Wave Front Method Based Path Planning Algorithm for Mobile Robots

Bhavya Ghai and Anupam Shukla

Abstract Path planning problem revolves around finding a path from start node to goal node without any collisions. This paper presents an improved version of Focused Wave Front Algorithm for mobile robot path planning in static 2D environment. Existing wave expansion algorithms either provide speed or optimality. We try to counter this problem by preventing the full expansion of the wave and expanding specific nodes such that optimality is retained. Our proposed algorithm 'Optimally Focused Wave Front algorithm' provides a very attractive package of speed and optimality. It allocates weight and cost to each node but it defines cost in a different fashion and employs diagonal distance instead of Euclidean distance. Finally, we compared our proposed algorithm with existing Wave Front Algorithms. We found that our proposed approach gave optimal results when compared with Focused Wave Front Algorithm and faster results when compared with Modified Wave Front Algorithm.

Keywords Wave front · Path planning · Static environment · Mobile robot

1 Introduction

Path Planning is one of the key research areas in Dynamic Robotics [1]. Although the field of robot path planning is more than 30 years old but still it is an active topic for research. In very raw form, Path planning is moving of the robot from the starting node to the target node without any collisions. The evaluation criteria for Path Planning Algorithms may vary from Completeness [2], Computational Complexity [2], optimality, etc. as per the application. Path Planning can be modeled as a multi objective optimization problem [3]. Objectives may be to reduce

B. Ghai (✉) · A. Shukla
ABV—Indian Institute of Information Technology and Management, Gwalior, India
e-mail: bhavyaghai@gmail.com

A. Shukla
e-mail: anupamshukla@iiitm.ac.in

S.C. Satapathy and S. Das (eds.), *Proceedings of First International Conference on Information and Communication Technology for Intelligent Systems: Volume 2*, Smart Innovation, Systems and Technologies 51, DOI 10.1007/978-3-319-30927-9_28

the energy consumption, path length, execution time, communication delay, etc. [4] Mobile robots have wide domestic, military and industrial applications [4]. They may be used for cleaning where they navigate around the entire space [5]. They are widely used in dangerous environments which may be hazardous for humans such as aerospace research, mining industry, defense industry, nuclear industry, etc.

Path Planning algorithms may deal with known/unknown environments, static/dynamic [6] obstacles, single/multiple robots, 2d/3d space, etc. Numerous methods are employed to deal with all this problems such as heuristics, Genetic Algorithms, soft computing, statistical approaches, etc. In our case, we have dealt with static 2d environment. We have proposed a new approach based on wave front method. In wave front based methods, values are assigned to each node starting from target node. It is followed by traversal from start node to target node using the values assigned. Our major concern is to ensure optimal path length along with faster execution time. We tried to address this problem by preventing the full expansion of waves and used a new cost function so that optimality is not compromised. Finally, we compare our proposed approach with the existing wave front based path planning algorithms to verify the effectiveness of our proposed algorithm. We used Player Stage Simulator for testing. Player/Stage is a widely used open source multi robot simulator which is compatible with multiple platforms [7, 8].

This paper is organized as follows. Section 2 will discuss about the related work in this field. Section 3 will discuss about the major wave front based algorithms and also present our proposed approach (Optimally Focus Wave front Algorithm) to this problem. Section 4 will discuss about the assumptions and the comparison of our approach with existing algorithms. Sections 5 and 6 contain Conclusion and Future Work respectively.

2 Related Work

Numerous methods have been employed to solve different aspects of Path Planning Algorithm such as Heuristics [9], Wave Front Method [1, 2], Genetic Algorithms [10], Neural Network [11], etc. Some of the common examples include A*, artificial potential field, D*, etc. Environment for Path Planning algorithms can be modeled as grid of Polygons. Typically, it is modeled as a rectangular Grid but it also be modeled as a triangular grid so that the number of directions and hence path length can be further optimized [5]. Path planning algorithms are of two types based on data available about environment: static and dynamic. In static path planning, entire information about obstacles is known beforehand. We have the entire map of the environment at the beginning then we go for preprocessing based on the map and starting and goal node positions. The algorithm returns a path and then robot simply follow the co-ordinates of the path [1, 2]. In case of dynamic Path Planning,

Robot is dependent on its sensors. Only small fraction of information about obstacles is known in advance. The robot has to take navigation decisions while moving. As the robot moves and interacts with the environment, more information becomes available about obstacles. There are many algorithms for dealing with static environment effectively [2, 10–12]. In case of dynamic environment obstacles may change their position. This kind of situation is dealt using sensor information [4, 6, 9].

