Educational Administration: Theory and Practice
2023, 29(4), 1497-1508
ISSN: 2148-2403
https://kuey.net/

Research Article

Multi-Point Path Planning Using Bidirectional Path Search In Static And Dynamic Environments

Narayan Kumar[1*], Amit Kumar[2]

[1]Research Scholar, National Institute of Technology, Patna; Assistant Professor, Government Engineering College Vaishali.
[2] Professor, Department of Mechanical Engineering, National Institute of Technology, Patna.

Citation: Narayan Kumar, et al (2023), Multi-Point Path Planning Using Bidirectional Path Search In Static And Dynamic Environments, *Educational Administration: Theory and Practice, 29(4),* 1497-1508
Doi: 10.53555/kuey.v29i4.6468

ARTICLE INFO

ABSTRACT

A multi-point robot path planning system has been presented to improve path length, safety, and smoothness. Path representation techniques that begin coding from the start point and progress one grid at a time to the destination point are presented. The purpose of multi-goal path planning is to establish a collision-free path between a set of goal poses while minimising travelled distance, maximising path safety, and maximising path smoothness. Our algorithm guarantees that planned trajectories remain safe distances from barriers, lowering the likelihood of collisions. Dijkstra's method is one of the most basic shortest-finding algorithms. A star (A*) method is a variant of Dijkstra's shortest path first algorithm that is often utilised in heuristic-based games. The idea behind A* is to assign weight to each open node and then use a heuristic to compute the total cost from beginning to end. In robotics, A* finds the most optimal path using heuristics and cost functions. Bidirectional A* reduces computation by determining the shortest path from both the source and the destination. Bidirectional search is a search approach that uses both starting and target locations to find an initial solution. Smoothing increases path robustness.

I. Introduction

In recent years, there has been significant progress in the field of industrial robotics, although mobile intelligent robots have primarily been limited to research labs. To be deployed in real-world applications, an autonomous vehicle must be capable of self-navigating and making intelligent decisions. Vehicle path planning entails not only creating collision-free routes from a given place to a target point, but also determining the best path that minimises or maximises specific essential objectives. The majority of research on path planning has focused on determining a viable shortest path by creating a single-objective optimisation problem. However, choosing the best path is never a one-objective problem because many other characteristics, such as path safety (or vulnerability) and path smoothness, are also desirable for a navigation vehicle. The path vulnerability objective is related to how closely the vehicle travels past an impediment. To achieve this goal, we used Dijkstra's algorithms and the A* algorithm. Path smoothness is proportional to the number of turns a vehicle must make along the path from beginning to finish of its motion. Aside from the single vs many competing objectives connected with the path planning problem, a number of other concerns must be addressed during the optimisation process. The scheme used to represent a path within the optimisation method is a crucial consideration, as the efficiency [1] of an optimisation algorithm is heavily influenced by the chosen representation scheme.
The A* algorithm is one of the most common pathfinding methods. A* is one of the most important algorithms for various pathfinding methods. A* enables us to perform pathfinding by determining the location of the source and destination nodes and calculating the heuristic function. Then the node with the highest value of the heuristic function known as the f-cost is chosen, and its neighbours are extended. We detect each node's neighbouring nodes and put them in an open array. Once the f-cost heuristic has been calculated for all of these nodes, we identify the node with the lowest value, which is the node closest to the target. This node is now stored in a separate list so that we can return to it while determining the necessary path. The process described above is repeated for each selected node until we reach the destination node's neighbouring node. Once such a node is discovered, the algorithm stops computing the heuristic function and returns to all nodes on the path, resulting in the shortest path from the source to the destination. The proposed system just implements

Dijkstra's method through parallel processing, and the Bidirectional A* Search is more faster and more efficient in a range of cases than the simple Dijkstra's method [2]. The main difference between the proposed and existing systems is that in particular cases, the suggested Bi-directional A* search may not be enough due to unnecessary calculations.
The fundamental purpose of Multi-Goal Path Planning is to find a collision-free path. The problem of robot path planning has been studied for several decades, with several significant contributions [3]. Point-to-Point (PTP) Path planning techniques, which can discover a collision-free path from one start configuration (point) to a destination configuration (point), are highly theoretically interesting but rarely employed in practice due to their computational difficulty [4]. The Multi-Goal Path Planning (MTP) problem, which computes both a collision-free path and the optimal sequence, has not yet been addressed. In our opinion, resolving the MTP issue can improve off-line created programmes [5] and thus reduce overall programming time.
This study demonstrates an effective form of the bidirectional search strategy. It uses the Dijkstra algorithm and the A* evaluation function to boost the method's information, which accelerates convergence during the search and improves path smoothness. A security camera-assisted obstacle identifier approach might be used to the planner surface. The surveillance system's function is to monitor and detect fixed and moving impediments. In this study, we analyse the proposed algorithm technique's effectiveness in terms of calculations, memory utility, and area coverage. The remaining sections of the paper are organised as follows: Section II provides a review of the literature on previous works in this topic. Section III discusses bidirectional path search algorithms in both static and dynamic situations. The next section, IV, discusses the multi-point path planning method employed in this study, and section V examines experimental outcomes. The conclusion and future scope are covered in Section VI.

II. Related Work

Pathfinding algorithms can be used to do a variety of jobs with ease. Path-finding algorithms, on the other hand, are widely employed in the gaming industry to discover the shortest path between two points. Currently, independent game creators employ serial implementations of pathfinding algorithms. Researchers made significant advances in modern pathfinding by implementing Dijkstra's algorithm in the GPU, which reduced the time required to discover the shortest route between the source and the destination. The proposed system just implements Dijkstra's method through parallel processing, and the Bidirectional A* Search is more faster and more efficient in a range of cases than the simple Dijkstra's method. Dijkstra searches for all feasible nodes from the source to the destination, showing a path if one is available. Bidirectional A*, on the other hand, begins at the source and ends at the destination, and it can be divided into smaller threads that can be executed simultaneously from both sides to discover the shortest path. Many conventional approaches for two-dimensional path planning have been developed using classical optimisation methods, including the artificial potential field method [6], visibility graph [7], Voronoi road map [8], and so on. Many artificial intelligence techniques [9], such as neural networks [10], fuzzy logic methods [11], genetic algorithms [12], and ant colony optimisation [13], have lately gained traction in path planning.
The proposed classification considers the behaviour of path-planning algorithms. Many prior studies divided between two types of path planners: online and offline, based on whether the environment is dynamic or static. A mobile robot's path planning is used to create a collision-free path from a starting point to a destination while optimising a performance criterion such as distance, time, or energy, with distance being the most commonly utilised criterion. Global path planning refers to offline route planning for robots in situations where extensive knowledge about stationary impediments and the trajectory of moving items is known ahead of time. When complete information about the surroundings is not available ahead of time, the mobile robot collects data as it moves through the environment utilising sensors. This is known as online or local path planning [14]. Essentially, online path planning begins its original path offline and then switches to online mode when it detects new changes in obstacle conditions.

