Multi-Objective Optimal Path Planning for Autonomous Robots with Moving Obstacles Avoidance in Dynamic Environments

Kadari Neeraja[1], Dr. G Narsimha[2]
Research Scholar[1], Associate Professor[1], Professor[2]
Department of CSE, JNTUH, Hyderabad, India

***Abstract*—Path planning is vital for robust autonomous robot navigation. Driving in dynamic environments is particularly difficult. The majority of the work is based on the premise that a robot possesses a comprehensive and precise representation of its surroundings prior to its starting. The problem of partially knowing and dynamic environments has received little attention. This circumstance occurs when an exploratory robot or a robot without a floor plan or terrain map must move to its destination. Existing approaches for dynamic-path-planning design a preliminary path based-on known knowledge of the environment, then adjust locally by replanning the total path as obstacles are discovered by the robot's sensors, thereby sacrificing either optimality or computational efficacy. This paper presents a novel algorithm. A Near-Optimal Multi-Objective Path Planner (NO-MOPP), capable of planning time-efficient, near-optimal, and drivable paths in partially known and dynamic environments. It is an expansion of our earlier research contributions called "A Multi-Objective Hybrid Collision-free Optimal Path Finder (MOHC-OPF) for Autonomous Robots in known static environments" and "A Multi-Objective Hybrid Collision-free Near-Optimal Path Planner (MOHC-NODPP) for Autonomous Robots in Dynamic environments". In the environment, a mix of static and moving dynamic obstacles are present, both of which are expressed by a hybrid, discrete configuration space in an occupancy-grid map. The proposed approach is executed at two distinct levels. Using our earlier method, A Multi-Objective Collision-free Optimal Path Finder (MOHC-OPF), the initial optimal path is found in environment that includes only known stationery obstacles at the Global-path-planning level. On the second level, known as Local Re-planning, this optimal path is continuously refined by online re-planning to account for the movement of obstacles in the environment. The proposed method, A Near-Optimal Multi-Objective Path Planner (NO-MOPP), is used to keep the robot's sub-paths optimum while also avoiding dynamic obstacles. This is done while still obeying the robot's non-holonomic restrictions. The proposed technique is tested in simulation using a collection of standard maps. The simulation findings demonstrate the proposed method's ability to avoid static as well as dynamic obstacles, as well as its capacity to find a near-optimal-path to a goal location in environments that are constantly changing without collision. The optimal-path is determined by taking into account several performance measures, including path length, collision-free path, execution time, and smooth paths. 90% of studies utilizing the proposed method demonstrate that it is more effective than other methods for determining the shortest length and time-efficient smooth drivable paths. The proposed technique reduced average 15% path length and execution time compared to the existing methods.**

***Keywords*—Autonomous mobile robots; dynamic environment; planning; collision-free; time-efficient paths**

I. INTRODUCTION

Path planning for mobile robots, especially when the environment is known, is a well-researched problem [1-8]. However, one issue that arises when putting theory into practice is the fact that incomplete information about the environment is often available. In most cases, it seems unrealistic to expect to have a detailed map with all the obstacles clearly marked. Recent years have seen tremendous progress realized in the realm of path planning in dynamic environments across a wide range of domains. Particularly, mobile robots have found practical use across a wide range of domains. Applications incorporate emergency rescue management in natural disasters [12], planetary exploration [10, 11], inventory control [12], the manufacturing industry [13], etc.

In the 1960s, research began in the arena of path planning for different kinds of robots [1, 2, 12]. The Path-Planning is the procedure of establishing a path-way in an environment that is no-obstacles and that connects a predetermined starting point and an intended ending point [14-15]. The environments in which robots operate can either be static or dynamic. When working in a static known environment, the locations of obstacles remain the same, but when working in a dynamic environment, their positions shift over the course of time. The goal of employing path-planning algorithms is to translate the high-level-specifications which humans execute into low-level-steps [15]. This is accomplished by locating the optimal-path and presenting to the robot in the form of a series of waypoints that it should follow as moving directions.

The dynamic environment comprises moving obstacles, the path-planning algorithm is a necessity not-only to determine the optimal-path but also to observe it. In order to persist responsive to its environments, the approach must know the current position of an obstacle, forecast upcoming paths, and bring up-to-date its path in real-time with sufficient frequency.

