# A New Approach of Path Planning for Mobile Robots

Jitin Kumar Goyal[1]
Department of Electrical & Electronics Engineering
G L Bajaj Institute of Technology & Management, Greater Noida, India
(Email- [1]jitinnitj@gmail.com)

K.S. Nagla[2]
Department of Control & Instrumentation Engineering
Dr BR Ambedkar National Institute of Technology, Jalandhar, India
(Email- [2]naglaks@nitj.ac.in)

**Abstract- Path planning is fundamental task of the mobile robots navigations, where the accuracy of path depends upon the environmental mapping and localization. Several path planning approaches are already used for accurate path planning such as Dijkastra's Algorithm, Visibility Graphs, Cell Decomposition Technique, A* and Modified A* Algorithms etc. A* method does not support the accurate path planning if the size of the robot is larger than the size of the cell. In such situations it is difficult to move the mobile robot through narrow door or passage. This paper presents the new approach of path planning technique in which the virtual size of the obstacle present in the environment is assumed to be increased approximately *(2n+1)* times of the size of the cell. The experimental analysis of the proposed method shows the improvement in the path planning which reduces the chances of collisions. The paper is organized as follows: the first section of the paper represents the detail literature review and path planning strategies. The second part of the paper deals with the problem statement and proposed methodology. The last section shows the simulation results for indoor environmental path planning.**

**Keywords:** *Path Planning, A* algorithm, autonomous mobile robot.*

## I. INTRODUCTION

Path planning is considered as the important task for autonomous mobile robots. The main requirement of the accurate and optimal path planning is the accessibility of environmental and odometric information. This depends upon how accurate the environmental and odometric information is attained by the robot. In such situations the sensors play fundamental role. Another requirement in path planning is that the robot must reach the destination without colliding with obstacles and the path must be the shortest. The shortest path is considered as the time saving constraint for mobile robots and the safest path is considered as the safety constraints for mobile robots. To meet such requirement there are many path planning methods are already proposed by researchers.

Christos Alexopoulos et al. and J. Crowley described a learning technique in which the robot develops a global model and a network of nearby places. The developed network is used to plan the path for the robot and approach called 'The Visibility Graph' approach [1-2]. In the Visibility Graph approach the system plans and executes paths as a sequence of straight line motions where the shortest path is not always guaranteed. Another problem of this method is that the obstacles boundaries also considered as a path for the robot.

Vladimir J. Lumelsky et al. [3] introduced Bug algorithms (Bug1 & Bug2) for mobile robot path planning. In this technique the robot search its path as an insect (aunt), the robot moves directly towards its target and it follows the obstacle's boundary if the obstacle is in between the path. While following the obstacles boundary as the robot comes to know that its face is towards the target, it stops following the obstacle's boundary and moves towards the target. The problem of this technique is that it do not guarantee of shortest path.

Kambhampati S. et al. [4], Alexander Zelinsky [5] worked on the 'Quadtree' cell decomposition technique. In this technique the whole map is decomposed in four quarters, if the quarter is not occupied then it is not decomposed further but if the quarter is occupied then it is further decomposed and the process is repeated. A disadvantage of quadtree representation of free space is that it does not localize the effect of obstacles on the representation.

Dijkstra's Algorithm [6-7] is considered as an efficient method for the shortest path finding. Peter E. Hart et al. [8] proposed A* algorithm which is same like the Dijkstra's algorithm, but it use the heuristics to improve its efficiency in terms of the shortest path. Later on several modifications are made on A* algorithm and it is also modified according to the application [9-14]. In A* algorithm it is observed that if the robot dimensions are larger than the cell size then there are the possibilities of hitting the obstacle is high whrn robot run on the shortest path. Means, the shortest path may let the mobile robot very close to obstacles, which is not safe for robot and obstacles, so it is required to search a shortest path which consider the robot dimensions and the possibilities of hitting the obstacles is reduced. Therefore, in a proposed study a logically modified A* algorithm is introduced to search a shortest path which consider the robot dimension and to avoid the collision of mobile robot with the obstacles.

## II. PATH PLANNING

### *A. Detail example of A* Algorithm* [15]

In robotics, A* algorithm is mainly employed on an environmental grid in which the inclusion of the heuristic function h(n) make it different from the Dijkstra's algorithm [6]. The heuristic function selected as the distance function which gives the distance between the current

expanding cell and the goal. Here the function does not affect with the presence of the obstacle in the path. This heuristic function makes the algorithm efficient to get additional knowledge about the grid. A* search begins by expanding the start node and placing all of its neighbors on a stack. The stack is ordered according to the smallest $f(n)$ value that includes the heuristic function $h(n)$. The lower cost cell is then extracted and expanded. This process continues until the goal node is explored. The cost evaluation function is evaluated of each cell by:

$$f(n) = g(n) + h(n) \quad (1)$$

Where,
$n = (x_n, y_n)$ is the current expanding node or cell.

