2017 2nd IEEE International Conference On Recent Trends in Electronics Information & Communication Technology (RTEICT), May 19-20, 2017, India

Complete Coverage Path Planning Algorithm For Known 2d Environment

Sumit Gajjar
Department of Electronics and Communication Engineering, GEC – Bhuj,
sumitgajjar19@gmail.com

Jaydeep Bhadani
Department of Electronics and Communication Engineering, GEC – Bhuj,
bhadani_jaydeep@gtu.edu.in

Pramit Dutta
RHRTD Division, Institute for Plasma Research, Gandhinagar, India,
pramitd@ipr.res.in

Naveen Rastogi
RHRTD Division, Institute for Plasma Research, Gandhinagar, India,
naveen@ipr.res.in

Abstract— Path-planning is an important primitive for autonomous mobile robots that lets robots find the optimal path between two points. Typical path-planning algorithms deal with finding an optimized path from start to end using a map of the environment and the robot to be aware of its location with respect to the map. Complete coverage path planning algorithm, on the other hand, is used in application where an autonomous mobile robot is required to visit all the points at least once in the given environment size with known obstacles. In this paper, a complete coverage path planning algorithm is developed and tested on a actual hardware. The algorithm is development for a known 2D environment with static obstacles with an optimization on minimal coverage time. The scaling of computational time with increasing map sizes and simulation results for different types of obstacle geometries are also presented. This kind of algorithm can be used for area inspection, cleaning robots, painter robot, land mine detectors, lawn mowers, agricultural field machines etc.

***Keywords*—Coverage Path Planning, Obstacle Avoidance, MATLAB Simulation, Autonomous Guided Vehicle**

I. INTRODUCTION

Path planning algorithms have been a subject of research since the very beginning of robotic technology. These algorithms aim at providing an obstacle free trajectory for a robot to travel from its starting position to its target location. Motion path planning for autonomous agents are usually designed with optimization on various type of cost functions. These include, path length, energy consumption, travel time etc. Further, these planning algorithms, assume either a global behavior – known environment, or local behavior – partially known behavior[1]. Heuristic based planning algorithm have become very common recently. These algorithms use cost functions to determine the shortest path from start to end.

Coverage Path Planning (CPP) Algorithms ensure to visit all points in a given environment at least once. Such planning algorithms have a wide range of applications ranging from military to civilian surveys and inspection. An elaborate comparison and survey of various CPP algorithm has been provided in [2]. CPP algorithms for known 2D environment using template method has been presented in [3] with applications in cleaning task for large industrial units. Similar algorithm for coverage of unknown environments using cellular decomposition method has been discussed in [4].

In this paper, Complete Coverage Path Planning algorithm is developed using cost functions to find out the path which can cover the environment by visiting each point at least once. It uses the similar technique to the boustrophedon cellular decomposition. The difference in here is it follows the path based on the calculated cost which updates itself as the robot progresses within the map. The cost function used here is a two-dimensional array, which shows cost of each grid of map. The entire space is defined based on grids where, each grid size is factor of the automated guided vehicle size. Same way the obstacles are defined on the basis of the grid size used.

Section II and III of this paper define the procedure of selecting the initial cost and updating of the cost function when the robot advances with the coverage. Section IV provides details of the complete algorithm. Section V provides the results of the algorithm for various types of obstacle shapes and an analysis of the computational time with increasing grid sizes.

II. INITIAL COST EVALUATION

The path planner in this algorithm uses a cost function to find the optimal path for complete coverage using a heuristic approach. It is assumed in this algorithm that the coverage area has a bounded limit with static obstacles. The entire map is then divided in cells which is a factor of the robot dimension. The obstacles, may be greater than the robot dimensions and occupy a larger number of grids.

To start the path search, an initial cost is found for the given environment. It is assumed that the starting point of the robot within the map is either pre-defined or explicitly defined by the user. The initial cost evaluation function starts to evaluate the cost from the starting position. Each cell, say ***Cell i*** have eight possible cells surrounded to move as shown in Figure 1. Depending on these eight surrounding cells the current cell cost is determined. From these eight surrounding cells the cost is found based on the number of cells representing the obstacles or wall. The initial cost for all the cells in environment is derived using the above method.