In most cases, an autonomous mobile robot is free to follow any one of a number of predetermined routes. The length of the path, the amount of time it takes, and the amount of energy it

takes are all factors that go into determining what constitutes the optimal path. Numerous algorithms were developed to handle the problem of path planning; these algorithms can be categorized as either classical or intelligent. Artificial Potential Field [16], Rapid Exploration of Random Trees (RRT) [15, 17] and its variants RRT*, etc. [15,18], Partitioned Learning Traditional methods, such as D* [20], are utilized to solve dynamic path planning problems. As the search space grows in size, however, these methods become inefficient and get stuck at local maxima. Thus, intelligent optimization techniques like the Genetic Algorithm [21-22], Particle Swarm Optimization [23], Bees algorithm [24] [29] and etc. have been employed to solve path planning difficulties.

Real-time dynamic path planning needs more investigation [30], as stated earlier. This study proposes an A*[3] based Near-Optimal Multi-Objective Path Planner (NO-MOPP) to swiftly identify a near-optimal drivable smooth path in a dynamic environment while taking the kinematic restrictions of the robot into account. The following are some of the contributions made in this work:

1) "A Multi-Objective Hybrid Near-Optimal Dynamic Path Planner (NO-MOPP)" is a new dynamic path planning technique that finds no-collision near-optimal-drivable paths in a dynamic environment with a hybrid environment representation. Since A* ensures both optimality and completeness, the A* algorithm will serve as the foundation for this approach. This algorithm performs comparably to A*.

2) The proposed technique performs on two distinct levels. Initially, using global path planning, the optimal-path is determined in an area with known static-obstacles. The second level, known as Local-replanning, adjusts the optimal path online with the assistance of sensors in order to prevent collisions with dynamically generated immovable and moving obstacles. After that, path tracking is performed, and the path is optimized during path tracking without sacrificing the algorithm's real-time performance.

3) Kinematic constraints, like a robot's orientation, are employed in order to find the most efficient driving smooth paths in ever-changing real-time environments.

4) It finds application in a wide range of different dynamic environments. The percentage of successful attempts is 90%.

5) When compared to RRT and RRT*, our suggested method achieves superior outcomes in dynamic environments in relation to the amount of time required for execution, execution time, and the total length travelled path-length.

The paper is organized as follows. Section II explains related research that is pertinent to the techniques for planning paths. The technique and underlying algorithm for path-planning in the presence of static as well as dynamic obstacles are presented in Section III. Section IV looks at how well the suggested strategy works and gives the results of the experiments. Section V brings the article to a conclusion, which also offers guidelines for future work.

II. Related Work

Past decades have seen many path-planning algorithms. Graph-based techniques include Dijkstra's algorithm [4,24], A* [3], D* [20], and etc. After discretizing the path planning state space into a graph structure, they employ graph search to find a feasible path. A* and Dijkstra's algorithms are suitable for lower-dimension static environments. D* is used for dynamic environments. Optimal Path Planning using Memory Efficient A*. Improved A* Path Planning Method Based on the Grid Map. Sensors [25], Time-Efficient A* Algorithm for Robot Path Planning [26], Safe Path Planning of Mobile Robot Based on Improved A* Algorithm in Complex Terrains [27], Optimal Path Planning using Memory Efficient A*[31] and Fast path planning using modified A* method [32].

The graph-based approach is full and resolution optimal, meaning it finds an optimal-path if a viable path-exists and fails otherwise. The graph-based partition of the state-space yields a massive search space, which makes these graph-based approaches unsuitable for large-scale issues. The recent updates on A*, in research papers like Dynamic-Algorithm for Path-Planning using A* with Distance-Constraint [13] and Improved-Analytic-Expansions in Hybrid A* Path-Planning for Non-Holonomic Robots [9]. They are suffering from high computation time.

Another significant kind of path-planning algorithm is the sampling-based path-planning approach. Instead of discretizing the state space, it generates a graph or tree by randomly selecting points. Sampling-based path planning algorithms beat graph-based ones in large-scale situations. The sampling-based path planning strategy is probabilistically complete, thus when the trials number reaches infinite, the likelihood of discovering a suitable path-way approaches one. Sampling-based planners employ RRT [15,17] and PRM [19] algorithms. The RRT, a single-query path planning method that traverses state space by generating a tree rooted at the start state, is faster than the PRM. Despite finding an initial path in high-dimensional space quickly, RRT has many downsides. RRT's path may not be ideal because it is randomly generated. RRT* [18] advanced RRT. The RRT* takes time and memory to identify the best path. RRT* likewise experiences significant search time variability. Though, these techniques perform poorly and trap in local optima when the search space is big.