In case of Wave front algorithms, Robot moves from the source node to target node based on the waves emitted by the target [1]. In this paper we have purposed a modified version of Focused Wave Front Algorithm.

3 Proposed Approach

Wave Front algorithm uses breadth first search from the target node to the start node. In the wave front algorithm values are assigned to each node in increasing order from target node. The nodes in a wave i.e. nodes at equal distance from the target node are assigned the same value provided node is not an obstacle. The numbers assigned to cells in adjacent waves differ by 1. In our case, we have considered that robot can move in 8 directions so waves are square in shape. The following formula to assign value to each cell [1]:

$$\text{map}(i,j) = \begin{cases} \min(\text{neighborhood}(i,j)) + 1 & \text{Empty Cell} \\ \text{Nothing} & \text{Obstacle Cell} \end{cases}$$

Here i, j are the co-ordinates on the grid. Neighborhood (i, j) represents the cell adjacent to the cell (i, j). In our case, each node will have 8 neighbors. In every stage each cell who has not got any values will get values. This goes on until all the nodes in the map are assigned a value. After all nodes are assigned a value, Traversal from the start node begins towards the target node such that at each step it chooses the next node with minimum value. This algorithm always provides a path if it exists but it has two major drawbacks. Firstly, it is very time consuming and computationally expensive as it needs to explore all nodes i.e. it assigns value to each node. Secondly, it is possible that two or more nodes in the neighborhood have the same value. Hence, we have to choose the best path among different possible paths.

As the name suggests, Modified Wave Front Algorithm is an improved version of Wave Front Algorithm. The main advantage of MWF over wave front algorithm is that it returns the best optimal path. Like Wave Front Algorithm, MWF also explores all nodes and allocates value to each node in increasing order starting from the target node. The key difference lies in the way it allocates values to each node.

MWF differentiates between orthogonally adjacent and diagonally adjacent nodes. This is reflected in the following formula:

$$\text{map(i,j)} = \begin{cases} \min(\text{neighborhood(i,j)}) + 4 & \text{Empty Diagonal Cell} \\ \min(\text{neighborhood(i,j)}) + 3 & \text{Other Empty Cell} \\ \text{Nothing} & \text{Obstacle Cell} \end{cases}$$

The above formula describes the value allocated to the node with coordinates (i, j). This algorithm provides a solution if it exists(Completeness) and gives the optimal solution. The only drawback is that it is very slow as it explores all nodes. Specifically for bigger maps, it might take long to calculate optimal path.

Focused Wave Front Algorithm is a further modification to MWF. This algorithm is quite faster than previous algorithms because it explores only a limited number of nodes. Each node is allocated two values—weight and cost. Weight is the value assigned to node depending on its position. It is assigned in exactly same fashion as we allocate values in modified wave front algorithm. Weight can be understood as a measure of minimum path length of a node from the target node although it may be on a different scale. If we consider each node to be of unit length and we increment weight by 3 and 4 between adjacent nodes then weight of a node will be approximately 3 times of the path length from the target node. Cost of a node is its Euclidean distance from the start node.

Initially target node is assigned 0 weight. All its neighbors are assigned weight and cost value. The node with the minimum cost is expanded until source node is reached. This algorithm follows a greedy approach whereby it gives priority to those nodes which are near to start node. It reaches the start node quite swiftly. It is time efficient but not optimal. It will return a path if it exists although it may suggest a relatively longer route. It might not be suitable when movement cost is high and optimal path length is a priority.

Optimally Focused Wave Front Algorithm (OFWF) is our proposed approach. Optimal Path length is one of the most important properties sought in Path Planning Algorithms for a vast number of applications. OFWF is a further modification of FWF and it returns path with optimal path length. Like FWF, it explores only a limited number of nodes and hence is quite faster than MWF. FWF focuses on those nodes which are closer to source node irrespective of its distance from the target node. Hence, it doesn't provide optimal solution due to its greedy approach. On the other hand, OFWF weighs distance from the target node and approximate minimum distance from the start node equally. Weight is assigned in the same way as FWF. OFWF also expands nodes with minimum cost but cost is defined in a different fashion. Cost of a node with coordinates (i, j) is defined as follows:

$$\text{Cost(i, j)} = \text{Weight(i, j)} + \text{heuristic(i, j)}$$
$$\text{heuristic(i, j)} = 3 * ((\text{dx} + \text{dy}) + (\sqrt{2} - 2) * \text{Min(dx, dy)})$$

where

dx	absolute difference of x coordinates of the given node and start node
dy	absolute difference of y coordinates of the given node and start node
Min(dx, dy)	returns minimum value between dx and dy

