Journal of Computer Science 2012, 8 (12), 2032-2041
ISSN 1549-3636

doi:10.3844/jcssp.2012.2032.2041 Published Online 8 (12) 2012 (http://www.thescipub.com/jcs.toc)

TRAVERSAL ALGORITHM FOR COMPLETE COVERAGE

Kavitha Thiayagarajan and Coimbatore Ganeshsankar Balaji

Department of Information Technology, School of Computing, SASTRA University, Tamilnadu, India

Received 2012-07-12, Revised 2012-10-25; Accepted 2012-12-15

ABSTRACT

There are many applications which require complete coverage and obstacle avoidance. The classical A* algorithm provides the user a shortest path by avoiding the obstacle. As well, the Dijkstra's algorithm finds the shortest path between the source and destination. But in many applications we require complete coverage of the proposed area with obstacle avoidance. There are LSP, LSSP, BSA, spiral-STC and Complete Coverage D* algorithms which do not realize complete (100%) coverage. The complete coverage using a critical point algorithm assures complete coverage, but it is not well suited for applications like mine detection. Also for covering the missed region it keeps the obstacle as a critical point which is not advisable in critical applications where obstacle may be a dangerous one. To overcome this and to achieve the complete coverage we propose a novel graph traversal algorithm Traversal Algorithm for Complete Coverage (TRACC). Here the area to be scanned is decomposed into a finite number of cells. The traversal is done through all the cells after making sure the next cell has no obstacle. TRACC assures thorough coverage of the proposed area and ensuring that all the obstacles are avoided. Hence the TRACC always have the safer path while covering the entire area. It also reports the obstacle placed or blocked cell.

Keywords: Complete, Area Coverage, Traversal Algorithm, Path Planning, Demining Algorithm

1. INTRODUCTION

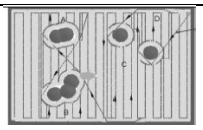
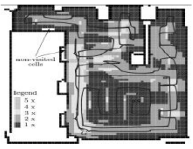
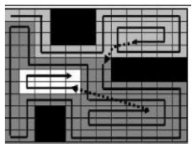
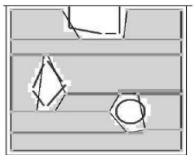
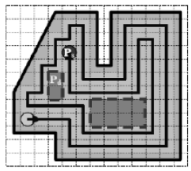
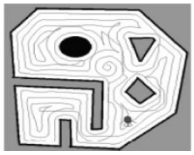
There are many classical algorithms for graph traversal and area coverage. The problems emphasized there are finding the shortest path, traversing through nodes and finding a path by avoiding the obstacles. But when applications require both full coverage and obstacle avoidance, there are very few algorithms like linked spiral path, complete coverage D* algorithm, complete coverage algorithm using critical points, backtracking spiral algorithm, linked smooth Spiral Path and spiral-STC which serves our needs. But those algorithms are not assuring full coverage. Also they are well suited only for soft real time systems. Therefore we need a heuristic algorithm which serves best to any hard real time application which requires real complete coverage and object avoidance. When it is a hard real time system like mine detection, the improper coverage cannot be taken easily as it costs human life. Hence it is very important to be stringent in covering the complete area and detecting all the mines. In this study, we overcome the detection failure because of improper scanning by proposing a novel graph traversing algorithm TRACC.

Applications such as humanitarian demining (Nicoud and Habib, 1995), lawn mowing (Huang *et al*., 1986) and floor cleaning (Ulrich *et al*., 1997) and harvesting (Ollis and Stentz, 1996) are widely using autonomous robots. These robots require a good path planning and complete coverage traversal algorithm to use their maximum potential. Among the various traversal and path finding algorithms, the conventional Dijkstra algorithm (Dijkstra, 1959) which finds the shortest path between nodes, the breadth first search and depth first search algorithms (Cormen *et al*., 2001) visit all the nodes in a graph by traversing them in a particular fashion, the A* algorithm (Russell and Norvig, 2003) which finds the path from start to end point by avoiding obstacles.

The complete coverage algorithm using critical points (Garcia, 2004) is enhanced of Choset's algorithms for both unknown (Acar and Choset, 2002) and known (Choset, 2000) environments.

