-cell only has an input edge: Follow the boundary of the C-Space and return to the previous mega-cell.

In the SmSTC *off-line* algorithm, a spanning tree is created in step 1 using DFS algorithm. However, depending on the characteristics of each mega-cell, the connection edge between nodes is different. The execution of the robot in step 2 started from a sub-cell of *S*. The robot performs the right-side spanning tree to move around the spanning tree and ensure complete coverage (Step 2.1). Whenever the centre of the robot is located on the boundary of the C-Space the robot will move on the boundary from right to left and returns to the previous mega-cell (Step 2). The algorithm stops when the robot returns to the original mega-cell S. Details of this algorithm are illustrated in Figure 11.

4.5.2. The SmSTC algorithm in unknown environment

In the SmSTC *on-line* algorithm, robots will have on knowledge in advance relating to information in the DOE. They perform DOE coverage using sensors to detect obstructions. The SmSTC *on-line* algorithm continuously divides the working area into mega-cells and ignores the mega-cells completely occupied by obstacles. The remaining mega-cells will be used to create a graph called a spanning tree that each node is the centre of the mega-cells and each side has the ability to connect the adjacent mega-cells.

Each time the spanning tree is expanded, the robot divides the mega-cell into four sub-cells equal size for the robot. For every free mega-cell, the robot follow the path through each sub-cell surround the spanning tree. With mega-cells partially occupied by obstacles, robots perform calculations in

Peer-reviewed version available at Robotics 2019, 8, 44; doi:10.3390/robotics8020044

C-Space. Here, it performs a new border movement of C-Space of that mega-cell. The algorithm stops, when all mega-cells are covered or robot returns to the start position. A mega-cell is called "unvisited" if one of its 4 sub-cells has not been covered and vice versa mega-cell will be called "visited".

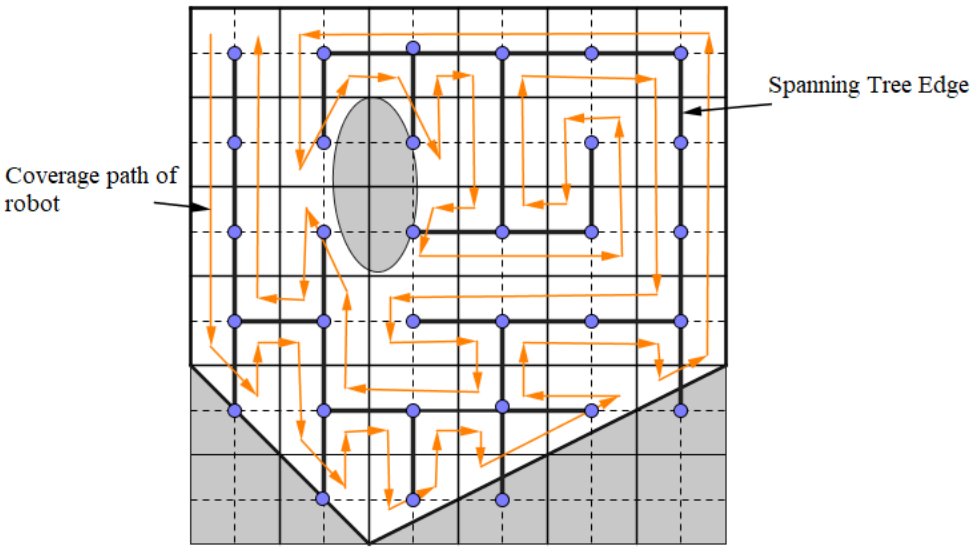


Figure 11. Input edge and output edge in SmSTC algorithm

SmSTC *on-line* Algorithm:

- **Sensors**: Position and direction sensors, detect obstacles in adjacent 4 mega-cell sensors.
- **Input**: Location of start mega-cell S and no information about the environment .
- **Recursive function**: SmoothSTC (w, x), where x is the current node, w is the parent node in spanning tree.
- **Initialisation**: SmoothSTC (null, S).