$f(n)$ = estimated minimum cost of the cell between all the paths from start node S to the goal node G forced to go through the node n.

$g(n)$ = the actual cost function from start node $S(x_0,y_0)$ to the current expanding node n.

$h(n)$ = the heuristic estimation of the minimum cost of a path from the current expanding node n to the goal node $G(x_G, y_G)$

Now $g(n)$ can be estimated as

$$g(n) = \begin{cases} \text{distance}(\text{prev}(n), n) & \text{if prev}(n) = S_0 \\ g(\text{prev}(n)) + \text{distance}(\text{prev}(n), n) & \text{else} \end{cases} \quad (2)$$

Where $\text{prev}(n) = S_k$, (read as previous n) and $S_k$ is the current node of the $k^{th}$ step, and in (2)

$$\text{distance}(\text{prev}(n), n) = 10\sqrt{(x_k - x_n)^2 + (y_k - y_n)^2} \quad (3)$$

And $h(n)$ in (1) can be estimated as

$$h(n) = 10\sqrt{(x_n - x_G)^2 + (y_n - y_G)^2} \quad (4)$$

This $h(n)$ is the Euclidean distance between the goal node and the current expanding node. The heuristic function can be taken as Manhattan distance or any other distance as per convenience of the path planner.

Now the lowest cost solution can be find out with the help of above equations and backtracked from the goal position G to start position S. the simulation results of this example is given in Fig. 1.

*B. Problem Statement*

The experiment conducted for indoor environment which consists of various obstacles like doors, walls, table, chair etc. The occupancy grid mapping of the environment is attained. The probability of cells is converted in the form of value '1' for obstacles and '0' for empty cell. For simulation the goal and the start position of the robot is defined manually on the grid map. Where the goal position is defined by placing the numerical value '-2', and the initial position is defined by placing '-1'. Now the traditional A* algorithm is applied to find the path between goal and the start node by applying (1) to (4). With this approach if the cell size is taken larger than the robot size then there is no problem with path planning. But the planned path may not be shortest. To get accurate path the resolution of the map decomposition should be increased means the cell size should be taken smaller than the robot size. But as we decrease the cell size the problem occurs with the traditional A* algorithm. Fig.1 shows that the robot is passing very close to the obstacles, which result that the robot may collide with the walls and other objects in the environment. On the other hand if the door passage is narrow the robot may trap. In our approach we proposed a modified method in which the virtual size of the obstacles is considered as larger than the actual size; the proposed algorithm is given in section *C*.

*C. Proposed Methodology*

To sort out the above problem the logically modified A* algorithm is proposed in this paper. In this method the environment is modified by placing the virtual obstacles around all the obstacles logically as per (5). For this all grid cells are tracked for value '1' in sequence. The logic of placing virtual obstacles is given below:

if map $(x_n , y_n) = 1$;
then map$(x_{n-1},y_{n-1}) = 2$,
map$(x_{n-1},y_n) = 2$,
map$(x_{n-1},y_{n+1}) = 2$,
map$(x_n,y_{n-1}) = 2$,
map$(x_n,y_{n+1}) = 2$,
map$(x_{n+1},y_{n-1}) = 2$,
map$(x_{n+1},y_n) = 2$,
map$(x_{n+1},y_{n+1}) = 2$, (5)

The cells which have the value '1' and '2' are put in the closed list. The closed list is the list of the cells in which the robot cannot move. The grid cells with value '2' are considered as virtual obstacles. Now the traditional A* algorithm is applied with proposed logic (5).The addition of logic (5) in A* is considered better for the robots whose size is equal to the size of the cell.

*Pseudo code for A* Algorithm*

Summarizing everything by the pseudo code is given here:

A. Generate the map with walkable and non-walkable nodes.
B. Create "open list" and "closed list" that is initially empty.
C. Place the entire non-walkable nodes on closed list.
D. Create start node and target node. Place start node on open list.