Corresponding Author: Kavitha Thiayagarajan, Department of Information Technology, School of Computing, SASTRA University, Tamilnadu, India

Table 1. Literature survey of complete coverage and obstacle avoidance algorithms

Algorithm	Concept	Illustration	Coverage
Complete coverage using critical points	This algorithm covers the area other than the vertical vertex of the obstacle by forth and back motions.		100% coverage is assured.
Complete coverage D*	It is an extension of Path Transform algorithm which uses D* algorithm in place of wavefront algorithm to make quicker re-planning when the environment changes.		Near 100% coverage of the proposed area with changing environments.
Backtracking spiral algorithm	Region is covered using spiral filling paths. The covered regions are linked through backtracking mechanism.		An average of up to 93% coverage is achieved.
Spiral-STC	The robot covers the current sub region while recording its neighboring sub regions so that it may cover them as well.		Assured coverage is 85%.
Linked spiral path	Portions of the area are covered by spiral filling paths 94.99%. and Constrained Inverse Distance Transform is used to link all the portions.		Coverage achieved up to
Linked smooth spiral path	Coverage is achieved through circle following, wall following and object side following.		Assured coverage is 97%.

This algorithm assures complete coverage of the given area. The complete coverage D* algorithm (Dakulovi, 2011) implements the path transform algorithm (Zelinsky *et al*., 1993) with D* (Stentz, 1994), to give nearly 100% coverage. The Backtracking Spiral Algorithm (Gonzalez *et al*., 2005) is an extension of the basic BSA algorithm (Gonzalez *et al*., 2003). Refinements of backtracking spiral algorithm are proposed in spiral-STC (Wong and MacDonald, 2003), Linked Spiral Path (Young *et al*., 2009) and Linked Smooth Spiral Path (Lee *et al*., 2010) algorithms. Most of the grid based path planning algorithms uses zig-zag like pattern to traverse the path as discussed by (Gonzalez *et al*., 1996), (Hert *et al*., 1996) and (Acar and Choset, 2001).

Table 1 provides a literature survey of some of the complete coverage and obstacle avoidance algorithms.

2. MATERIALS AND METHODS

Though there are many algorithms available for complete coverage with obstacle avoidance, we propose a special algorithm-TRaversal Algorithm for Complete Coverage (TRACC). TRACC promises that no cell of the proposed area is left out without being sensed. Also, the objects (or) blocked cells in the area are found out. TRACC traverses to the next cell through the safest path and reaches its destination after sensing all the cells.

The TRACC gives us the path from the start to end by visiting each cell of the proposed area and ensures there is no obstacle. **Figure 1** is the graphical representation of the proposed region to be scanned.

We decompose the entire region into finite cells of 11 rows and 11 columns. S is the starting point and E is the ending point. The C1, C2, C3...C121 are decomposed cells of the region.

The cells with the circles are the blocked cells and which are to be avoided. The IA is an intelligent agent which is capable of sensing its four cardinal neighbors. The F, B, L and R for front, back, left and right respectively used for sensing. The algorithm for TRACC is given in **Table 2**.

The **Fig. 2** gives the flow path of IA, when there is no object. The IA will start from C1 and traverse to the next cell after checking each cell in its path. If it identifies the cell as safe, then it will proceed further to the next cell. After reaching the boundary line, i.e., C11, it will move to C33 through C22 and continues towards C23. Once the boundary is reached next it will move to C45 through C34.

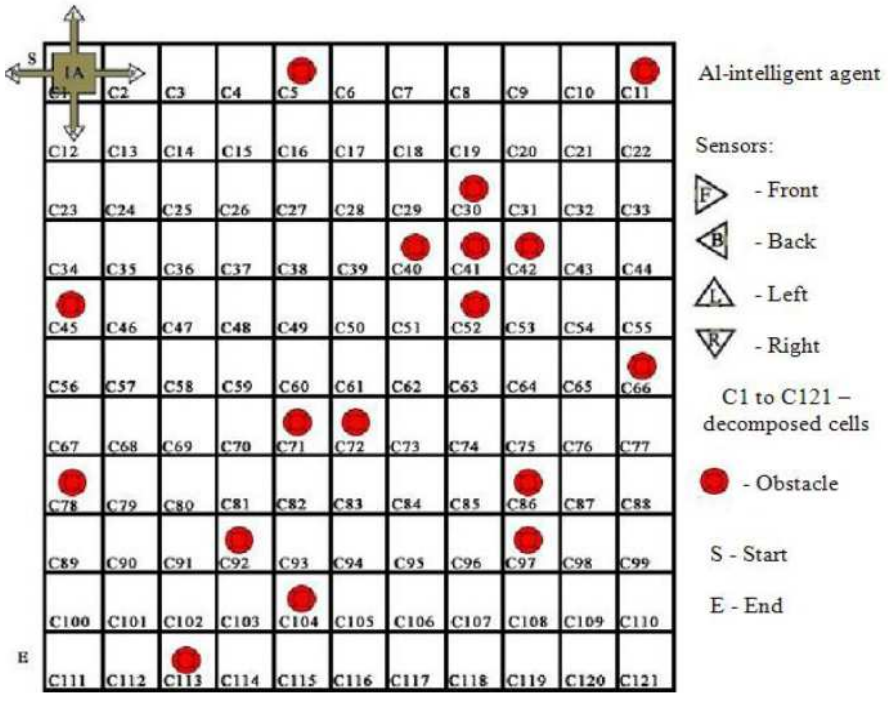


Fig. 1. Proposed region to be scanned (11×11)