***Algorithm*:**

Step 1: Insert target node into priority queue
Step 2: c = Pop node from priority queue
Step 3: if c == start node goto Step 7
Step 4: Assign weight and cost to neighbors of c
Step 5: Insert neighbors of c to priority queue
Step 6: Goto Step 2
Step 7: Traverse from start to target node by choosing the node with least weight among neighborhood at each step

Internally, Priority queue will arrange the nodes in ascending order based on the cost of each node. Instead of using Euclidean distance for measuring approximate minimum distance from start node, we use a variant of diagonal distance [13] which is better suited in our case as our robot can move in only 8 directions. Since this algorithm focuses on optimal path, so we decided to call it Optimally Focused Wave Front Algorithm (OFWF). Firstly, we push target node into priority queue. Then we allocate weight and cost to its immediate neighbors and push them into priority queue as well. Then we pop a node from the priority queue and repeat this process until start node is popped out. Lastly, we traverse from start node to target node by moving to nodes with least weight among other neighbors.

4 Results

We simulated MWF, FWF, OFWF using Player 3.0.2 and Stage 3.2.2 on Ubuntu 12.04 Platform. We feed the starting and target locations along with the environment as the input to the algorithm. The algorithm returns the set of x, y co-ordinates of adjacent cells which will form the path. We have used gray image to represent the 2d environment where black pixels represent obstacles and white spaces represent the free region. An image of size P × Q pixels represent a map of P × Q cells.

We have assumed that robot can move in 8 directions (North, West, East, South, North-East, North-West, South-East, South-West) and number of obstacles are finite and static. We have assumed that robot can rotate in clockwise and anti-clockwise direction, hence robot can rotate 45°, 90°, 135° or 180 °. We have used 4 different maps for comparison. In each map the starting node will be the top left cell and the target node will be the bottom right cell. To measure the total angle turned, we have considered that in the beginning the robot faces towards north.

Number of explored nodes is the count of all nodes to which weight and cost has been assigned. We have considered each cell to be of 1 unit length. For every horizontal or vertical movement, Path length will be incremented by 1 and for each diagonal move path length will be incremented by $\sqrt{2}$. We will compare the performance of MWF, FWF and OFWF based on 6 constraints i.e. Number of Nodes Explored, Number of Steps, Path Length, Execution Time, Number of turns and total angle turned (Fig. 1, Tables 1, 2, 3 and 4).

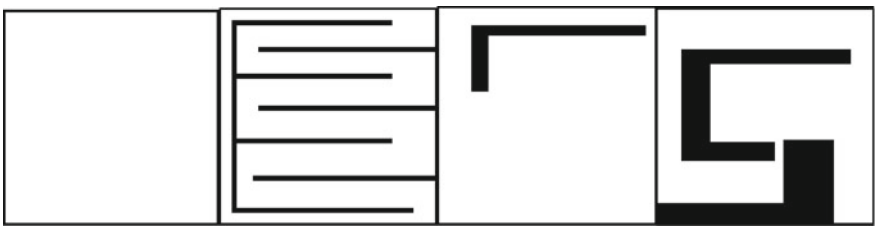

Fig. 1 Simulation environment

Table 1 Results for Map1 (200 × 200 pixels)

Parameters	MWF	FWF	OFWF
Nodes explored	40,000	989	994
Number of steps	199	199	199
Time (ms)	3895	49	53
Path length	281.428	281.428	281.428
Number of turns	1	1	1
Total angle turned (°)	135	135	135

Table 2 Results for Map2 (40 × 40 pixels)

Parameters	MWF	FWF	OFWF
Nodes explored	1344	938	405
Number of steps	75	197	75
Time (ms)	120	73	48
Path length	76.2426	213.154	76.2426
Number of turns	4	27	7
Total angle turned (°)	270	1530	450

Table 3 Results for Map3 (200 × 200 pixels)