Hence, intelligent optimization procedures have been employed in the process of solving path-planning problems. Some examples of these algorithms include the Genetic Algorithm [21-22] [28], Particle Swarm Optimization [23], Simulated Annealing [12], Ant Colony Optimization [8], Bees [29] and etc. Even if these algorithms conquered the difficulties of path planning, they still wouldn't be usable without the partitioning and pre-processing of environment maps. This is because such maps need to be prepared in advance. The accuracy is reduced as a result of discretization and pre-processing, which also results in non-optimal pathways.

III. PROPOSED PATH PLANNING SYSTEM

The purpose of path-planning is to discover a continuous path that will lead a system from its current state to a desired one. Finding a path across a dynamic environment that a non-holonomic robot or vehicle can follow without colliding with any of the environment's obstacles is the goal of dynamic path planning.

1) Problem formulation: The path-planning issue's main objective is to identify an optimal-path for an autonomous-robot to proceed from a given beginning-point to a certain goal location in a dynamically changing environment containing static and dynamic stationery as well as moving obstacles by satisfying optimization-criteria. The path-planner's goal is to discover the optimal- or near-optimal-path for a mobile-robot that avoids obstacles in the surroundings.

The environment is denoted as a Grid. The initial step in mobile-robot-path-planning is establishing an environment-model for the mobile-robot's 2-dimensional. As identical square cells, grids are used to represent the mobile-robot's workplace. Each grid-cell is either free (logic 0) or forbidden (logic 1) by an obstacle. There are both static as well as dynamic stationary and moving obstacles in this area.

2) Optimization criteria: The proposed path planning system NO-MOPP determines a no-collision smooth path that obeys multi-objective optimization criteria. The criteria is: first one is the Cost objective-function , minimum-cost path for a mobile-robot to move from its start-point to the goal-point, provided that it is a smooth and safe path, i.e., the mobile-robot travels with no collision with obstacles. This measure is specified by Eq. (1) :

$$\text{Cost } f = g + h + SO + DSO + DMO \quad (1)$$

where cost-f is the sum-of-the-costs from the start-node to current-node (*g)*, the estimated-cost (*h)* to the goal from current-node, the additional-cost for changing the orientation angle, SO is the cost for switching orientation, DSO corresponds to the cost of avoiding dynamic stationary obstacles, and DMO indicates the cost of avoiding dynamic stationary obstacles during local replanning.

The second Criterion is the *Execution time objective* needed for finding a minimum length and safe path. This is given by equation (2):

$$T_{total} = T_{Global\text{-}path\text{-}planning} + T_{local\text{-}replanning} \quad (2)$$

where T_{total} is the time essential for the completion of execution of the path planner, $T_{Global\text{-}path\text{-}planning}$ is the time taken by the offline line path planner and $T_{Local\text{-}raplanning}$ is the time needed to update the initial optimal path to skip dynamic stationery obstacles.

The optimal path-planning problem can be stated as:

"Find the lowest cost and least time taking near-optimal smooth path between the start-point and the goal-location, such that the above optimization-criteria Cost function f and Execution time T objective functions given in above equations (1) & (2) are lessened by taking non-holonomic constraints of the robot into account".

3) The proposed dynamic path planning system's architecture: The component structure of the dynamic-path-planning system is depicted in Fig. 1. The system will function efficiently on the two levels. At the first level, global path planning, the optimal route is determined using the information that is currently known about the environment, including the known static obstacles, for instance. After that, the robot will continue along this optimal path. This optimal path is updated online at the second level, which is known as "Local Replanning," in order to skip collisions with-obstacles which are dynamically presented and may be either stationary or dynamic.

Primary components of the dynamic-path-planning system are depicted in Fig. 1. The environment is dynamic and comprises both static and moving obstacles. To represent this ever-changing environment, a binary occupancy grid map is employed. Constraints: Non-holonomic car-like robots or vehicles have kinematic constraints. Optimization criteria: Path smoothness, path length, and the time required to locate a path comprise optimization criteria.

4) Path-planning: The path-planning algorithm is the crucial component of a dynamic-path-planning system that addresses a path-planning issue. In this proposed system, there are two levels. A Multi-Objective Optimal Path Finder (MOHC-OPF) is used to obtain a quick initial optimal-path in an environment that includes known static obstacles only in the First level of Global Path Planning. Local-Replanning is the second level of our proposed dynamic path planning approach, a Near-optimal Multi-Objective Path Planner (NO-MOPP), which is employed to avoid dynamic obstacles in dynamically changing environments.