Algorithm 2:

SmSTC(w,x):

1. Mark the current mega-cell x as "visited"
2. While x still has an "unvisited" adjacent that is not completely occupied by an obstacle.
    - 2.1. Search for new neighbours of x in an anti-clockwise direction. Call y the first neighbour of x.
    - 2.2. Build a connection edge between x and y.
    - 2.3. If y is completely free:
        - From x move to a sub-cell of y along the output edge of x.
    - 2.4. If y is partially occupied by an obstacle:
        - 2.4.1. Build C-Space space at y.
        - 2.4.2. From x just moving along the new contour of C-Space at y has just followed the access edge of y to return to x.
    - 2.5. Call SmSTC (x, y)
3. End the loop.
4. If x differs from S, move back to x.
5. Returns results (SmoothSTC(w, x) stops).

- Finding adjacent mega-cells in the counter-clockwise direction at step 2.1 ensures the path of robot when it encircles the spanning tree in a uniform direction (Figure 12a;

Peer-reviewed version available at Robotics 2019, 8, 44; doi:10.3390/robotics8020044

- At step 2.2 if x is completely free and the robot is located at a certain sub-cell of x and it can move to an "unvisited" mega-cell y;
- An edge of the spanning tree $\vec{xy}$ will be created from the mega-cell x to y. This edge will be treated as the output edge from x and the input edge in y. With the mega-cell x partially occupied, it only has input edges;
- The coverage is carried out in steps 2.3 and 2.4 for two abilities to be completely free or partially occupied by obstacles. If y is completely free then from the sub-cell position in x the robot can move to a certain sub-cell in y by moving along the right side of the $\vec{xy}$ edge (Figure 12b;
- Because y is not occupied by an obstacle, this coverage path is always guaranteed in one direction (step 2.3). If y is partially occupied by an obstacle, the robot needs to calculate the C-Space surrounding y (step 2.4);
- Assuming a robot with a circle of centre I that can see its movement as a movement of the centre I. At this point, the robot still follows the edge to enter the y until its centre I lies on the road The boundary of the space C-Space moves centre I on this boundary from right to left of the $\vec{xy}$ edge, until the centre I is on the other side of edge $\vec{xy}$ then the robot follows the edge to return to x (Figure 12);
- Call recursively for the current node y and parent node x (step 2.4). If the current mega-cell x_o has no neighbours marked as "unvisited" then x_o is a leaf node when the robot will move back to the parent node (Figure 12c) or x_o is the start node S. So, robot has finished covering and the algorithm ended. Figure 12 details the robot movement in the sub-cells of this algorithm.

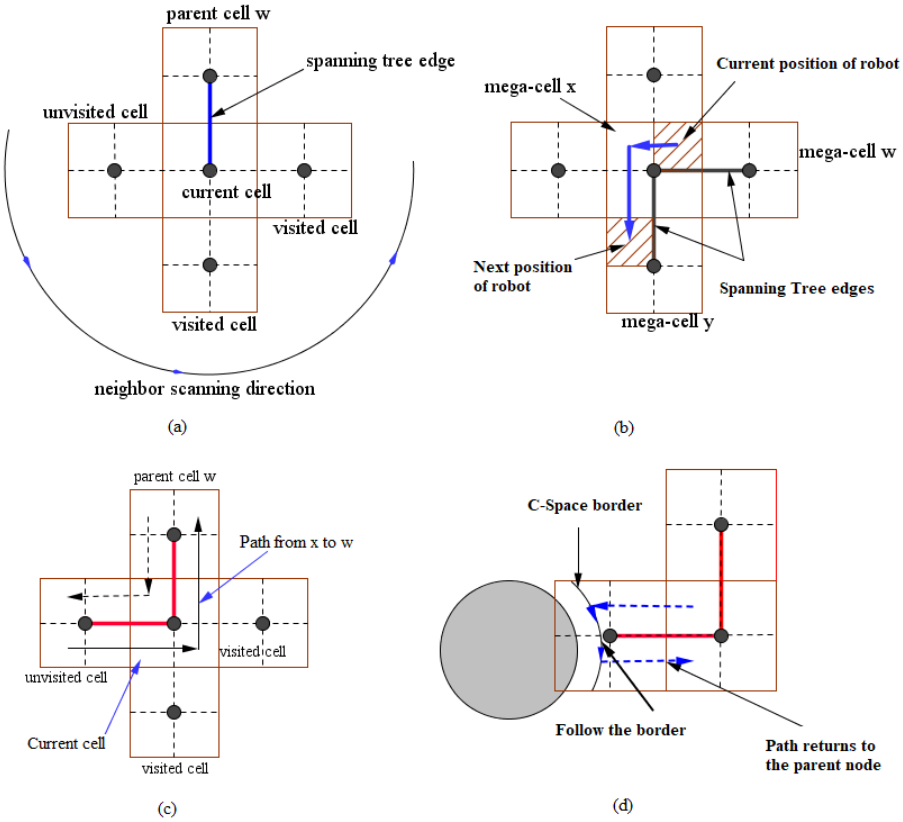


Figure 12. Search and move steps of robot

5. The proposed algorithm

The implementation of algorithm is built in the Java language and program interface uses Swing java library.