Figure 1: The principle of mobile robot global/local path planning [15]

Because of its high computing speed, a non-replanning approach might be used online. The opposite may also occur. For example, the Dynamic Window technique (DWA) is a Reactive Computing technique that is widely used for local planning [15] but may also be used for global planning. A novel evaluation function is developed

by using the result of global path planning as a reference trajectory to ensure the optimal trajectory [16]. Furthermore, the study suggests three assessment sub-functions: direction, smoothing speed, and acceleration, which ensure the direction, smoothness, and speed of movement, respectively. Figure 1 illustrates the principle of mobile robot global/local route planning, while Figure 2 depicts the broad classification of mobile path planning.

Autonomous surface vehicles are gaining popularity worldwide due to the promise for greater safety and efficiency. This has generated interest in developing path-planning techniques that can lessen the likelihood of collisions, groundings, and strandings at sea, as well as associated costs and time spent. There is an interesting contrast between algorithms that require a preliminary map representation (Classic) and those that do not (Advanced). Classic includes Graph Search methods, whereas Advanced focuses on Soft Computing and Sampling-Based algorithms. Figure 3 depicts the distinction between path planning terminologies in the literature [17]. Path planning classifications are clear and reasonable [18]: according to the robot model (holonomic, non-holonomic, kinodynamic); according to the map model requirement (needed or not beforehand); according to the replanning capability (offline or online); and according to whether the algorithm always calculates the same solution based on preliminary configuration parameters (deterministic or probabilistic).

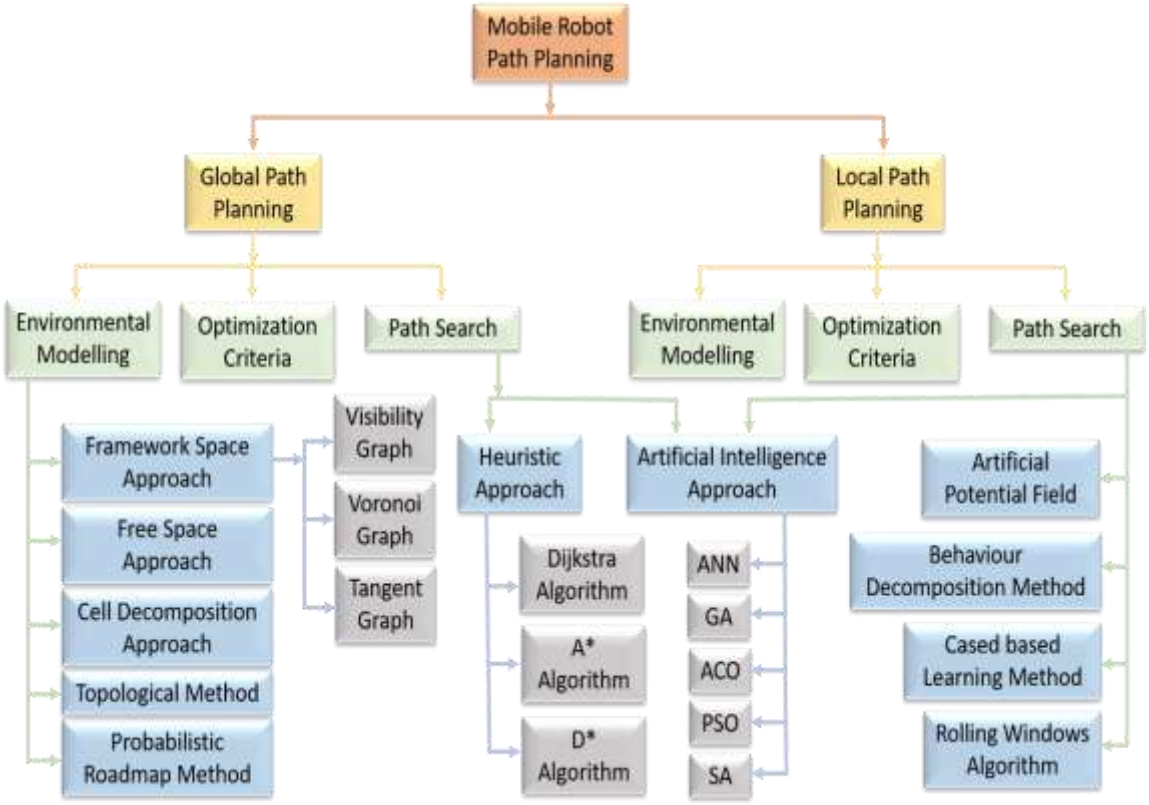


Figure 2: Classification of path planning [15]

Robot autonomous navigation is an appealing study topic due to its vast range of applications. Navigation requires four major components: perception, location, cognition, path planning, and motion control, with path planning being the most important and exciting. The classical methods are cell decomposition, potential field method, subgoal network, and road map. The techniques are straightforward, but they typically need expensive computation and may fail if the robot faces uncertainty. In contrast, heuristic-based robot path planning algorithms include neural networks, fuzzy logic, nature-inspired algorithms, and hybrid algorithms. The present mobile robot research focuses on path-planning algorithms and optimisation in both static and dynamic environments. Mobile robot path-planning tactics can be classified into two types: classical approaches and heuristic methods. There are four subcategories: analytical approaches, enumerative methods, evolutionary methods, and meta-heuristic methods [19].

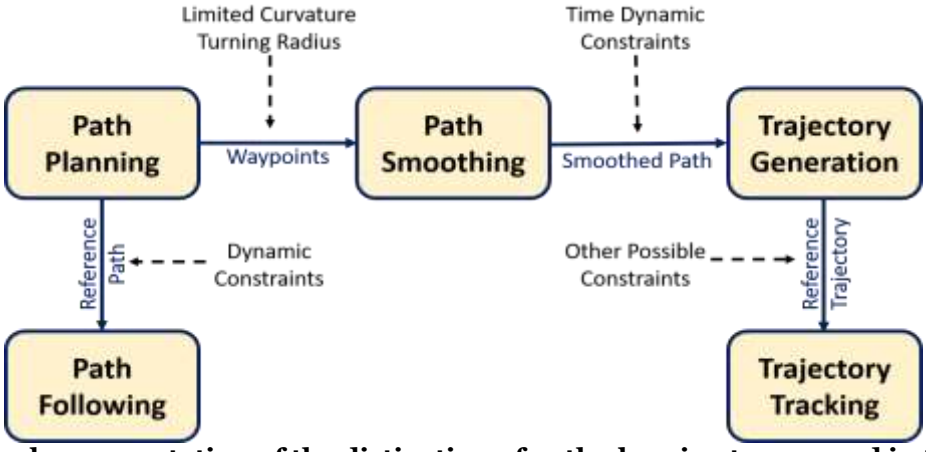


Figure 3: Visual representation of the distinction of path planning terms used in the literature [17]