978-1-5090-3704-9/17/$31.00 © 2017 IEEE

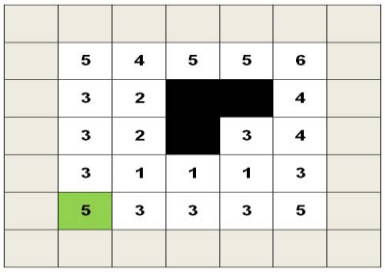

Figure 1. Initial Cost Evaluation

The above figure 1, shows the initial cost evaluation for a 5x5 maps with obstacle defined in of the cells. The evaluation starts from the starting cell, represented with green color. It also includes the obstacles as defined by the user, shown in figure with black cells. Evaluation starts from the starting node, as the starting cell has only 3 open cells surrounding it, the cost for the starting cell is defined as 5. Similarly, the algorithm defines an initial cost for all valid cells given environment size. The cost for the cell with obstacles is defined with infinite value and the cell with no surrounding walls or obstacles is defined to zero.

III. PATH COST EVALUATION

The coverage path for the environment can be mapped on the basis of the initialized cost. To map the path further we need to select the next cell to move. For a single cell, there are eight surrounding cells to move. Each and every cell has a cost defined known as the Coverage Cost(C-cost). The Coverage Cost is defined as shown in the below equation:

$$Coverage\ Cost = S - U \quad (1)$$

Where,

S=Total Number of Surrounding Cells
U=Number of Uncovered Cells

From this above equation cost of each cell is evaluated. Depending on this cost of each cell the next cell is selected. The Coverage Cost is denoted as C-cost in the algorithm.

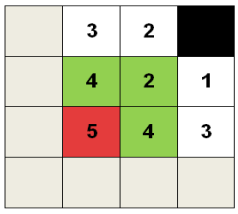

Figure 2. Path Cost Evaluation

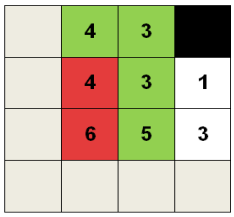

Figure 3. Updating Path Cost

These cells are represented in an array of C-cost which updates after each robot movement depending on the path covered. Now before selecting the next cell, cost of the neighboring cells is updated as shown in Figure 2. When the current cell, represented with red color in Figure. 3, is covered by the vehicle, the cost of the remaining valid cells in the neighborhood is incremented by one. The next cell, is then selected is the cell with maximum Coverage Cost but less than 8. This procedure is then repeated until the cost of all the cells are incremented to 8.

IV. COMPLETE COVERAGE PATH CALCULATION

To calculate the complete coverage, path the algorithm is explained using the Flow-chart shown in Figure 4. The path planning starts with defining the map with obstacles present and starting cell of the robot. Initial Coverage Cost(C-cost) mapping is used further to identify walls and obstacles defined prior to path planning. Then starting from the initial cell the C-cost updating carried out on the Cost calculated previously.

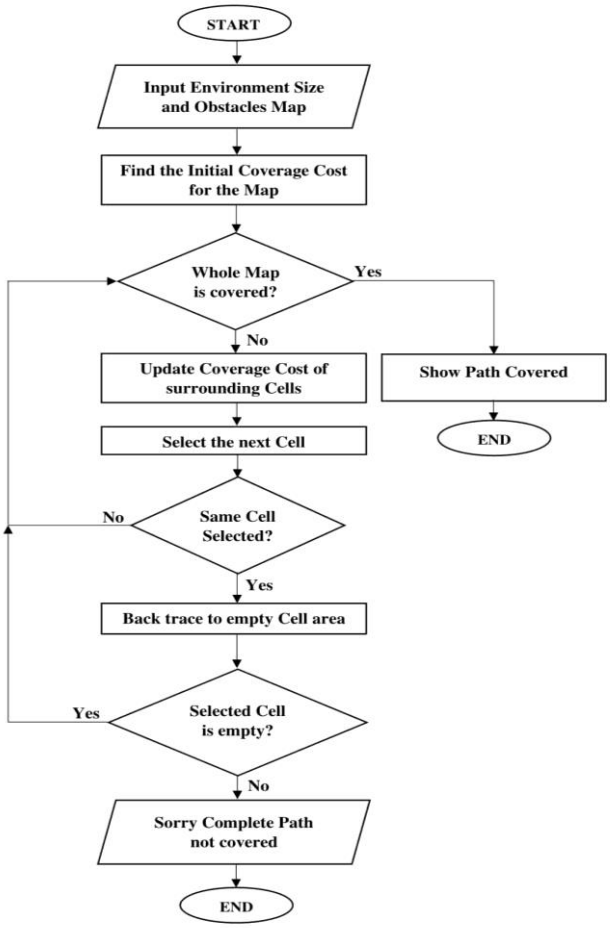

Figure 4. Flow Chart for Coverage Path Planning

Before updating the cost or selecting the next cell, a check is performed to ensure if the entire map coverage is completed. For selecting the next cell the diagonal cells are ignored from the horizontal and vertical cells, a cell with maximum C-cost but cost less than eight is selected. Same

way further coverage path is mapped and the surrounding cells are updated according to path covered. The example of map covering the whole area / using the complete coverage path planning algorithm is as shown in Figure 5. The arrows represent the direction of movement of the robot within the map. A double arrow represents that the robot visited the cell twice in order to complete the map.