E. While open list does not become empty and path not found, do the following:
   a) Remove the node with lowest f(n) cost value from the open list. This node is now called *current node*.
   b) Change the status of current node as closed.
   c) Do the following for each adjacent node (eight in our case) of the current node:
      - If adjacent node is within the map boundaries and it is walkable and its status is not closed then do the following:
      - If adjacent node is not on open list then place it on open list, make current node as parent of this node, and store its g(n) cost and f(n) cost values. Also change its status to open.
      - If adjacent node is on open list (means its status is already open) then recalculate g(n) cost value. If new g(n) cost is less than already computed g(n) cost then change the parent node and store newly computed g(n) cost and f(n) cost values and resort open list.
   d) When the target node's status becomes open, the path is found.
F. When open list becomes empty, it means path does not exist.

For safety measures if the robot is two to three times larger than the grid cell, then (6) is proposed to expand the obstacle further with value '3'. In this step all cells are tracked for value '2' and the map is modified as per (6).

$$\begin{aligned} &\text{if map } (x_n, y_n) = 2; \\ &\text{then map}(x_{n-1},y_{n-1}) = 3, \\ &\quad \text{map}(x_{n-1},y_n) = 3, \\ &\quad \text{map}(x_{n-1},y_{n+1}) = 3, \\ &\quad \text{map}(x_n,y_{n-1}) = 3, \\ &\quad \text{map}(x_n,y_{n+1}) = 3, \\ &\quad \text{map}(x_{n+1},y_{n-1}) = 3, \\ &\quad \text{map}(x_{n+1},y_n) = 3, \\ &\quad \text{map}(x_{n+1},y_{n+1}) = 3, \end{aligned} \tag{6}$$

The cells which have the value '1', '2' and '3' are put into the closed list and traditional A* algorithm is applied with (5) and (6).

*Pseudo code for Proposed Method*

Summarizing everything by the pseudo code is given here when the robot size is equal to the size of the cell.

A. Generate the map with walkable and non-walkable nodes.
B. Place the virtual obstacles of value '2' around non-walkable nodes.
C. Create "open list" and "closed list" that is initially empty.
D. Place the entire non-walkable nodes and virtual obstacles on closed list.
E. Create start node and target node. Place start node on open list.
F. While open list does not become empty and path not found, do the following:
   a) Remove the node with lowest f(n) cost value from the open list. This node is now called *current node*.
   b) Change the status of current node as closed.
   c) Do the following for each adjacent node (eight in our case) of the current node:
      - If adjacent node is within the map boundaries and it is walkable and its status is not closed then do the following:
      - If adjacent node is not on open list then place it on open list, make current node as parent of this node, and store its g(n) cost and f(n) cost values. Also change its status to open.
      - If adjacent node is on open list (means its status is already open) then recalculate g(n) cost value. If new g(n) cost is less than already computed g(n) cost then change the parent node and store newly computed g(n) cost and f(n) cost values and resort open list.
   d) When the target node's status becomes open, the path is found.
G. When open list becomes empty, it means path does not exist.

## III. RESULTS

For the experiment consider an environment whose mapping in the form of occupancy map is already known. The map is of $50 \times 50$ sizes. The occupancy map is changed into the logical map by replacing '1' for the obstacles (the cells which are occupied) and '0' for the free space (the cells which are empty). The location of the robot and the goal is provided into the map and the modifications are applied into the map using the proposed methodology.

*Simulation Results*

Simulation results of traditional A* algorithm is shown in Fig.1 and the results of proposed method for different sets of the robot size are shown in the Fig. 2, 3 and Fig. 4. In Fig.1 it is observed that the searched path by the traditional A* algorithm allowed the mobile robot to move very close to the obstacle. In Fig.2 it is clear that the proposed method is planning the safe path for the mobile robot. Now if the robot size is two to three times larger than the cell size, then there is a possibility that the robot may collide with the obstacles. In such case for safe path planning the proposed logic is executed again as the results shown in Fig.3. If the robot size is four to five times larger than the cell size then for safe path (6) is executed and the result is shown in Fig.4. In this paper, the robot size is four to five times larger than the cell size, is chosen for the comparison with traditional A* algorithm. In Fig.5, two different paths are compared by using traditional A* algorithm and the proposed method. In Fig.5 (a), it is observed that the shortest path lets the mobile robot to move very close to the obstacles. In such situations

the mobile robot may collides the obstacles in the real environment. In Fig.5 (b), it is found that the problem has been solved by the proposed logically modified A* algorithm.