Each of the above tactics has advantages and cons. However, the main difficulty is that analytical procedures are too complex for intangible applications, whereas enumerative methods are concerned with the size of the search space. When a path-planning technique's search space is too large, many evolutionary techniques fail. To address these drawbacks, meta-heuristic methods have inspired a lot of interest in this broad field of study. Path planning for mobile robots has resulted in the development of several systems worldwide. The navigation of static and dynamic circumstances is explored (for single and multiple robot systems), and reactive approaches are found to be more robust and perform well in all terrains than classical approaches. Reactive techniques, such as hybrid algorithms, have also been seen to improve the performance of classical approaches. As a result, reactive strategies for mobile robot path planning are gaining popularity and widespread use [20]. The term "heuristic" has been used to describe evolutionary and artificial intelligence systems, as well as Graph Search-based planners. Table 1 displays the nature-inspired methods developed in chronological sequence.

Table 1: Nature-inspired methods in chronological order [21]

Year	Nature inspired methods	Year	Nature inspired methods
1965	Evolution Strategies	2005	Honey Bee Algorithm
1966	Evolutionary Programming	2005	Harmony Search Algorithm
1975	Genetic Algorithms	2007	Intelligent Water Drop
1979	Cultural Algorithm	2007	Firefly Algorithm
1983	Simulated Annealing	2009	Gravitational Search Algorithm
1989	Tabu Search	2009	Cuckoo Search Algorithm
1992	Ant Colony Optimization	2010	Artificial Bee Algorithm
1995	Particle Swarm Optimization	2010	Bath Algorithm
2002	Estimation of Distribution Algorithm	2012	Keill Herd
2002	Bacterial Foraging Algorithm		

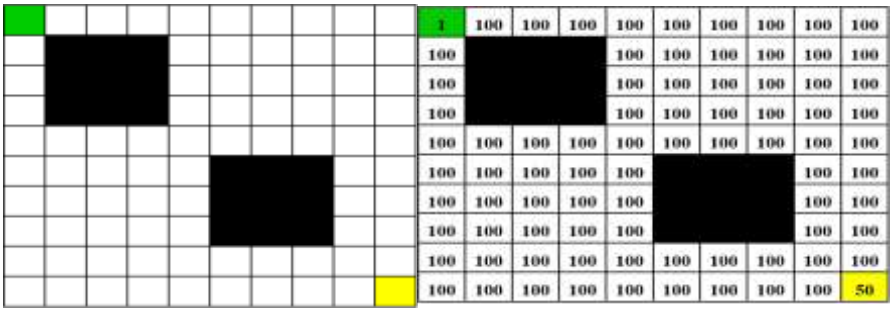
The smoothness of a path is determined by adding the angles of each turn the vehicle must make while traversing the path. The smoothness of a trajectory is a desirable feature in vehicle path planning [22]. Smooth pathways reduce unnecessary curvature discontinuities and possible stops, lowering the likelihood of slippage. It reduces power usage and travel time. Though smoothness is desirable in a path, it should not be utilised as an objective function in most path planning optimisation assignments, because the truly optimal smooth path, which ignores path vulnerability and other manoeuvring difficulties, may not be what a path planner is seeking. In this work, we look at smoothness as a secondary goal of the path-planning assignment. Path smoothness can be utilised as a tie-breaker or decision-making aid in a bi-objective optimisation framework that aims to minimise both path length and path vulnerability.

III. Bidirectional path search

The primary principle behind bidirectional search is that two distinct searches are initiated from both the start and the goal, i.e., search forward from the initial state and backward from the goal state until the two search frontiers intersect. The path from the initial state is then concatenated with the inverse path from the destination state to create the complete solution path. In principle, these planners can minimise the number of expanded states dramatically, making them a promising family of algorithms. It is well understood that informative heuristics considerably improve search efficiency for unidirectional searches, thus employing them for both searches in a bidirectional algorithm is a natural extension.

A. Static environment

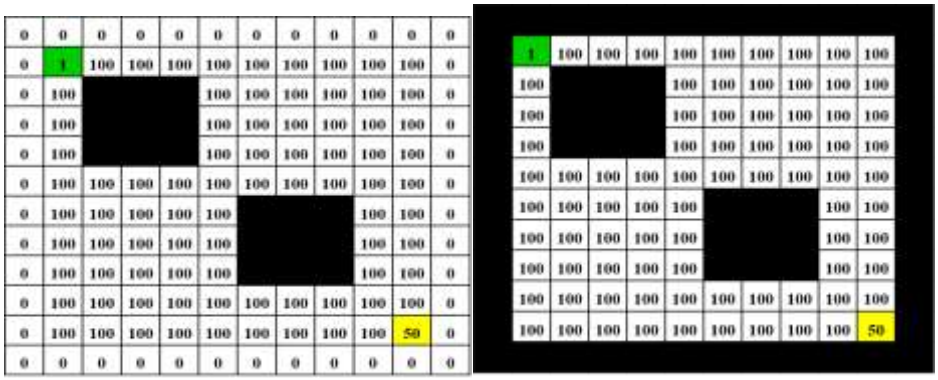
For path planning in a static environment, first of all, a map is called in which there are a start point, endpoint, and obstacles. Initially, the path planning objects have a minimum distance avoiding the obstacles in the path and then added a minimum number of turns with minimum distance avoiding the obstacles. The flowchart in Figure 8 shows the algorithm that works in a way to achieve this. The map is seen in terms of the matrix for the problem simplification.



(a) (b)

Figure 4: Static path planning problem: (a) start position in green, goal position in yellow, black cells are obstacles and white cells are free spaces, and (b) '1' is the start point, '50' is the goal point, '0' are obstacles, and '100' are free space.

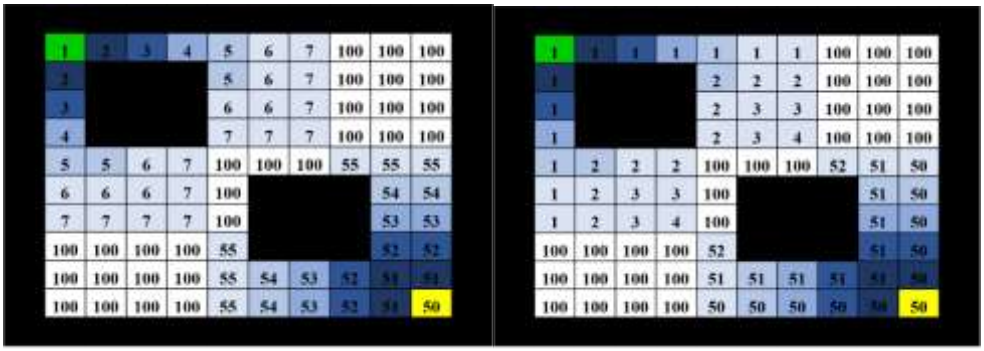
This is because the map is first surrounded by zero patches, the obstacles inside the maps are also numbered as zero. The initial cell is numbered 1, the end position is numbered 50, and the rest of the cells are numbered 100. If the given map is of dimension 10*10 then the patches of zeros surround it means we add zero rows and zero columns at the start and end of row and column respectively. After adding zero patches to surround the map the dimension of the matrix becomes 12*12. Figure 5 shows the initial map given and the modified matrix to solve the problem in MATLAB.