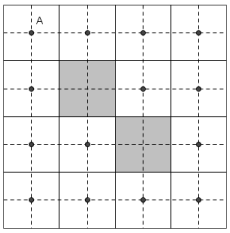
With the SmSTC algorithm *off-line*, the robot has *a priori* information relating to the DOE. Therefore, before finding the path, the algorithm needs to build an optimal spanning tree that can use any tree

Peer-reviewed version available at Robotics 2019, 8, 44; doi:10.3390/robotics8020044

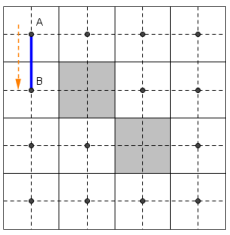
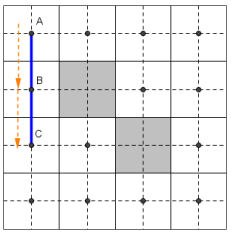
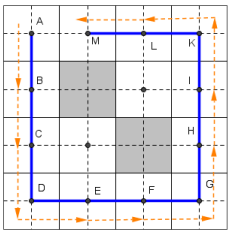
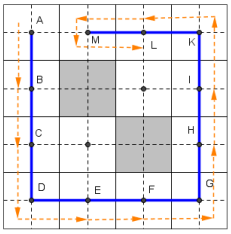
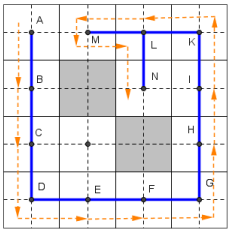
traversal algorithm, DFS is an example of such an algorithm. A spanning tree is generated that allows the robot to perform coverage by following this tree. The authors used a stack to save information about the nodes after each robot visit.

1. *Push* the start button in the stack, from the current node found node n which is the first unvisited neighbour in a anticlockwise order;
2. If the n node is not empty, move the robot from the current node to node n. Mark the current button as the previous node of node n (setPreNode is the current node). Mark the n button as visited;
3. If the next node n is empty, pop the current node off the stack.;
4. The robot moves from the current node to the stack top button. Mark the previous node of the top of stack with the current node.
5. This process repeats until stack is empty. The stack operation illustration is shown below.

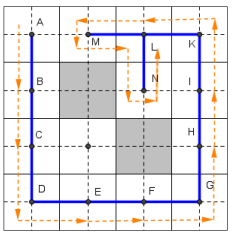
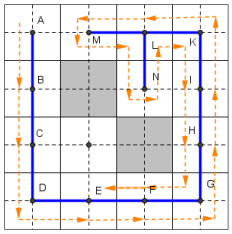
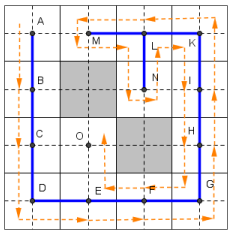
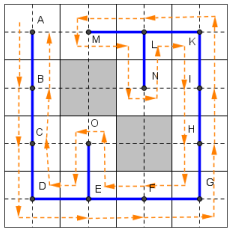
Table 1. SmSTC algorithm *off-line*

Push A in stack


Peer-reviewed version available at Robotics 2019, 8, 44; doi:10.3390/robotics8020044

Push B in stack		
S	A-B	
	…	…
Push C in stack		
S	A-B-C	
	…	…
Push M in stack		
S	A-B-C-D-E-F-G-H-I-K-L-M	
Pop M off the stack		
S	A-B-C-D-E-F-G-H-I-K-L	
Push N in stack		
S	A-B-C-D-E-F-G-H-I-K-L-N	

Peer-reviewed version available at Robotics 2019, 8, 44; doi:10.3390/robotics8020044

Pop N off the stack		
S	A-B-C-D-E-F-G-H-I-K-L	
Pop F off the stack		
S	A-B-C-D-E	
Push O in stack		
S	A-B-C-D-E-O	
	…	…
Pop A off the stack		
S	Null	
	Finish	

6. Experimental Results and Discussions

The evaluate our proposed SmSTC algorithm we conducted experimental testing using a simulation, all experimental runs use the same size of robot, the same starting position, and the starting time (the coverage ratio and the relationship to time for the Smooth-STC on-line algorithm is greater than our SmSTC on-line and Full-STC algorithms as shown in Figure 13.