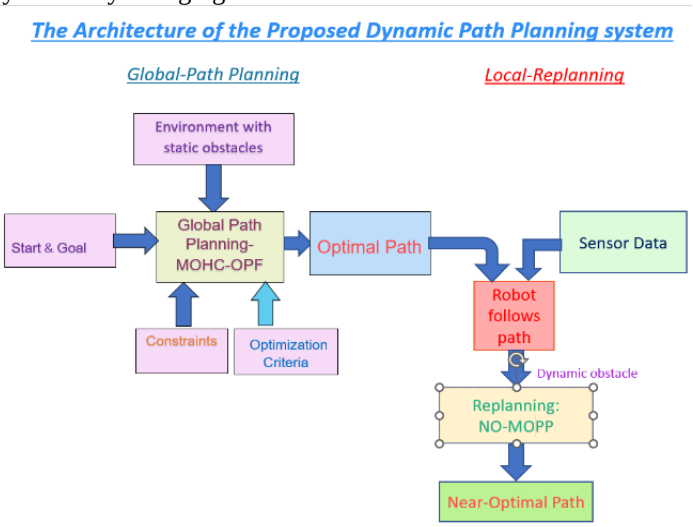

Fig. 1. Block diagram of the dynamic path planning system

A. The Working Principle of NO-MOPP

A novel method called NO-MOPP has been proposed as a way to avoid the limitations of the traditional A* methodology. The kinematics of the car-like robot or vehicle is added to predict the movement of the robot which is dependent on the steering angle in a continuous search space. The proposed

system has a collection of continuous states represented by the coordinates (x_p, y_p, θ), here (x_p, y_p) represents the location of the robot or vehicle and θ represents its orientation. Non-holonomic robots and vehicles can benefit from this feature since it helps the path planner choose the best successor state for them to follow. One of the five steering actions, maximum-left, left, maximum-right, right, and no-steering, expands the states and leads to an arc of a circle with a minimum turning radius, in accordance with the kinematic restrictions of the simple car-like robot or vehicle. On the basis of these operations, the proposed method, the NO-MOPP algorithm selects the states depicted in Fig. 2.

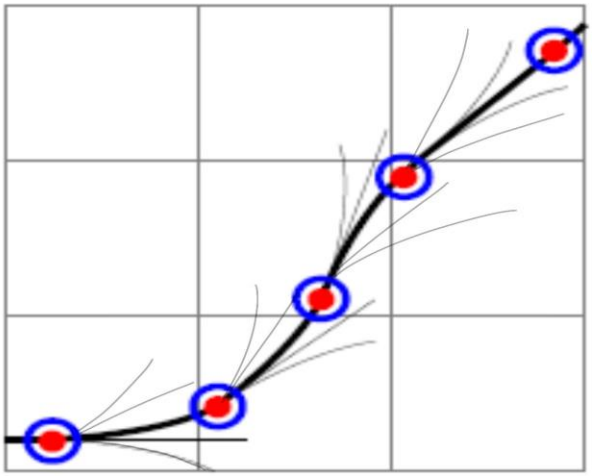

Fig. 2. NO-MOPP incorporates kinematic constraints with 5 steering angles

B. Multi-Objective Functions for Optimization

As part of this effort, different objectives are analysed and taken into consideration so that the updated path can be optimized.

Cost Function: It calculates the cost of driving from the present-point to a neighboring node. This cost f is the total-cost from the start node to the current node (g), the anticipated-cost from the present node to the goal-(h), and the cost for adjusting the orientation angle (SO). Eq. (1) is utilized to calculate this cost.

1) Path length: The final path is made up of a series of path segments denoted by the notation P= {P1, P2, …, Pn}. The ultimate length of the path is equal to the totality of the lengths of all path segments that connect the Start state to the Goal state via any intervening states. This is illustrated in Fig. 3. To get the length of this final path, Eq. (2) is utilized.

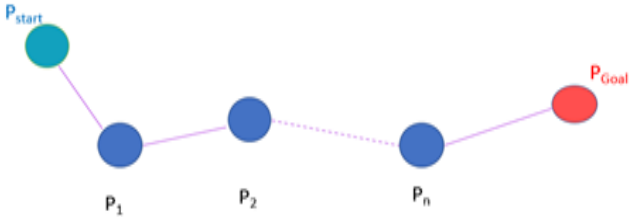

Fig. 3. Path length

$$\text{Path Length} = \sum Pi \text{ where } i = 1 \text{ to } n. \quad (3)$$

At each stage, the next state with the lowest possible cost is chosen, and the arcs in the path are optimized to have the minimum turning radius. As a result, it ensures that the final path will be the shortest and most optimal one.