(a) (b)

Figure 5: Modified static path planning problem: (a) surrounding the workable areas with a patch of 0 and (b) making the surrounding patch obstacles.

Two types of matrices have been formulated; the first matrix gives information about the adjacent cell and the second matrix gives information about diagonal or non-diagonal cells. As the algorithm moves to the next step, the first matrix identifies the adjacent cells and the condition of break. In dealing with up-front waves the adjacent cell numbers should be greater than 49 (49 not included) and less than 100 (100 not included), then only the adjacent cells will be numbered; otherwise, the condition of brake met and numbering stops. In dealing with a down-front wave the adjacent cell numbers should be greater than 0 (0 not included) and less than 50 (50 not included), then only the adjacent cells will be numbered; otherwise the condition of break met and numbering stops.



(a) (b)

Figure 6: Matrix formulation (a) numbering for adjacent cells in blue (b) numbering for identification of diagonal or non-diagonal cells in blue.

The other matrix is for diagonal or non-diagonal identification. In the upfront forwarding direction if it goes to a non-diagonal direction then the number will not be changed and if it goes in a diagonal direction then the number will change by unity w.r.t. existing number. The same is the condition down front in the backward direction shown in Figure 7.

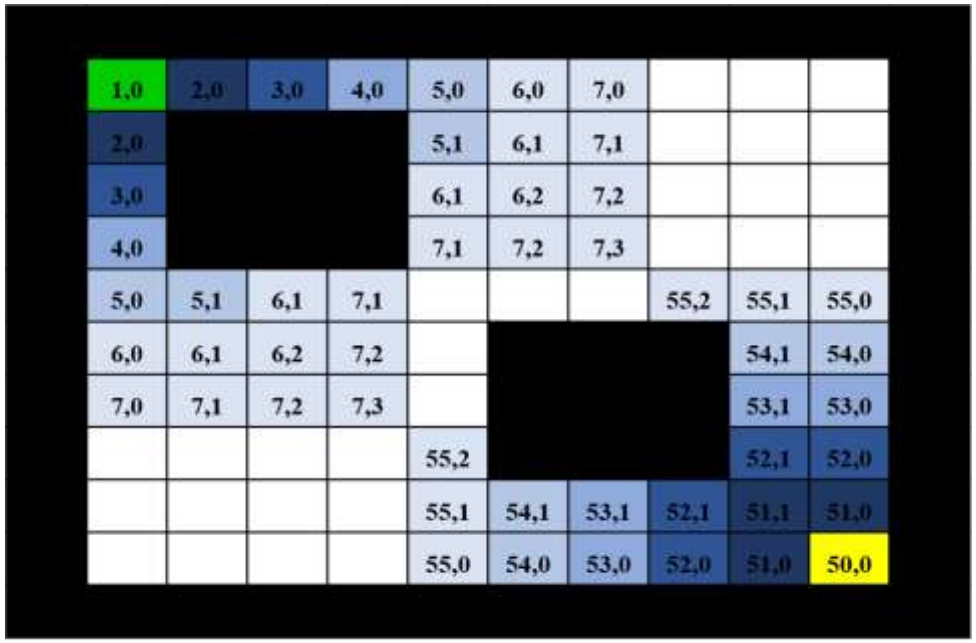


Figure 7: Combining the matrix of Figure 6 after some modification.

The combined matrix of Figure 6 is shown in Figure 7. For the formulation of this matrix, the first element of cell number indicates the matrix of Figure 6 (a) and the second element of cell number indicates the matrix of Figure 6 (b) with some modification. The cells which are un-calculated are left to be 100 and this is erased from the combined matrix. the modification for the second number is as '1' is subtracted from the upfront wave and '50' is subtracted from the downfront wave.