During the first 6 seconds (of the experimental run) the coverage rates of all algorithms is the same because in that time there are no obstacles. From the 7th second, there is a difference: the runtime of SmSTC is shorter because of the shorter path travel distance resulting in reduced time to cover the DOE. The runtime for the Full STC and Smooth-STC algorithms are very similar is not much different; however, the achieved coverage rate of Smooth-STC is almost 100 % while this rate in Full STC is only over 93%.

Peer-reviewed version available at Robotics 2019, 8, 44; doi:10.3390/robotics8020044

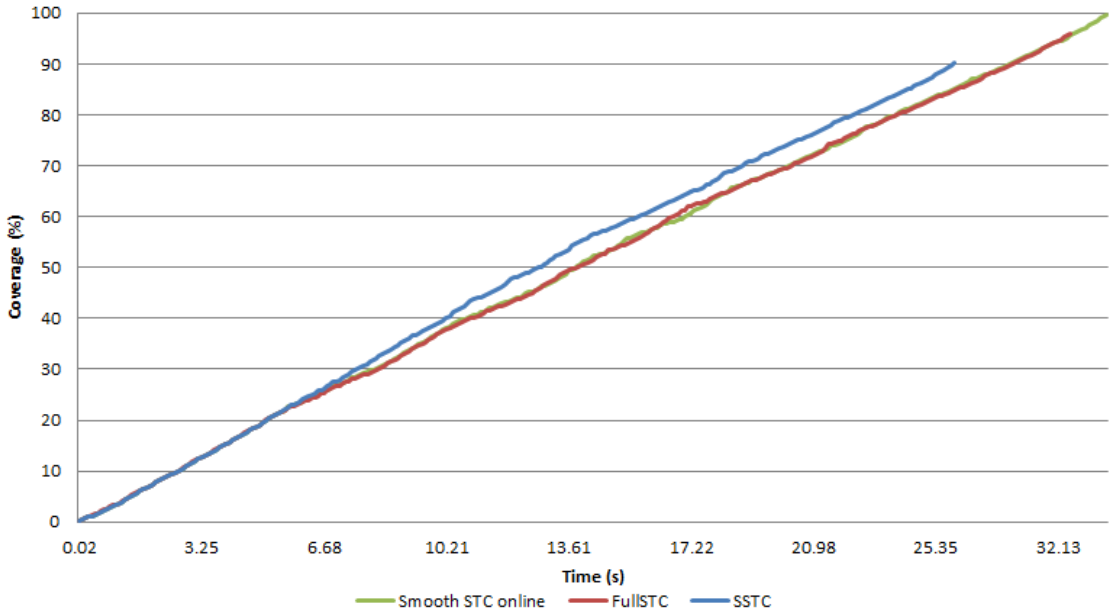


Figure 13. The coverage ratio depend on time of Smooth-STC, FullSTC and SmSTC

The influence of the robot size on the coverage ratio is realised by changing and gradually increasing the robot size. The results in Figure 14 show that the coverage ratio depends on the robot size and the bigger it's size, the lower the ratio. This indicates that the SmSTC algorithm achieved coverage ratio close to 100% in most cases and this result is an improvement over the alternative algorithms considered, this is especially evident when the robot size is bigger.