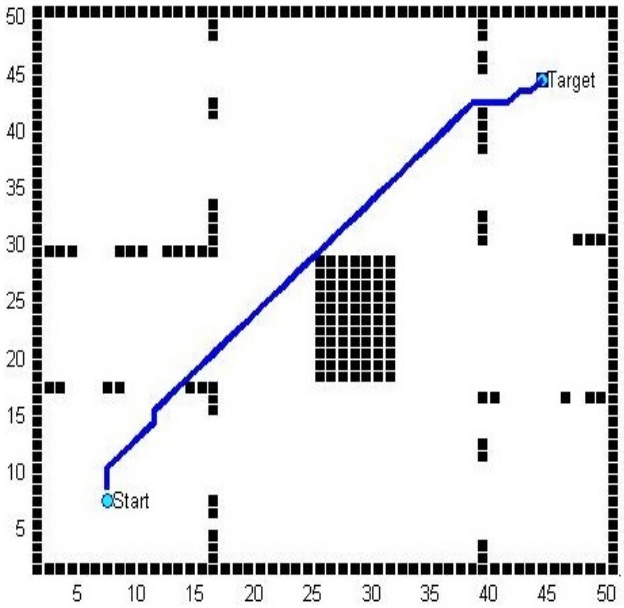


Fig.1 Path Planning by the traditional A* algorithm

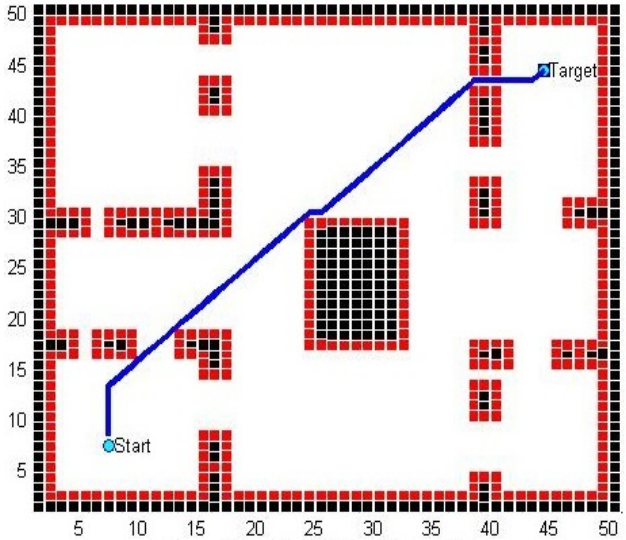


Fig.2 Path planning by the proposed method when robot size is equal to the cell size

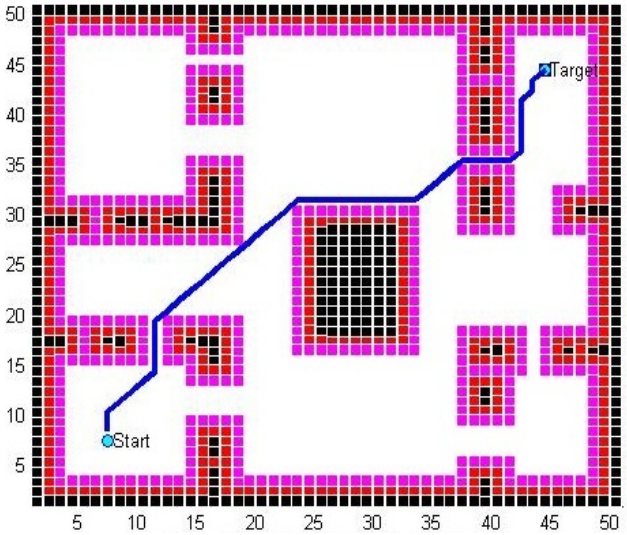


Fig.3 Path planning by the proposed method when robot size is two to three times larger than the cell size

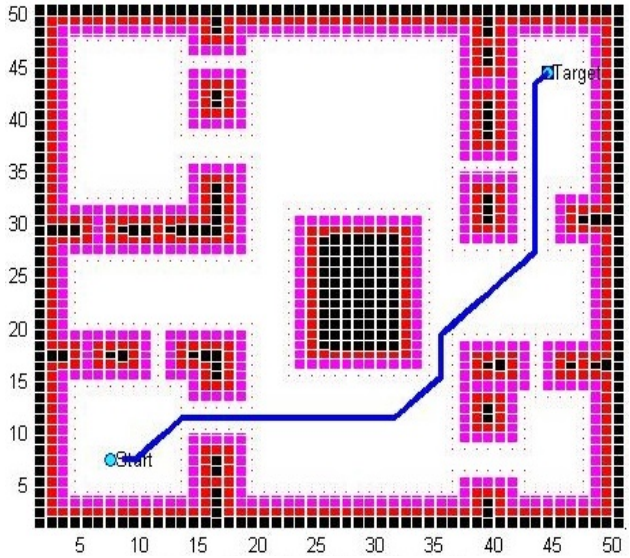
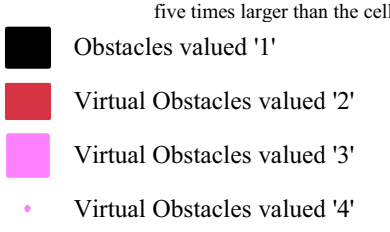
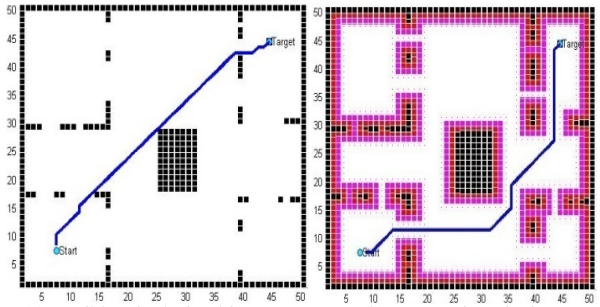