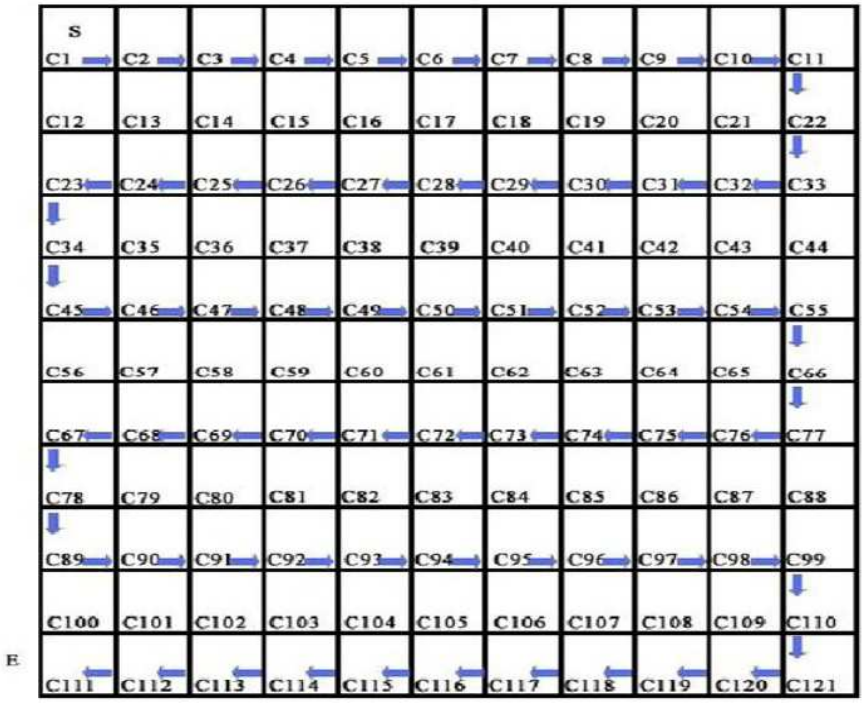


Fig. 2. The actual flow path

Table 2. TRaversal Algorithm for Complete Coverage (TRACC)

```
BEGIN
STEP1 scanCell(cell_id) if no
        obstacle
                move into the cell
                findNeighbour(cell_id) returns list of cardinal neighbors for all cardinal
                neighbors
                        scanCell(cell_id)
                                if no obstacle
                                        remove from uncheck list else if
                                obstacle
                                        remove from unchecked list
                                        add the cell_id in list_of_unsafe_cells cell_id =
                  next cell in the flow path
                  goto STEP1 else if
         obstacle
                shortestpath(cell_id, next cell in the flow path, list_of_unsafe_cells) returns the shortest
                and safest path list
                goto STEP1 for every cell in the shortest path list
STEP2 if the last cell in the flow path list is reached if
           list_of_unchecked_cells is not empty
                for the elements of list_of_unchecked_cells
                        if at least one of the cardinal neighbors of the element is safe
                                shortestPath(cell_id,unchecked_cell,list_of_unsafe_cells)
                        Else
                                exit(unchecked cell resides at blocked area)
END traverse successful with complete coverage
```

We have three major functions, scan cell (cell_id) which will scan a cell and returns true if obstacle is detected, FindNeighbour (cell_id) which will return a list_of_cardinal_neighbors, shortestPath (current cell_id, next unchecked cell in the flow path, list_of_unsafe_cells) which will find the shortest path between the current cell and the next cell by avoiding the unsafe (with obstacles) cells. We also have a list_of_unchecked_cells, which initially has all the cell_ids. After a cell is sensed by IA, the cell_id of the sensed cell is removed from the list_of_unchecked_cells.

3. RESULTS

The following assumptions are made to implement this algorithm:

- The Intelligent Agent (IA) can sense its four adjacent cells in its cardinal points viz., front, back, left and right
- The proposed area is virtually decomposed into equal sized cells
- The area of a cell the sensing area of the sensor
- The first cell has no obstacle

Initially, the agent IA starts from cell C1. Before the IA enters into each cell, it scans the cell to check whether the cell has obstacle. If there is no obstacle, then it moves into the cell. The cardinal neighbors of the cell are found and they are sensed for obstacle. If any of its neighbors are found with obstacle, then the cell_id is added in the list_of_unsafe_cells. Then the IA will take the next neighboring as per the flow path given in **Fig. 2** and continue scanning. This process is continued till the End (E). If there is any obstacle in the flow path, then the shortest path between the current cell and the next cell (in the flow path) is found by avoiding the unsafe cells. By repeating the above said steps, the IA reaches E. Once it reaches E, the list_of_unchecked_cells will be empty. If any cell_id exist in the list_of_unchecked_cells, then we check whether all its neighboring cells are in list_of_unsafe_cells.