2) Execution time: The amount of time required to carry out the method that was proposed for discovering a path and Eq. (2) is used for determining this.

The travelled distance must be sufficient in order to exit the current cell, and the equation that describes this requirement is below (4).

$$l > \sqrt{2}\, s \quad (4)$$

In this equation, l represents the length of an arc, and s indicates the size of a single cell in the grid map. At each node, the continuous state is rounded off to a discrete state in order to prevent the search graph from becoming an increasingly huge structure. This, in turn, results in a reduction in the amount of search time necessary to locate the ultimate path.

3) Smooth path: Kinematic constraints determine the next node in this proposed approach, resulting in minimum turning radius curves. Therefore, the final path produced is smooth.

C. The Heuristic Function

The heuristic function predicts the minimum-cost from any node to the destination on the map. This reduces node exploration. Thus, heuristic function selection directly impacts the performance of path planning approaches. The Euclidean distance is employed as a heuristic function. Equation (5) determines each node's heuristic values.

$$h = \sqrt{(x2 - x1)2 + (y2 - y1)2} \quad (5)$$

D. The Proposed Path Planning Approach

The proposed approach executes the path-planning procedure in a dynamic-environments with both dynamic as well as static obstacles. The method operates on two levels. Global Path Planning 2. Local Redevelopment Replanning. In the first level, the Global-path-planning method is applied to determine the optimal-path through a static obstacle environment. The attained path is provided for the robot to track during the second level. Simultaneously, the algorithm modifies the path in real-time to prevent a collision by means of any new obstacles, to ensure the sub-paths are optimum. Using proposed path-planning system, the path's optimality is preserved. Fig. 1 depicts this process.

1) The global path planning: By adhering to kinematic constraints, the MOHC-OPF method, which was the objective of our previous research, is used in Global Path Planning to swiftly construct an optimal path for a given environment containing only static obstacles. The MOHC-OPF algorithm employs Open-list and Closed-list. Comparable to the conventional A*, they keep track of the states while searching. The open list includes the neighbours of states that have been expanded during the search process. The closed list comprises all states for which processing has been finalized. Here is a summary of the MOHC-OPF algorithm.

Input: Occupancy grid map of the environment with static obstacles information, starting position (xstart), target position (xgoal), heading θ, and kinematic constraints of simple car-like robot and U(x) actions set with five forward steering angels.

Output: A mobile robot's optimal path from its starting point to its goal, including path length and execution time.

Step 1: Set Open list and Close list

Step 2: Assign Xstart to the start state,

Step 3: For each of the 5 steering angles, find Xstart 's 5 neighbors. Using a simple car-like robot's kinematic model with a global location of (xg, yg, θ) generated using the equation (6) (LaValle, S.M. Planning Algorithms, 2006. [15]).

$$Xg. = ug\cos\theta$$

$$Yg. = ug\sin\theta$$

$$\varphi = us/l\tan\theta$$

$$\rho\min = l/\tan\varphi_{max} \quad \text{_____} \quad (6)$$

Where u is an action set {0,1}, φ is the steering angle, ρ min is the minimum turning radius and l is the front-rear-axles distance of a simple car.

Step 4: If any of the neighbors is a goal state, then quit.

Step 5: Estimate the cost of each neighbor using the cost function equation (1) if they are not likely to collide.

Step 6: Assign Xstart to a neighbor having the minimum cost function f-value, then execute the related action on the map. Keep the former Xstart and the f values in an open list.

Step 7: Repeat from step 3 until Open-list is empty.

2) The local replanning: Global path planning generates the optimal-path in a dynamic-environment with stationary obstacles only. In Local re-planning, the autonomous mobile robot or vehicle takes this optimal path from the starting-point to the target-point in a dynamically changing situation. The robot moves along the course with the aid of surrounding sensors and a scanning procedure. The robot's sensors allow it to survey an area from a 360-degree angle.

The proposed method, NO-MOPP, begins by analysing sensor data to identify any new static or moving dynamic obstacles on the optimal path for tracking. Once an obstacle reaches the robot sensor's exposure range, sensor readings provide all information regarding the robot's movement and location, as well as all obstacles in the surroundings. Using this information, the likelihood of a robot and obstacle collision is evaluated. If there is no collision, the robot will continue along its original path as shown in Fig. 4(a).