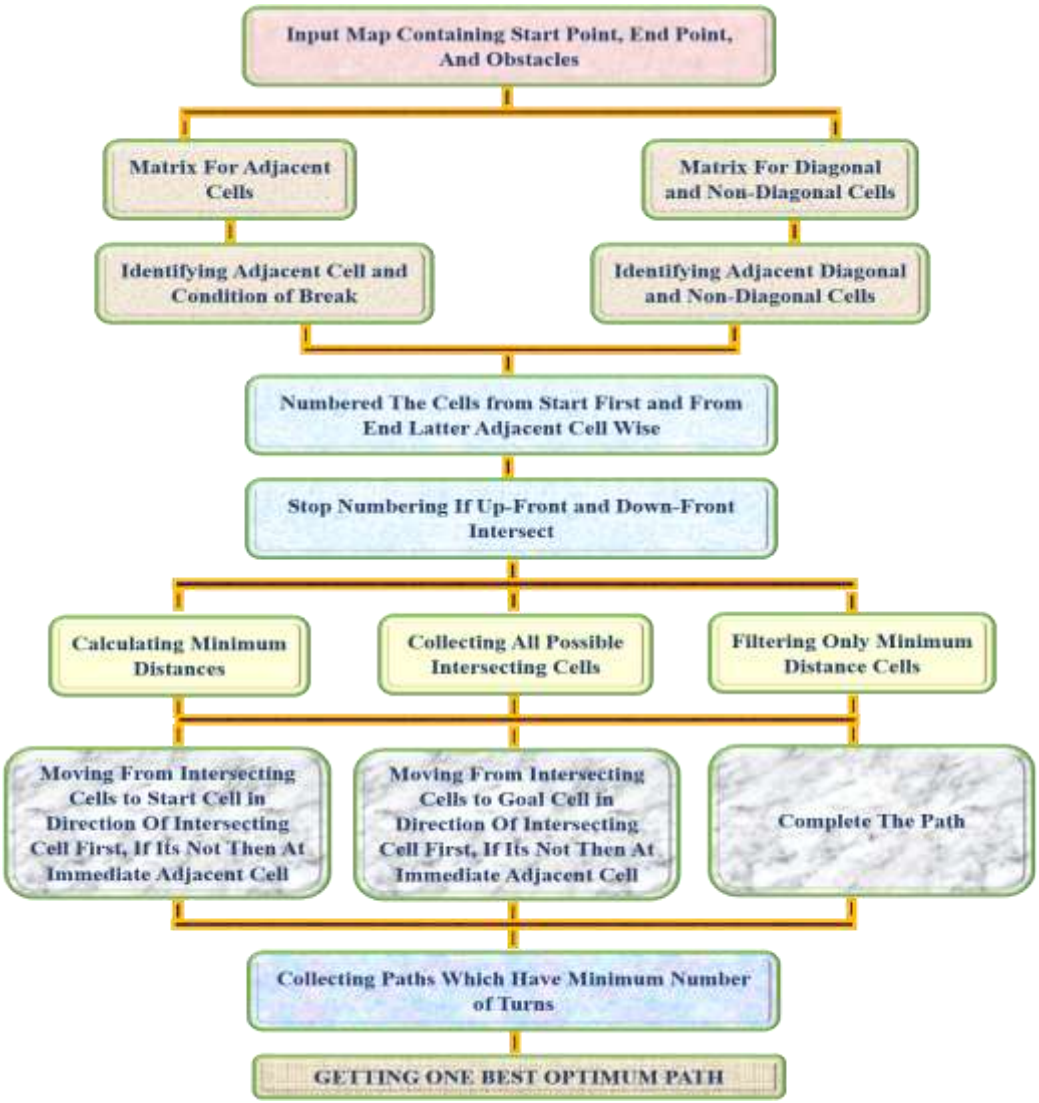


Figure 8: Flow chart of static environment algorithm in MATLAB.

As the wavefronts intersect, the possible ways are identified through this intersection. Calculates the distances of all possible paths through the intersections. We filter the ways of intersection which gives the path of minimum distance. The intersecting cells of minimum distance are collected the moved one by one to the next cells to complete the path. From the open upfront cell, move in a backward direction in the same direction of intersection if possible otherwise change the direction for the next cell. This way complete the path in an upfront wave. Then from the intersecting open downfront cell, move to the next forward cell in the direction of the intersecting cell if possible otherwise change the direction for the next cell and complete the path this way in the downfront wave. Combining the path of the upfront wave and downfront wave gives the complete path from the start cell to the goal cell. Here we filter the path that has a minimum number of turns and this path is assumed to be the best path according to this algorithm for a static environment.

B. Dynamic environment

For path planning in a dynamic environment, first of all, a map is called in which there are a start point, endpoint, static obstacles, dynamic obstacles, and free spaces. Initially, the path planning objects to have a minimum distance avoiding the obstacles in the path and then added a minimum number of turns with minimum distance avoiding the obstacles for each no. of steps. The flowchart in Figure 9; shows the algorithm that works in a way to achieve this.

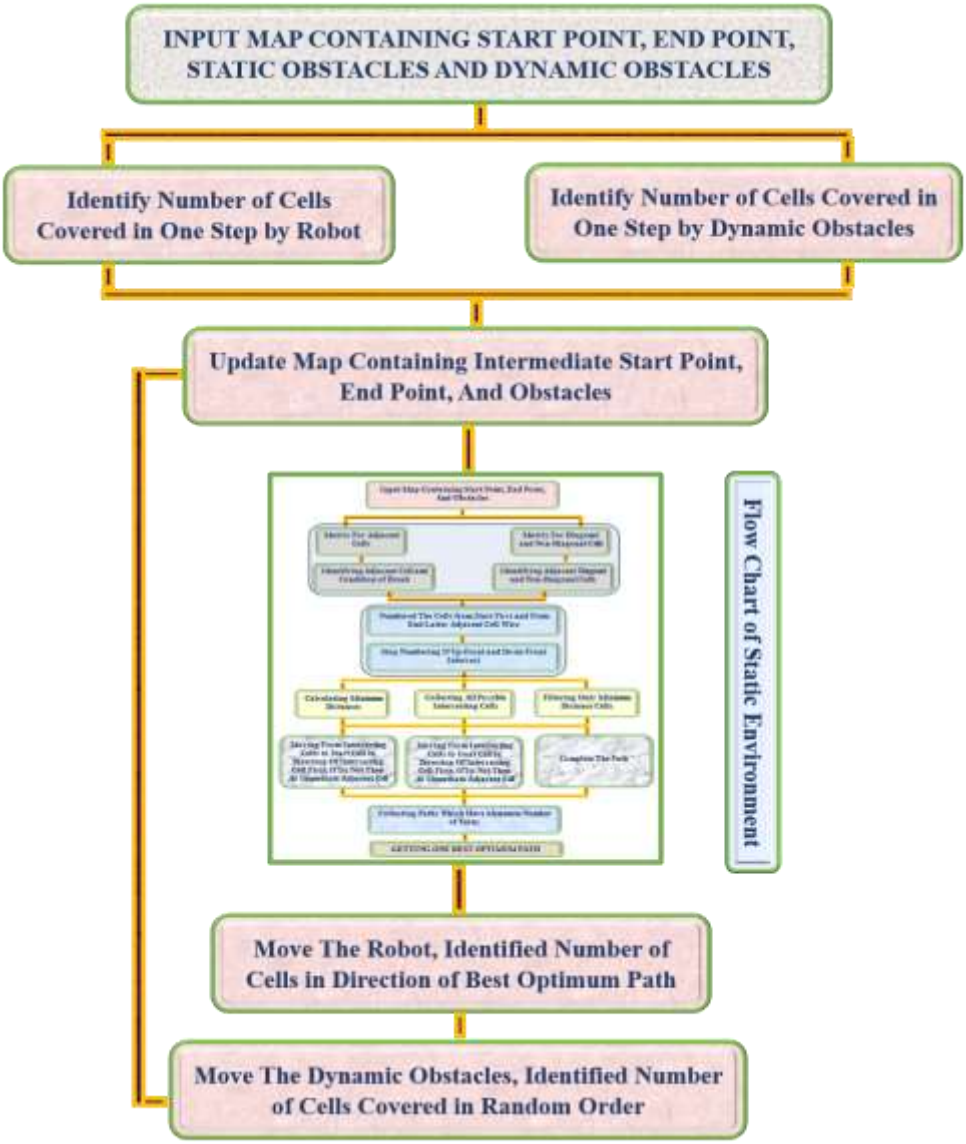


Figure 9: Flow chart of dynamic environment algorithm in MATLAB.

In a dynamic environment after calling the map, first of all, we identify the number of steps taken by the robot as well as the moving obstacles. The combination may be like

(i) Balance robot: The number of forward steps taken by the robot is equal to the number of forward steps taken by moving obstacles in an iteration. e.g. for the five forwarded steps taken by the robot, the moving obstacles also move five steps in an iteration.