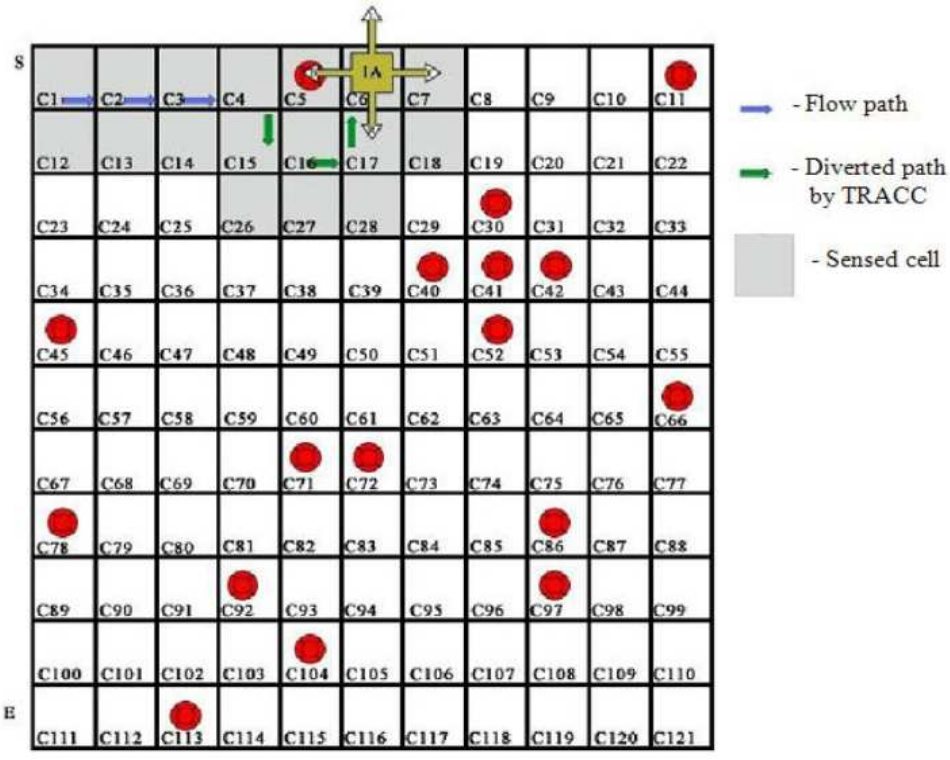
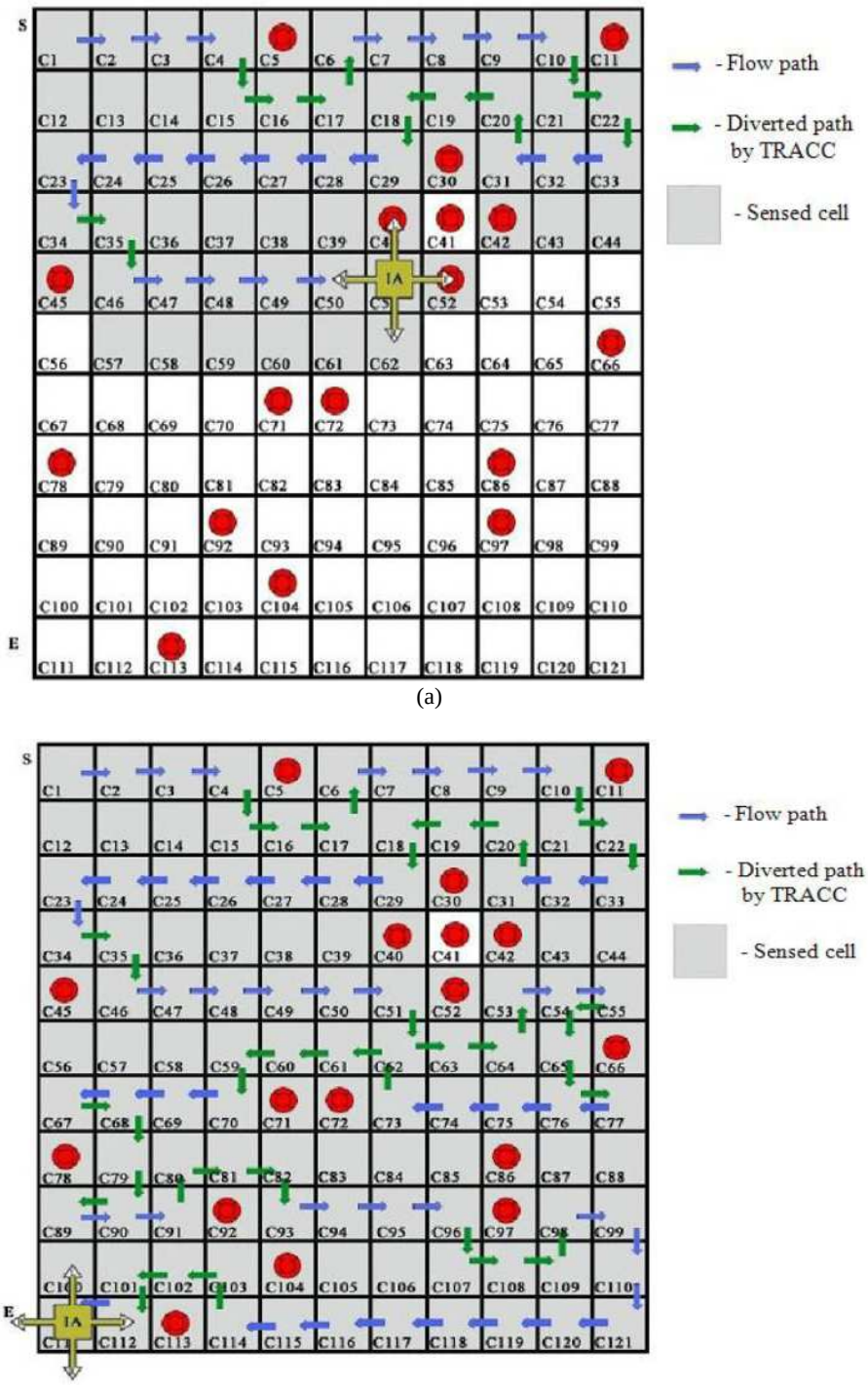