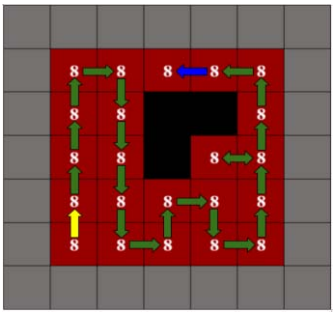

Figure 5. Final Costs after complete coverage

V. BACKTRACKING ISSUE

In case of cost based CPP algorithm as developed here, the issue of backtracking needs to be addressed with proper precautions. Backtracking is defined as over-riding the runtime algorithm when a cell is reached that is covered with obstacle and wall on all sides except for the last cell covered on the path. While previously the cell selected was the current cell so no next cell will be selected. Thus, to solve this issue, backtracking to path is done until it finds the empty area surrounding the cell. In this present algorithm, the backtracking is handled using an A-star algorithm current cell to last possible multiple option cell with a shortest path.

VI. ALGORITHM ANALYSIS

In this section, we provide the results of the algorithm for computational time scaling with increasing map sizes and obstacle avoidance for various geometries.

To evaluate this result, we used a specific system configuration of the computer which as shown below system configuration:

TABLE I. SYSTEM CONFIGURATION FOR ALGORITHM ANALYSIS

Processor:	Intel Core i5-5200 CPU @ 2.20 GHz
Installed memory (RAM):	4.00 GB
System type:	64-bit Operating System, x64-based processor

A. Environment Size:

According to the size of the environment, the computation time changes. To calculate the path calculation time of the given environment we take different size of environment, with same ratio of obstacles in all the environment. The algorithm is evaluated in MATLAB R2016b with CPU specifications as mentioned above in Table 1. The following are different input map sizes with obstacles the corresponding computation time:

TABLE II. COMPUTATION TIME FOR DIFFERENT GRID SIZE

Grid Size	Number of Obstacles	Computation Time (sec)
5x5	1	0.009
10x10	2	0.155
50x50	10	0.191
100x100	20	0.732

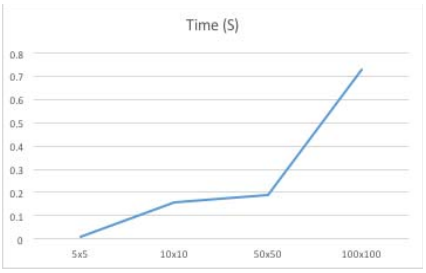

Figure 6. Computational Time Scaling Analysis

As shown, the computational time increases multifolds with increase in the grid size. This is can be improved by properly choosing the grid size to vehicle dimension ratio.

B. Obstacle Type:

On the basis of obstacle size and shape the resulting coverage path changes so for different shapes of obstacle the area covered are as:

1) Square Obstacle:

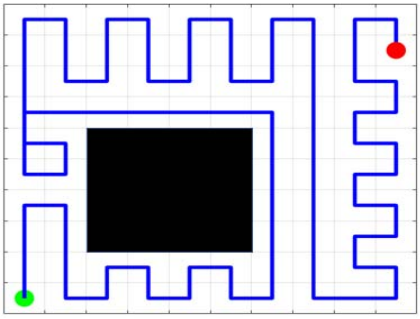

Figure 7. Path Planning for Rectangular Obstacle

When the square object appears in the environment we need to select the underlying cells for avoiding the obstacle in the coverage path. The square object is covered efficiently in here as the cells are square in shape.

2) Circular Obstacle:

For the circular shaped obstacle, the underlying grids need to be defined as the obstacle. The cells covered with small amount of obstacle area are not covered due to inefficient coverage due to robot dimensions.

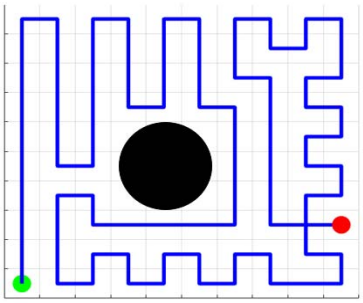

Figure 8. Path Planning for Circular Obstacle

3) Arbitrary Obstacle:

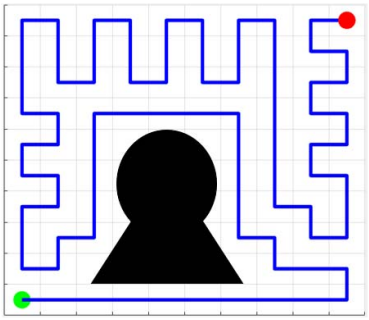

Figure 9. Path Planning for Arbitrary Obstacle

In the arbitrary shaped obstacle, same the underlying cells are not covered due to the obstacle dimensions.