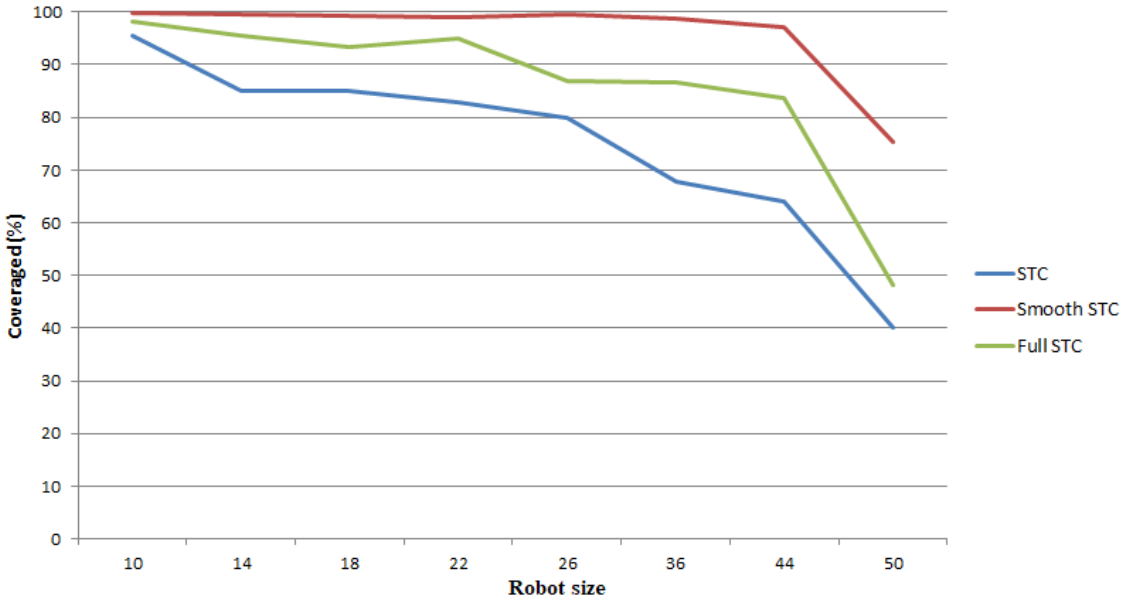


Figure 14. Effect of robot size on coverage rate

In the same map with the same complexity when changing the size of the robot, the authors obtained the overlap area ratio of those algorithms as shown in Figure 15. The ratio of overlap area of SmSTC proposed algorithm is still much larger than previous algorithms. The larger the robot size, the larger the repeating area. However, these overlap areas are mainly located around the obstructions. In fact, these areas have stain density more than areas far away from obstacles. So repeating in these areas is also meant to be cleaner.

Peer-reviewed version available at Robotics 2019, 8, 44; doi:10.3390/robotics8020044

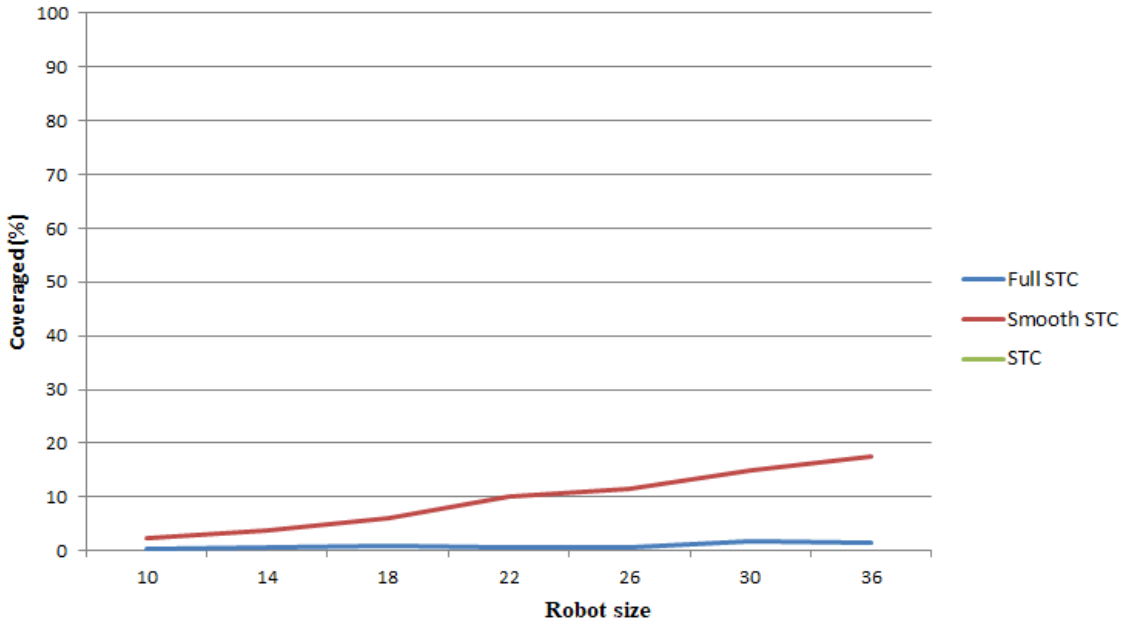


Figure 15. Overlap area ratio of algorithms when changing robot size

To compare the degree of influence of the complexity of the DOE to the coverage area, the experimental testing evaluated the e algorithms on maps with increasing complexity. The complexity of the map is objectively evaluated based on:

- Number of obstacles in the environment.
- The shape of each obstacle.