Fig. 3. Traversal by TRACC, avoiding the first obstacle in cell C5



(a)

(b)

Fig. 4. (a) Traversal of Intelligent Agent (IA) through the actual and diverted paths (b) complete coverage of the proposed area using TRACC

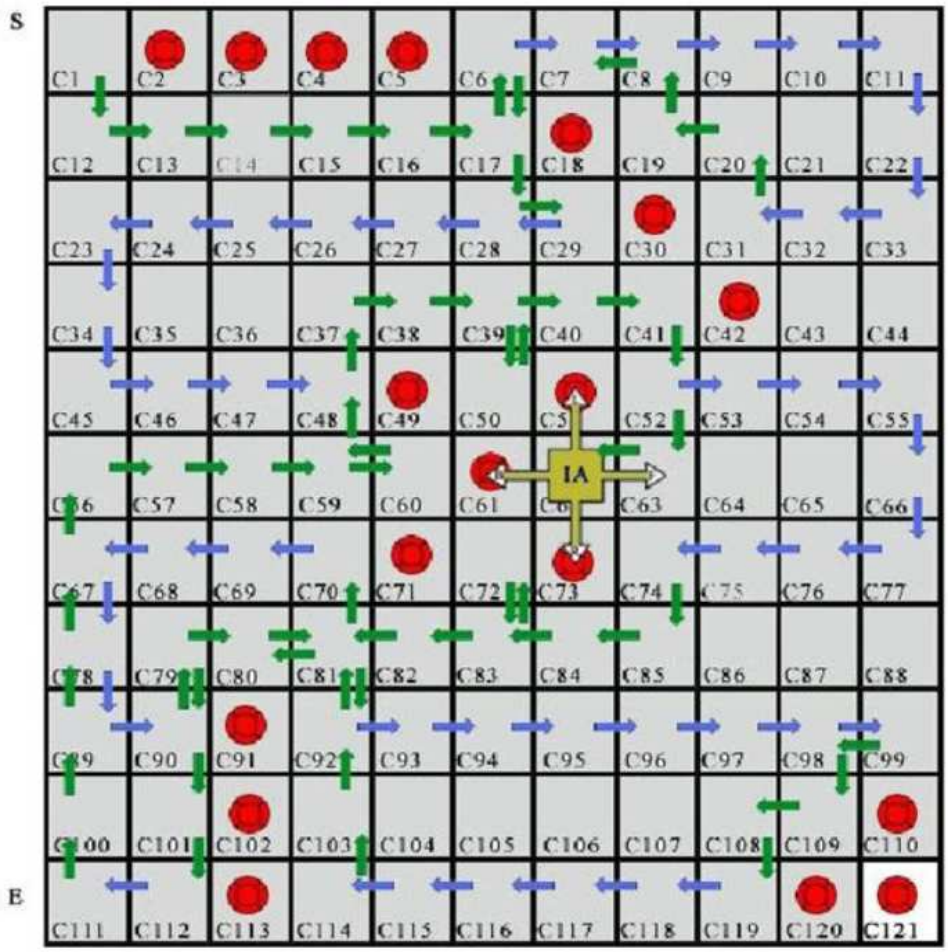


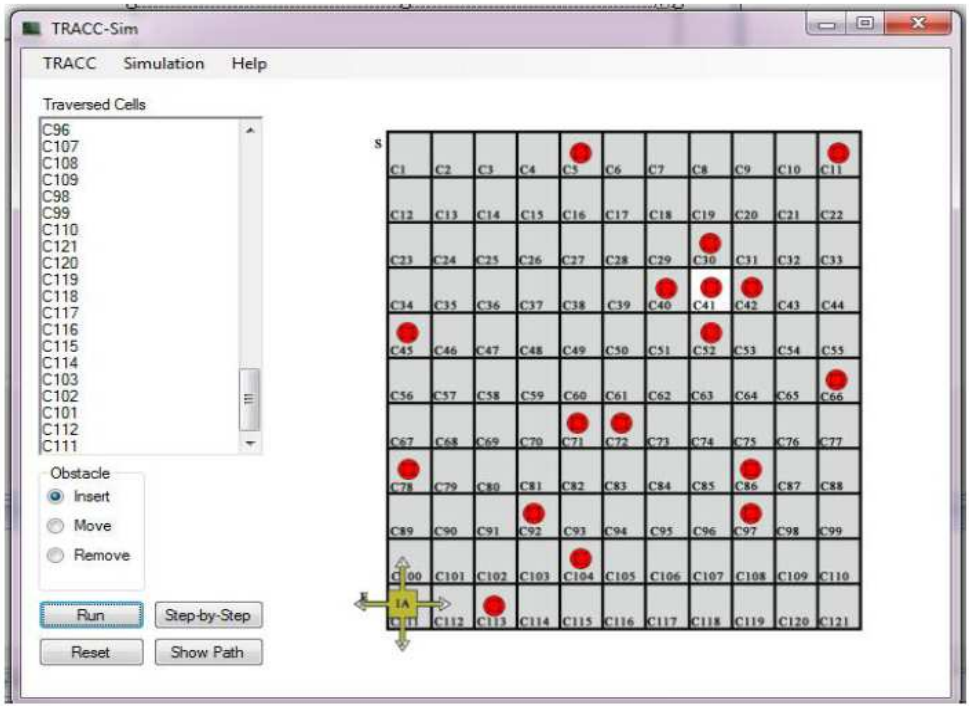
Fig. 5. TRACC implemented for a different scenario

If at least one of its neighbors is safe, then a shortest path is found between the current cell and the unchecked cell. The IA will traverse back through the path and scans the cell. This process may be