Parameters	MWF	FWF	OFWF
Nodes explored	37,548	2389	11,820
Number of steps	247	247	247
Time (ms)	3550	173	1095
Path length	309.546	322.801	309.546
Number of turns	4	8	39
Total angle turned (°)	270	540	1890

Table 4 Results for Map4 (200 × 200 pixels)

Parameters	MWF	FWF	OFWF
Nodes explored	28,389	8625	14,153
Number of steps	342	373	342
Time (ms)	2724	807	1346
Path length	365.196	418.978	365.196
Number of turns	4	15	11
Total angle turned (°)	270	810	540

5 Conclusion

In uncluttered environment such as map1, all algorithms returned optimal results but MWF took considerably longer than FWF and OFWF. On observing the number of turns and Total angle turned for Map2, Map3 and Map4, we can deduce that MWF, FWF and OFWF propose 3 different paths for each case. Based on the observations, we can approximately compare the performance of OFWF with MWF and FWF. When we compare OFWF with MWF, we observe that path length and number of steps is same as both return optimal path length. Execution time and nodes explored is quite less for OFWF and number of turns and angle turned is better for MWF. When we compare OFWF with FWF, we observe that FWF execution time and number of nodes explored is better while path length and number of steps is better for OFWF.

Among the three, MWF provides the most optimal results in terms of path length, number of turns and total angle turned. However, it is computationally expensive and time consuming as it explores a relatively larger number of nodes. FWF is faster than MWF and OFWF but it compromises optimality for high speed. OFWF provides results with optimal path length and is a lot faster than the MWF. OFWF seems as a balanced algorithm which provides optimal path length with good execution time. If rotation cost is not a major concern, then OFWF may prove to be a good alternative among other path planning algorithms.

6 Future Work

In the future, Optimally Focused Wave front algorithm may be further modified so that apart from path length it may also optimize number of turns and total angle turned. The current algorithm may also be extended to work in 16 directions which will further optimize path length. The current algorithm might also be modified to work in unknown environment or with dynamic obstacles.

References

1. Nooraliei, A., Nooraliei, H.: Path planning using wave front's improvement methods. In: International Conference on Computer Technology and Development, ICCTD'09, IEEE, vol. 1, pp. 259–264 (2009)
2. Pal, A., Tiwari, R., Shukla, A.: A focused wave front algorithm for mobile robot path planning. In: Hybrid Artificial Intelligent Systems, pp. 190–197. Springer, Heidelberg (2011)
3. Liu, G., et al.: The ant algorithm for solving robot path planning problem. In: Third International Conference on Information Technology and Applications (ICITA), pp. 25–27 (2005)
4. Ganeshmurthy, M.S., Suresh, G.R.: Path planning algorithm for autonomous mobile robot in dynamic environment. In: 3rd International Conference on Signal Processing, Communication and Networking (ICSCN), IEEE, pp. 1–6 (2015)
5. Oh, J.S., Choi, Y.H., Park, J.B., Zheng, Y.F.: Complete coverage navigation of cleaning robots using triangular-cell-based map. IEEE Trans. Ind. Electron. **51**(3), 718–726 (2004)
6. Zelek, J.S.: Dynamic path planning. In: IEEE International Conference on Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century, vol. 2, pp. 1285–1290 (1995)
7. Biggs, G., et al.: All the robots merely players: history of player and stage software. IEEE Robot. Autom. Mag. **20**(3), 82–90 (2013)
8. Player/Stage Source Forge Homepage, http://playerstage.sourceforge.net10
9. Guo, X.: Coverage rolling path planning of unknown environments with dynamic heuristic searching. In: 2009 WRI World Congress on Computer Science and Information Engineering, IEEE, vol. 5, pp. 261–265 (2009)
10. Manikas, W., Ashenayi, K., Wainwright, R.: Genetic algorithms for autonomous robot navigation. IEEE Instrum. Meas. Mag. **10**(6), 26–31 (2007)
11. Du, X., Chen, H.-H., Gu, W.-K.: Neural network and genetic algorithm based global path planning in a static environment. J. Zhejiang Univ. Sci. **6**, 549–554 (2005)
12. Behnke, S.: Local multiresolution path planning. Preliminary version. In: Proceedings of 7th RoboCup International Symposium, Padua, Italy, pp. 332–343 (2003)
13. Diagonal Distance, http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html#diagonal-distance