(ii) Faster robot: The number of forward steps taken by the robot is greater than the number of forward steps taken by moving obstacles in an iteration. e.g. for the five forwarded steps taken by the robot, the moving obstacles also move two steps in an iteration.

(iii) Slower robot: The number of forward steps taken by the robot is lesser than the number of forward steps taken by moving obstacles in an iteration. e.g. for the two forwarded steps taken by the robot, the moving obstacles also move five steps in an iteration.

Here in the problem formulation, the moving obstacles can be seen in two ways. The first is that surround the moving obstacles by a cell and count it as an obstacle too for the iteration. This is done so that the chances of obstruction of the robot to moving obstacles are reduced to minimal. The second is that assume the obstacles exactly of cell dimensions and design the path planning. This will give a more accurate path but the chances of the obstruction are there a little bit.

After deciding the number of steps to be taken by the robot and moving obstacles the algorithm of path planning of a static environment is applied. This will give the path for this iteration. Move the robot on the path in the desired number of steps and then move the moving obstacle in the desired number of steps in random order. The next iteration will start from the previous robot's position as a starting point, updated obstacle positions, goal positions, and updated free spaces. The iteration runs until the start position becomes equal to the goal position.

IV. Multi-Point path planning

For multi-point path planning, first of all, a map is called in which there are a start point, endpoint, static obstacles, dynamic obstacles, and free spaces. Initially, the path planning objects to have a minimum distance avoiding the obstacles in the path and then added a minimum number of turns with minimum distance avoiding the obstacles for each no. of steps. The flowchart in Figure 10; shows the algorithm that works in a way to achieve this.

According to requirements the multiple points may be selected through which the robot needs to go through. So numbering the map with multiple start and end points. Update the initial two intermediate points as start and goal points. Apply the algorithm of the dynamic environment to get the path for this intermediate start and goal points. Then for the next iteration next two selected cells are the start cell and the goal cell respectively. The iteration will go through until the start position reaches the final goal cell.

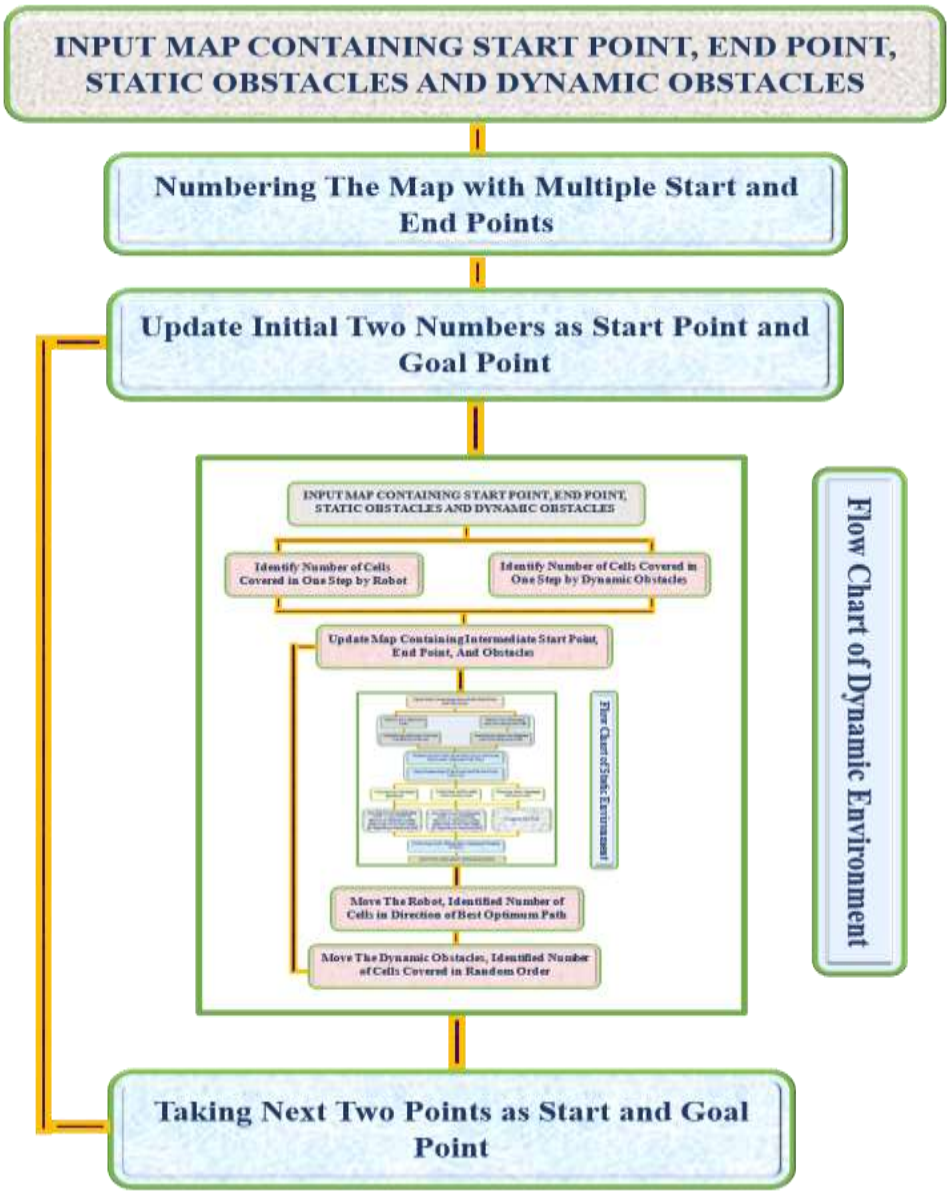


Figure 10: Flow chart of multipoint path-planning algorithm in MATLAB.

V. Experimental Result

The multi-point path planning problem depicted in Figure 11; can be modeled as a railway platform with pre-installed webcams for surveillance and obstacles in static and moving modes.