repeated if more than one unchecked cell is present. But this case is very rare. Hence by using TRACC, we achieve 100% coverage of the proposed area.

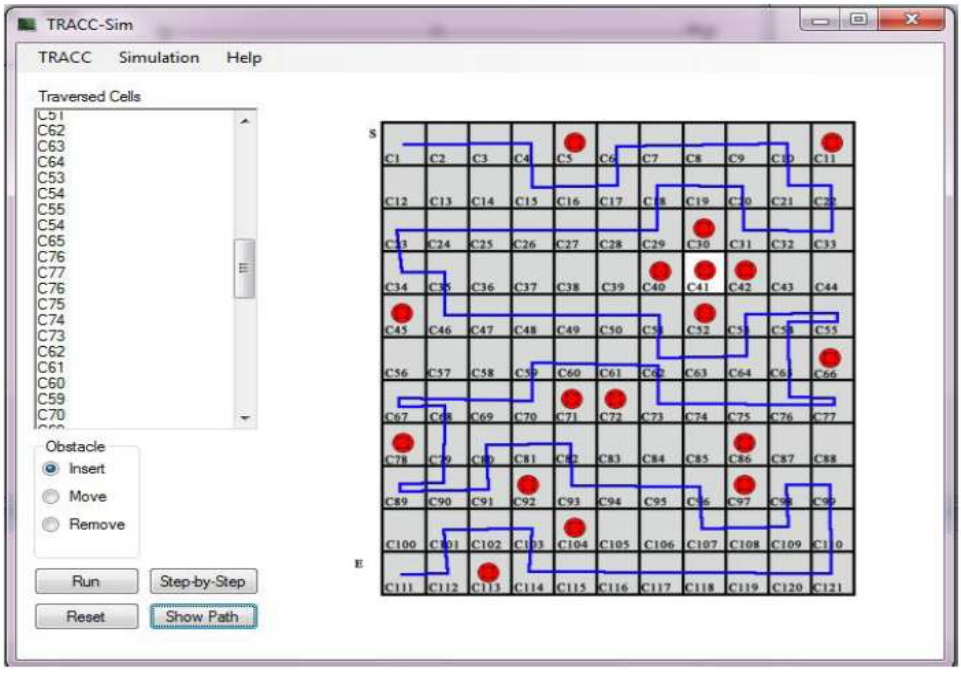
As shown in **Fig. 3**, the IA starts traversing from cell C1, scans its available cardinal neighbors-C2, C12 and moves to the next cell in the flow path-C2. The cells C2 and C12 are removed from the list_of_unchecked_cells. This process continues till it reaches C4. At this juncture, C5, which is the next cell to be traversed as per the flow path, is having an obstacle.