For a general robot with a size of 30, the experimental results are shown in Figure 16. The results results show that the higher the complexity of the environment, the better the STC and Full STC algorithms coverage. While our SmSTC always maintains a coverage of over 95%, it shows that SmSTC is effective in many different environments.

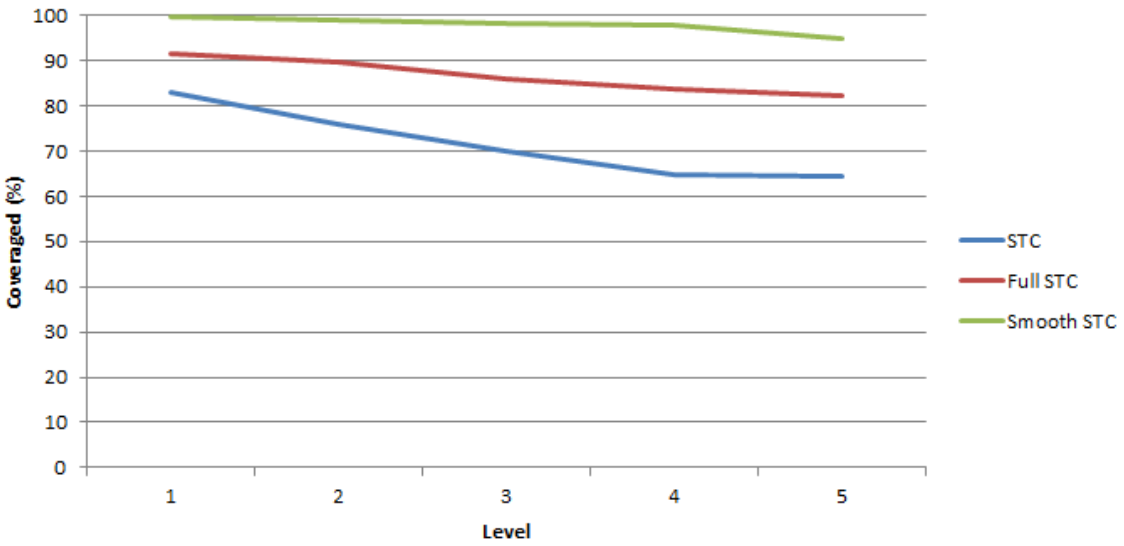


Figure 16. Effect of map complexity on coverage rate

Peer-reviewed version available at Robotics 2019, 8, 44; doi:10.3390/robotics8020044