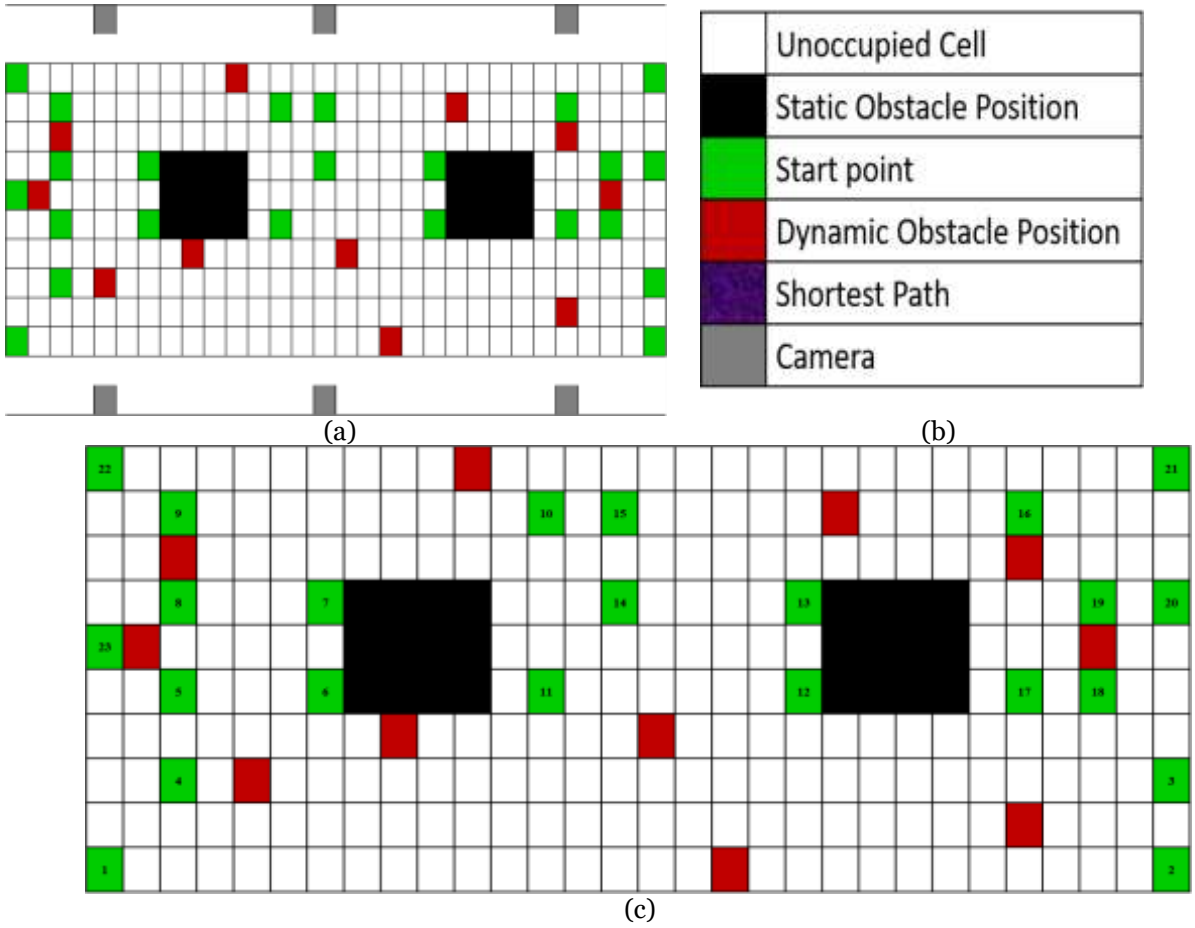


Figure 11: (a) Multi-point path planning problem for covering all nodes in turns (b) cell colour representation and (c) numbering of multi statrt and multi end cell grid.

The static obstacles may be treated as the beam support area and shops, and the moving obstacles as passengers. The service robot will move from its starting point to its destination, equipped with a cleaning and sanitization manipulator. Here in the multi-point path planning the size of the robot is assumed to be the size of the cell. The spraying area includes the adjacent cell of the current cell. The camera coordinates are transformed into service robot coordinates and perform the desired work.

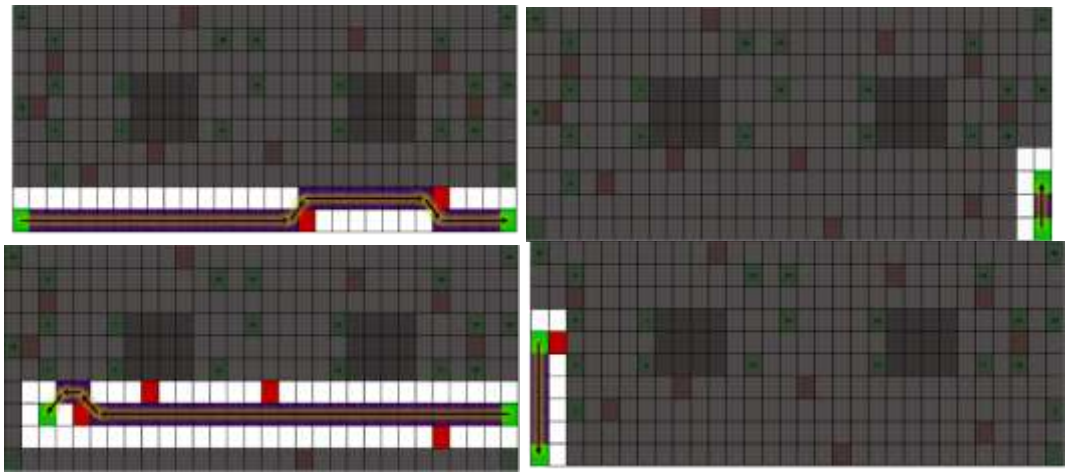


Figure 12: Path traced by robot in multi-point path planning.

The path tracing starts with the start cell number '1' as start cell and '2' as goal cell. As soon as the robot reached the cell number '2', it acts like start position and cell number '3' turns to be goal position. The goal position simultaneously converted to start position cell until it reaches to the very first start position cell that is the cell number '1'.

Table 2: Positional transformation robot with intermediate start and goal positions

No. of steps	Start Position		Goal Position		Distance	Operational Area	
(a)	(10,1)	1	(10,30)	2	31.82	9 to 10	1 to 30
(b)	(10,30)	2	(8,30)	3	2	7 to 18	29 to 30
(c)	(8,30)	3	(8,3)	4	29.82	7 to 9	2 to 30
(d)	(8,3)	4	(6,3)	5	2	5 to 9	2 to 4
(e)	(6,3)	5	(6,7)	6	4	5 to 7	2 to 8
(f)	(6,7)	6	(4,7)	7	2	3 to 7	6 to 8
(g)	(4,7)	7	(4,3)	8	4	3 to 5	2 to 8
(h)	(4,3)	8	(2,3)	9	4.82	1 to 5	2 to 4
(i)	(2,3)	9	(2,13)	10	10	1 to 3	2 to 14
(j)	(2,13)	10	(6,13)	11	4	1 to 7	12 to 14
(k)	(6,13)	11	(6,20)	12	7	5 to 7	12 to 21
(l)	(6,20)	12	(4,20)	13	2	3 to 7	19 to 21
(m)	(4,20)	13	(4,15)	14	5	3 to 5	14 to 21
(n)	(4,15)	14	(2,15)	15	2	1 to 5	14 to 16
(o)	(2,15)	15	(2,26)	16	13.82	1 to 3	14 to 27
(p)	(2,26)	16	(6,26)	17	6.82	1 to 7	25 to 27
(q)	(6,26)	17	(6,28)	18	2	5 to 7	29 to 27
(r)	(6,28)	18	(4,28)	19	4.82	3 to 7	27 to 29
(s)	(4,28)	19	(4,30)	20	2	3 to 5	27 to 29
(t)	(4,30)	20	(1,30)	21	3	1 to 5	29 to 30
(u)	(1,30)	21	(1,1)	22	31.82	1 to 2	1 to 30
(v)	(1,1)	22	(5,1)	23	4	1 to 6	1 to 2
(w)	(5,1)	23	(10,1)	1	5	4 to 10	1 to 2

The table represent the the distrance travelled in every steps. The total twenty three steps are needed to take for the complete path formation. In this case, the initial and final points are the same, and the service robot begins at point "1" sequentially covering all "16" nodes before returning to its initial position. The service robot is programmed in such a way that it will signal the moving man to move aside and wait for a while. If the obstacle does not move then the robot avoids the obstacle and completes the duty.