C5 will be appended to the list_of_unsafe_cells. Now, the IA must avoid this cell and move forward to the next cell to C5. This is done by finding the shortest path from the current cell C4 to the next cell to C5 (i.e., C6). So the IA moves through C4 =>C15 =>C16 =>C17 =>C6, which is the shortest path from C4 to C6. While traversing the path from C4 to C6, the IA will scan its cardinal neighbors of each cell it traverses. The blue arrow represents the flow path and the green arrow represents the diverted path taken by the IA with the help of TRACC in order to avoid the obstacle.

The IA uses TRACC to traverse further in the proposed area. The path taken by the IA to reach the end point E (C111) is shown in the **Fig. 4a and b**. The IA, after reaching C111, will check the list_of_unchecked_cells. The cell_id C41 will be present. IA will then check whether any of C41's cardinal neighbors are safe. As they are not safe in this case, IA will deduce that it is a blocked cell.



(a)



(b)

Fig. 6. (a) Simulation tool for TRACC (b) Path found by TRACC-Sim with complete coverage

TRACC algorithm is also implemented in different environments with challenging obstacles. The obstacles are placed at various places, forming varied patterns. **Figure 5** depicts one such a challenging environment. The path taken by the IA by implementing TRACC is shown in **Fig. 5**. In this scenario, after reaching C111, the cells C60, C62 and C121 will be available in the list_of_unchecked_cells. The shortest path is found and they are sensed one by one. As C121 is bounded by unsafe cells, it is discarded.

4. DISCUSSION

The simulation results reveal the following facts about TRACC.

4.1. Coverage

There is no possibility of uncovered area, unless the area is bounded by obstacles in all its four directions. Hence, we achieve complete coverage.

4.2. Redundancy

The redundancy rate is the ratio of the overlapped cell and the covered cell:

- Rr = (Uc/Cc) * 100 (1) Where
- Rr-is the rate of redundancy
- Uc-no. of uncovered cells
- Cc- no. of covered cells

According to TRACC's flow path, it is calculated as 45.45% of the cells are redundant. This redundancy may be considered as a drawback of this algorithm. But applications with sensors need more redundant cells to improvise the quality of sensing.

4.3. Cost

Though the complete coverage algorithm with critical points (Garcia, 2004) assures 100% coverage, it is costly in terms of time. The simulation of TRACC proves that, it is 1.71 times faster than the complete coverage algorithm with critical points.

Figure 6a and 6b show our simulator for TRACC algorithm, TRACC-Sim. The former shows the IA sensing the cells from C1 to C111. The later shows the path taken by the IA to reach C111 from C1. When the IA moves from one cell to another, the traverse cell updates in the box on the left side.

5. CONCLUSION

In this study we proposed a novel algorithm TRACC for complete coverage and obstacle avoidance. As we assured, our TRACC algorithm covers 100% of the proposed area. The simulation results show us it works perfect and visited all the cells through the safest path and avoided all the obstacles. As the results proved it is efficient, a prototype of a landmine detector is going to be developed using this algorithm.

By simulations, we infer that TRACC works the same as humans do. We have planned to apply this algorithm for land mine detection and applications which require full coverage as well obstacle avoidance. TRACC will also be used for applications like painting, lawn mowing and mopping. In this study, we have not considered the ordinal cells as neighboring cells. That will also be considered in the future explore to reduce the coverage time further.

6. REFERENCES

Acar, E.U. and H. Choset, 2001. Robust sensor-based coverage of unstructured environments. Proceedings of the International Conference on IEEE Retrieved from Intelligent Robots and Systems, Oct. 29-Nov. 3, IEEE Xplore Press, Maui, HI., pp: 61-68. DOI: 10.1109/IROS.2001.973337

Acar, E.U. and H. Choset, 2002. Sensor-based coverage of unknown environments: Incremental construction of morse decompositions. Int. J. Robo. Res., 21: 345-366. DOI: 10.1177/027836402320556368

Young, H.C., K.L. Tae, Choi, H.B. Sang and O.Y. Se, 2009. Online complete coverage path planning for mobile robots based on linked spiral paths using constrained inverse distance transform. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct. 10-15, IEEE Xplore Press, St. Louis, MO., pp: 5788-5793. DOI: 10.1109/IROS.2009.5354499

Choset, H., 2000. Coverage of known spaces: The boustrophedon cellular decomposition. Autonomous Robots, 9: 247-253. DOI: 10.1023/A:1008958800904

Cormen, T.H., C.E. Leiserson, R.L. Rivest and C. Stein, 2001. Introduction to Algorithms. 2nd Edn., MIT Press and McGraw-Hill, ISBN-10: 0262032937, pp: 1180.