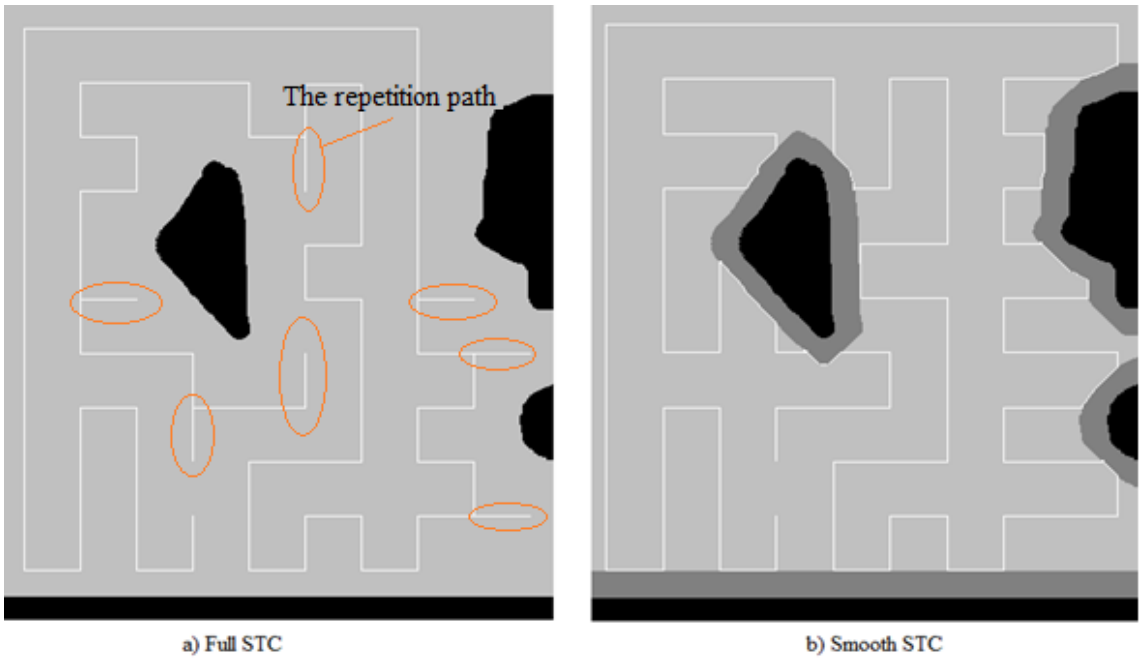


Figure 17. Path of Smooth STC compared to Full-STC

The proposed algorithm SmSTC algorithm generates a coverage path [for the robot] which avoids path repetition, minimises overlap, and avoids any backtracking during the movement as shown in Figure 17. In testing the Full STC algorithm made 7 turns in traversing the DOE while our SmSTC covered the DOE without any reversing or backtracking. In 'real-world' conditions, a sample test in a real case study of the CIST* BK robot (in plan-forest areas in Moc Chau areas) is shown in Figures 18 and 19. The real case study has demonstrated the most influence using sensors in uncertain environments. However, the coverage path along with the avoidance of multiple obstacles (in the fields of Moc Chau hill forest) shows that a real robot can deal with a real-word problems in different areas as shown in Figure 19.

Figure 18. A sample of Robot implemented in Smooth STC algorithm

Peer-reviewed version available at Robotics 2019, 8, 44; doi:10.3390/robotics8020044

Figure 19. A sample of Robot implemented in Smooth STC algorithm

To enhance the running time and performance of the real robot, we have improved a robot design with its smart sensors to capture paths while moving the fields of Moc Chau's hill forest. In addition, the robot has been faced with obstacles to identify trees or real obstacles. The experimental results shown the the proposed algorithm can be used in 'real-world' conditions to monitor agriculture at hill forest. The real robot has been implemented in square (DOE) of 60 x 40 m in Moc Chau's hill forest.

An outstanding problem is the inclusion of input and output for the robot that helps ensure a continuous path in coverage for monitoring in the forest. However, it also makes the total turning angle of the robot increase sharply. Therefore, the rotation angle of the robot when moving through the area around the obstacle changes constantly. This increases the time and cost of energy, to overcome this, there must be solutions to smooth out the areas around the obstructions with smart camera and the senses of the robot.

In considering potential directions for future research we contemplate further investigation into updating the rules in the knowledge base. Also, we plan to investigate combining various data sets from real case studies using data tracking logs, we consider that such data may be useful in finding optimal solutions for of considered rules in inference of the proposed system performance together with a robotic knowledge.

7. Conclusions

The proposed SmSTC algorithm solved the significant issue of finding an optimal coverage path while avoiding backtracking and achieving maximum coverage of the DOE. Experimental results indicate that the addition of C-space technique combined with the proposed model helps the robot to traverse the DOE effectively while avoiding collisions when entering dangerous areas where obstructions are located.

Our proposed SmSTC algorithm achieves an almost 100% coverage but still has overlap in areas located around obstructions. However, in reality, there are often more stains around obstacles than areas far away from obstacles. Therefore, repeating these areas also means cleaning the surface.

The proposed SmSTC algorithm achieves improved performance over the alternative algorithms considered and presents a useful 'real-world' approach to managing robotic operations in uncertain dynamic environments.

Acknowledgment

This work is supported by the Project No.T2018-PC-018, Hanoi University of Science and Technology.

Funding: This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-XXX.

Conflicts of Interest: The authors declare no conflict of interest.