VI. Conclusion and Future Scope

In conclusion, there is potential for research into the use of mobile robot systems for common problems. Single or multiple robot mobile robotic systems can be employed for a wide range of applications under various operating environments. In light of the current focus on intelligent field robotics, the ability of heterogeneous mobile robot systems to function well in unfamiliar situations is crucial. Robot activities in an outside natural setting can be quite difficult, especially when compared to industrial settings that give a regulated setup, where the environment can be calibrated and the tasks may be monotonous. The field of mobile robot path planning has two branches: online and offline. As a result, many algorithms and methods are still being developed to offer the best answers for all the problems they encounter.

The up-front and down-front are generated concurrently, so pre-installed webcams are required. The path with the fewest turns allows the robot to retain its trip continuity. Robots should choose the path with the fewest turns. With less turning, the robot may reduce basic motor activity, saving time and energy. In path planning problems, the algorithm generates a huge number of paths that cover the shortest distance. This paper selects the optimal ideal path from those that have the same minimum distance between the start and goal places. The criterion for the answer is the fewest number of bot turns. This considerably advances the bidirectional search algorithm in complicated maps for mobile robots, particularly in terms of shortest path length and overall fewer turns.

In this study, we provide the novel Multi-Goal Path Planning (MTP) approach, which is the first powerful assistance tool for human programmers to address MTP challenges in offline programming assignments. In future study, the proposed approach could be integrated with various path-planning algorithms in a dynamic setting. Some possible applications include service robots, rescue robots, and industrial robots.

References

1. [1]. Ahmed, F. and Deb, K., 2013. Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms. Soft Computing, 17, pp.1283-1299.
2. [2]. Sharma, S., Srijan, S. and JV, V., 2020. Parallelizing bidirectional A* algorithm. In Intelligent Systems and Computer Technology (pp. 558-562). IOS Press.
3. [3]. Hwang, Y.K. and Ahuja, N., 1992. Gross motion planning—a survey. ACM Computing Surveys (CSUR), 24(3), pp.219-291.

4. [4]. Gupta, K. and Pobil, A.P., 1998. Practical motion planning in robotics: Current approaches and future directions. John Wiley & Sons, Inc..
5. [5]. Wurll, C.H.R.I.S.T.I.A.N., Henrich, D.O.M.I.N.I.K. and Wörn, H.E.I.N.Z., 1999. Multi-goal path planning for industrial robots.
6. [6]. Chen, Y.B., Luo, G.C., Mei, Y.S., Yu, J.Q. and Su, X.L., 2016. UAV path planning using artificial potential field method updated by optimal control theory. International Journal of Systems Science, 47(6), pp.1407-1420.
7. [7]. Lacasa, L., Luque, B., Ballesteros, F., Luque, J. and Nuno, J.C., 2008. From time series to complex networks: The visibility graph. Proceedings of the National Academy of Sciences, 105(13), pp.4972-4975.
8. [8]. Niu, H., Savvaris, A., Tsourdos, A. and Ji, Z., 2019. Voronoi-visibility roadmap-based path planning algorithm for unmanned surface vehicles. The Journal of Navigation, 72(4), pp.850-874.
9. [9]. Mellit, A. and Kalogirou, S.A., 2008. Artificial intelligence techniques for photovoltaic applications: A review. Progress in energy and combustion science, 34(5), pp.574-632.
10. [10]. Bishop, C.M., 1994. Neural networks and their applications. Review of scientific instruments, 65(6), pp.1803-1832.
11. [11]. Eker, I. and Torun, Y., 2006. Fuzzy logic control to be conventional method. Energy conversion and management, 47(4), pp.377-394.
12. [12]. Sivanandam, S.N., Deepa, S.N., Sivanandam, S.N. and Deepa, S.N., 2008. Genetic algorithm optimization problems. Introduction to genetic algorithms, pp.165-209.
13. [13]. Tuani, A.F., Keedwell, E. and Collett, M., 2018. H-ACO: A heterogeneous ant colony optimisation approach with application to the travelling salesman problem. In Artificial Evolution: 13th International Conference, Évolution Artificielle, EA 2017, Paris, France, October 25–27, 2017, Revised Selected Papers 13 (pp. 144-161). Springer International Publishing.
14. [14]. Raja, P. and Pugazhenthi, S., 2012. Optimal path planning of mobile robots: A review. International journal of physical sciences, 7(9), pp.1314-1320.
15. [15]. Zhang, H.Y., Lin, W.M. and Chen, A.X., 2018. Path planning for the mobile robot: A review. Symmetry, 10(10), p.450.
16. [16]. Zhang, F., Li, N., Xue, T., Zhu, Y., Yuan, R. and Fu, Y., 2019, December. An improved dynamic window approach integrated global path planning. In 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO) (pp. 2873-2878). IEEE.
17. [17]. Vagale, A., Oucheikh, R., Bye, R.T., Osen, O.L. and Fossen, T.I., 2021. Path planning and collision avoidance for autonomous surface vehicles I: a review. Journal of Marine Science and Technology, pp.1-15.
18. [18]. Souissi, O., Benatitallah, R., Duvivier, D., Artiba, A., Belanger, N. and Feyzeau, P., 2013, October. Path planning: A 2013 survey. In Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM) (pp. 1-8). IEEE.
19. [19]. Dijkstra, E.W., 2022. A note on two problems in connexion with graphs. In Edsger Wybe Dijkstra: His Life, Work, and Legacy (pp. 287-290).
20. [20]. Hart, P.E., Nilsson, N.J. and Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics, 4(2), pp.100-107.
21. [21]. Stentz, A., 1994, May. Optimal and efficient path planning for partially-known environments. In Proceedings of the 1994 IEEE international conference on robotics and automation (pp. 3310-3317). IEEE.
22. [22]. Nash, A., Koenig, S. and Tovey, C., 2010, July. Lazy Theta*: Any-angle path planning and path length analysis in 3D. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 24, No. 1, pp. 147-154).