Dakulovi, M., 2011. Complete coverage D* algorithm for path planning of a floor-cleaning mobile robot. Proceedings of the Preprints of the 18th IFAC World Congress, Aug. 28-Sep. 2, Milano, Italy, pp: 5950-5955.

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Num. Math., 1: 269-271.

Garcia, E., 2004. Mobile-robot navigation with complete coverage of unstructured environments. Robotics Autonomous Syst., 46: 195-204. DOI: 10.1016/j.robot.2004.02.005

Gonzalez, E., M. Alarcon, P. Aristizabal and C. Parra, 2003. BSA: A coverage algorithm. Proceedings of the International Conference on Intelligent Robots and Systems, Oct. 27-31, IEEE Xplore Press, Colombia, pp. 1679-1684. DOI: 10.1109/IROS.2003.1248885

Gonzalez, E., A. Suarez, C. Moreno and F. Artigue, 1996. Complementary regions: A surface filling algorithm. Proceedings of IEEE International Conference on IEEE Robotics and Automation, Apr. 22-28, IEEE Xplore Press, Minneapolis, MN., pp: 909- 914. DOI: 10.1109/ROBOT.1996.503888

Gonzalez, E., O. Alvarez, Y. Diaz, C. Parra and C. Bustacara *et al*., 2005. BSA: A complete coverage algorithm. Proceedings of the IEEE International Conference on Robotics and Automation, Apr. 18-22, IEEE Xplore Press, pp: 2040-2044. DOI: 10.1109/ROBOT.2005.1570413

Hert, S., S. Tiwari and V. Lumelsky, 1996. A terrain-covering algorithm for an AUV. Autonomous Robots, 3: 91-119. DOI: 10.1007/BF00141150

Huang, Y., Z. Cao and E. Hall, 1986. Region filling operations for mobile robot using computer graphics. Proceedings of the IEEE International Conference on Robotics and Automation, (RA' 86), IEEE Xplore Press, pp: 1607-1614. DOI: 10.1109/ROBOT.1986.1087504

Nicoud, J.D. and M.K. Habib, 1995. The pemex-b autonomous demining robot: Perception and navigation strategies. Proceedings of the IEEE International Conference on Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots, Aug. 5-9, IEEE Xplore Press, Pittsburgh, PA., pp: 419-424. DOI: 10.1109/IROS.1995.525830

Lee, T.K., S.H. Baek, S.Y. Oh and Y.H. Choi, 2010. Complete coverage algorithm based on linked smooth spiral paths for mobile robots. Proceedings of the 11th International Conference on Control Automation Robotics and Vision, Dec. 7-10, IEEE Xplore Press, Singapore, pp: 609-614. DOI: 10.1109/ICARCV.2010.5707264

Ollis, M. and A. Stentz, 1996. First results in vision-based crop line tracking. Proceedings of the International Conference on IEEE Robotics and Automation, Apr. 22-28, IEEE Xplore Press, Minneapolis, MN., pp: 951-956. DOI: 10.1109/ROBOT.1996.503895 10

Russell, S.J. and P. Norvig, 2003. Artificial Intelligence: A Modern Approach 1st Edn., Pearson Education, India, ISBN-10: 8177583670, pp: 1081.

Stentz, A., 1994. Optimal and efficient path planning for partially-known environments. Proceedings of the IEEE International Conference on Robotics and Automation, May, 8-13, IEEE Xplore Press, San Diego, CA., pp: 3310-3317. DOI: 10.1109/ROBOT.1994.351061

Ulrich, I.R., F. Mondada and J.D. Nicoud, 1997. Autonomous vacuum cleaner. Robotics Autono. Syst., 19: 233-245. DOI: 10.1016/S0921-8890(96)00053-X

Wong, S.C. and B. MacDonald, 2003. A topological coverage algorithm for mobile robots. Proceedings of the International Conference on Intelligent Robots and Systems, Oct. 27-31, IEEE Xplore Press, pp: 1685-1690. DOI: 10.1109/IROS.2003.1248886

Zelinsky, A., R.A. Jarvis, J.C. Byrne and S. Yuta, 1993. Planning paths of complete coverage of an unstructured environment by a mobile robot. Adv. Robotics, 66: 533-538. DOI: 10.1107/S1600536810016661