Peer-reviewed version available at Robotics 2019, 8, 44; doi:10.3390/robotics8020044

References

1. Acar, E.; Choset, H. Sensor-based coverage of unknown environments: Incremental construction of morse decompositions. *International Journal of Robotics Research*, **2002**, *21*, 345-366.
2. Atkar, P.; Greenfield, A.; Conner, D.; Choset, H.; Rizzi, A. Hierarchical segmentation of surfaces embedded in r3 for auto-body painting. *Robotics and Automation*, **2005**, 572-577.
3. Cao, Z. L.; Huang, Y.; Hall, E. L. Region filling operations with random obstacle avoidance for mobile robotics. *Journal of Robotic Systems* **1988**, *5*, 87–102.
4. Choset, H. Coverage for robotics–a survey of recent results. *Annals of Mathematics and Artificial Intelligence* **2001**, *31*, 113–126.
5. Farsi, M.; Ratcliff, K.; Johnson, J. P.; Allen, C. R.; Karam, K. Z.; Pawson, R. Robot control system for window cleaning. *American Control Conf*, **1994**, *1*, 994-995.
6. Gabriely, Y.; Rimon, E. An online coverage algorithm of grid environments by a mobile robot. *Proceeding of the 2002 IEEE Robotics and Automation ICR*, **2002**, *1*, 954-960.
7. Gage, D. W. Randomized search strategies with imperfect sensors. *Mobile Robots* , **1994**, *8*, 270-279.
8. Galceran, E; Carreras, M. A Survey on Coverage Path Planning for Robotics. *Robotics and Autonomous Systems* **2013**, *61*, 1258–1276.
9. González, E.; Álvarez, O.; Díaz, Y.; Parra, C.; Bustacara, C. BSA: A Complete Coverage Algorithm. *Mobile Robots* , **2005**, *8*, 270-279.
10. Hert, S.; Tiwari, S.; Lumelsky, V. A terrain-covering algorithm for an auv. *Autonomous Robots*, **1996**, *3*, 91-119.
11. Hoang, H. V,.; Dang, V. H.; Laskar, N. U.; Chung, T. BA*: an online complete coverage algorithm for cleaning robots. *Applied Intelligence*, **2013**, *39*, 217-235.
12. Hodgkin, L.; Huxley, A. F. A quantitative description of membrane current and its application to conduction and exitation in nerve. *Journal of Physics London*, **1952**, *117*, 500–544.
13. Luo, C.; Yang, S. A real-time cooperative sweeping strategy for multiple cleaning robots. *Intelligent Control*, **2002**, 660 – 665.
14. Luo, C.; Yang, S. Approach to Complete Coverage Path Planning.*IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, **2004**, *34*, 718 - 724.
15. Luo, C.; Yang, S. A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments. *Published 2008 in IEEE Transactions on Neural Networks*, **2008**, *19*, 1279-1298.
16. Luo, C.; Yang, S. Safety Aware Robot Coverage Motion Planning with Virtual-obstacle-based Navigation*Transactions on Neural Networks*, **2015**, *19*, 1279-1298.
17. Mohd, N. Z.; Mohanta, J. C. Methodology for Path Planning and Optimization of Mobile Robots: A Review. *Procedia Computer Science* **2018**, *133*, 141-152.
18. Moravec, H.; Elfes, A. High resolution maps from wide angle sonar*Robotics and Automation*, **1985**, *2*, 116–121.
19. Najjaran, H.; Kircanski, N. Path planning for a terrain scanner robot. *Robotics*, **2000**, 132-137.
20. Palacin, J.; Palleja, T.; Valganon, I.; Pernia, R.; Roca, J. Measuring coverage performances of a floor cleaning mobile robot using a vision system. *Robotics and Automation*, **2005**, 4236–4241.
21. Shivashankar, V.; Jain, R.; Kuter, U.; Nau, D. Real-time planning for covering an initially unknown spatial environment. *In Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference*, **2011**.
22. Yasutomi, F.; Yamada, M.; Tsukamoto, K. Cleaning robot control. *Autonomous Robots*, **1988**, 1839–1841.
23. Zelinsky, A.; Jarvis, R. A.; Byrne, J. C.; Yuta, S. Planning paths of complete coverage of an unstructured environment by a mobile robot. *Proceedings of International Conference on Advanced Robotics*, **1993**, 533-538.