Fig.4 Path planning by the proposed method when robot size is four to five times larger than the cell size






(a) Traditional A* algorithm (b) Proposed Method

Fig.5 Comparison of the traditional A* and proposed path planning method

## IV. CONCLUSIONS

In this paper, a path planning method based on a logically modified A* algorithm is proposed to find a most safe path for mobile robots of particular dimension. The technique of putting virtual obstacle is used to enhance the size of obstacles in respect to the size of the mobile robot. With implementation of proposed method the robot does not collide with the obstacles. An example is presented to evaluate the performance of proposed path planning method in comparison with traditional A* algorithm. Although the A* algorithm can determine a shortest path, but the robot may collide with the obstacles in an actual environment. In the proposed method the path determined might not be shortest but it is guaranteed to be safe.

## V. REFERENCES

[1] Christos Alexopoulos and Paul M. Griffin, "Path Planning for a Mobile Robot," *IEEE transactions on Systems, Man and Cybernetics,* vol.22, no.2, pp.318 – 322 ,1992

[2] Crowley, J., "Navigation for an intelligent mobile robot", *IEEE journal of Robotics and Automation*, vol.1 ,no.1, pp.31 – 41, 1985

[3] Vladimir J. Lumelsky and Alexander A. Stepanov, "Dynamic path planning for a mobile automaton with limited information on the environment," *IEEE transactions on Automatic Control,* vol.31, no.11, pp.1058 – 1063, 1986

[4] Kambhampati S. and Davis L., "Multiresolution path planning for mobile robots," *IEEE Journal of Robotics and Automation,* vol.2, no.3, pp.135-145, 1986

[5] Alexander Zelinsky, "A mobile robot exploration algorithm," *IEEE transactions on Robotics and Automation*, vol.8, no.6, pp.707 – 717, 1992

[6] M. Barbehenn, "A note on the complexity of Dijkstra's algorithm for Graphs with Weighted Vertices," *IEEE Transactions on Computers*, vol.47, no.2, 1998

[7] E. W. Dijkstra, "A Note on Two Problems in Connecting with Graphs," *Numedsehe Yathematik*, vol.1, pp.269-271, 1959

[8] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths in graphs", *IEEE Transactions of Systems Science and Cybernetics*. vol.4, no.2, pp.100-107, 1968

[9] M. Fu and B. Xue, "A path planning algorithm based on dynamic networks and restricted searching area," *IEEE International Conference on Automation and Logistics*, pp.1193-1197, 2007

[10] I. Chabini and L. Shan, "Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks," *IEEE Transactions on Intelligent Transportation Systems*, vol.3, no.1, pp.60-74, 2002

[11] M. Nakamiya, Y. Kishino, T. Terada and S. Nishio, "A route planning method using cost map for mobile sensor nodes," *International Symposium on Wireless Pervasive Computing*, pp.168-174, 2007

[12] E. P. F. Chan and N. Zhang, "Finding shortest paths in large network systems," *9th International Conference on Advances in Geographic Information Systems*, ACM Press New York, NY, USA, 2001

[13] M. Hashemzadeh, "A fast and efficient route finding method for car navigation systems with neural networks," *10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06)*, pp.423-426, 2006

[14] G.E. Jan, K. Y. Chang, and I. Parberry, "Optimal path planning for mobile robot navigation," *IEEE/ASME Transaction on Mechatronics*, vol.13, no.4, pp. 451-459, 2008

[15] Chia-Jun Yu; Yi-Hong Chen; Ching-Chang Wong, "Path planning method design for mobile robots", *Proceedings of SICE annual conference*, pp.1681 – 1686, 2011