C. Simulated vs Experimental Results:

The developed algorithm has been tested on a prototype mobile platform with Arduino Mega based controller. The MATLAB code output from the simulation is converted to a '*.csv' file that contains the details of the environment boundary, known obstacle positions and path to be followed.

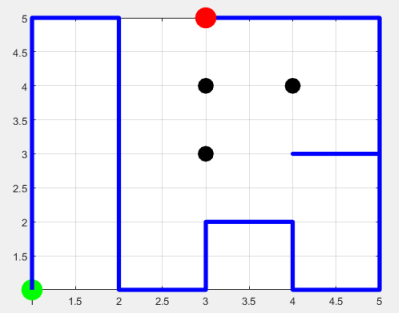

Figure 10. Path Planning from MATLAB

This file is loaded on the Arduino controller to initiate the movement. The present location of the robot is relayed back to the host PC using a USB communication. A 'Processing' application on the host system displays the coverage of the path in real time. The following example shows the path defined by the MATLAB algorithm and the processing application showing the position covered during operation for a 5x5 map with 3 obstacle cells.

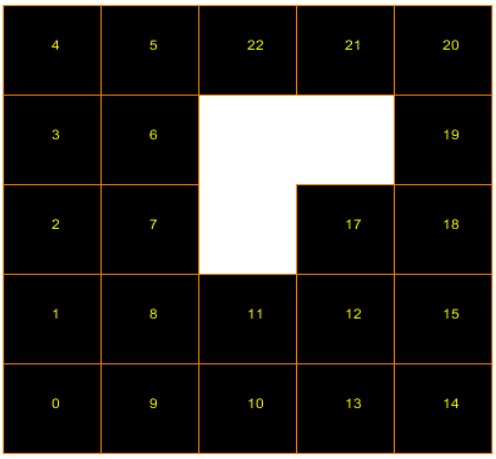

Figure 11. Processing Application for tracking the path

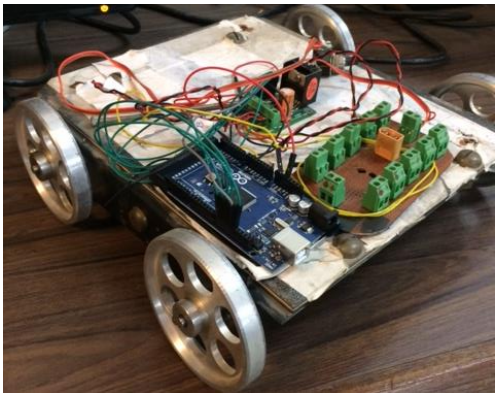

Figure 12. Robot used for Testing

VII. Conclusion and Future Work

In this paper, the development and testing of a complete coverage path planning algorithm is discussed and presented. The algorithm is efficient enough in traversing paths with different obstacle sizes and geometries. The computational time scaling shows that common desktop PC configurations can solve this algorithm within 1 second for map sizes below 100x100 grids. These algorithms can be used effectively in various applications like cleaning and surveying.

As a future work, this algorithm will be extended to include feedback of unknown obstacles from mobile agent using suitable ultrasonic sensors. The present communication link using USB can be used to update the model and recalculate the path when a new obstacle is found. Further, multi-agent based planning is already being developed where in, multiple mobile agents are used to cover the path with optimal time scale.

References

[1] X. W. X. Wang and V. L. Syrmos, "Coverage path planning for multiple robotic agent-based inspection of an unknown 2D

environment," 2009 17th Mediterr. Conf. Control Autom., pp. 1295–1300, 2009.

[2] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," Rob. Auton. Syst., vol. 61, no. 12, pp. 1258–1276, 2013.

[3] R. N. De Carvalho, H. A. Vidal, P. Vieira, and M. I. Ribeiro, "Complete coverage path planning and guidance for cleaning robots," Ind. Electron. 1997. ISIE '97., Proc. IEEE Int. Symp., vol. 2, pp. 677–682 vol.2, 1997.

[4] S. Brown and S. L. Waslander, "The constriction decomposition method for coverage path planning," 2016 IEEE/RSJ Int. Conf. Intell. Robot. Syst., pp. 3233–3238, 2016.