However, in the case of a collision, the proposed NO-MOPP technique replans the segment of the path containing a potential collision location. The newly found subpath must be the best and shortest possible. The inventive optimal path was adjusted such that the-robot will track the updated no-collision path. The process of alerting a robot to the presence of new obstacles is known as obstacle detection.

The proposed methodology enables the robot to-move toward the goal though detecting any new obstacles. The following stages are included in this implementation. The measurements of the sensor are recorded. The robot-obstacle distance is then estimated from the robot to the close exterior-surface of the obstacle.

3) Obstacle detection: The obstacle's presence is detected when sensors sense the obstacle. The obstacle-robot distance exceeds a certain threshold. The collision check method is invoked when an obstacle is encountered to find that the robot as well as the obstacle will crash.

4) Collision check: Even if an obstacle is within the sensor range of the robot, not all detected obstacles will cause a collision. If there is no collision as depicted in Fig. 4(a), the robot will follow the optimal reference path. If there is a collision possibility as depicted in Fig. 4(b) and the distance amid the robot & the obstacle is greater-or-equal to the threshold value, then replanning is performed using the proposed local search method called NO-MOPP; otherwise, the robot will halt and pause for the obstacle to go before continuing along the same path.

The likelihood of a collision is computed based on the robot's location and direction angle in relation to the sensed obstacle. Calculating the time and location of the collision is: If the robot's present location is $P_{r\text{-}p1}$ ($x_{r\text{-}p1}$, $y_{r\text{-}p1}$) and it is heading toward $P_{r\text{-}p2}$ ($x_{r\text{-}p2}$, $y_{r\text{-}p2}$), and if the obstacle's present location is P_{obst1} (x_{obst1}, y_{obst1}) and its goal is P_{obst2}(x_{obst2}, y_{obst2}). The formulas of the robot movement are given by (7).

Calculating robot motion requires the following formula:

$$x_{r\text{-}p} = x_{r\text{-}p1} + v_{robot}\, t_r \cos\theta$$

$$y_{r\text{-}p} = y_{robot1} + v_{robot}\, t_r \sin\theta \quad (7)$$

The equation for the movement of the obstacle could be expressed as (8).

$$x_{obst1} = x_{obst1} + v_{obst}\, t_{obst} \cos\varphi$$

$$y_{obst1} = y_{obst1} + v_{obst}\, t_{obst} \sin\varphi \quad (8)$$

The collision amid the robot and the moving obstacle occurs when the subsequent Eq. (9) is satisfied.

$$x_{robot} = x_{obst1} \quad (9)$$

$$y_{robot} = y_{obst1}$$

i, e.

$$x_{r1} + v_r\, t_r \cos\theta = x_{obst1} + v_{obst}\, t_{obst} \cos\varphi \quad (10)$$

$$y_{r1} + v_r\, t_r \sin\theta = y_{obst1} + v_{obst}\, t_{obst} \sin\varphi$$

where θ is orientation and v_{robot} is the robot-velocity, φ and v_{obst} are the orientation and the obstacle-velocity respectively, and tr and t_{obst} are the current time at which the robot and obstacle are there and they are positive. Then the point of the intersection can be determined by replacing Eq. (7) and (8) with Eq. (9) producing equation (10). In the case of a collision, the robot's sub-path consists of 3 locations. The robot's present location is represented by the first point, $X_{present}$ (x_1, y_1), the collision points by X_{colisn} (x_c, y_c), and the next point in its path

by $X_{new\_next}(x_n, y_n)$. If there is a collision, the obstacle will be on this sub-path. For this reason, a modified local search is proposed, NO-MOPP, for replanning in order to locate an alternate collision-free sub-path. When both of the below circumstances are true Local replanning is done by invoking the proposed method NO-MOPP. The first circumstance is when a new obstacle enters the robot sensor's coverage range. Second, when a collision detection judgment is made favourably. In this local search, NO-MOPP looks for the next neighbour with minimum cost value calculated using Eq. (1) in the Global path planning from its neighbor's list, named Open list. The computational cost is less because the next node is selected from the open list which is readily available. Therefore, the replanning has no impact on the time efficiency of the proposed technique. Once again, the new neighboring point is checked for collisions; if there are none, a minimal turning radius path segment is constructed. This is repeated until the goal is found.

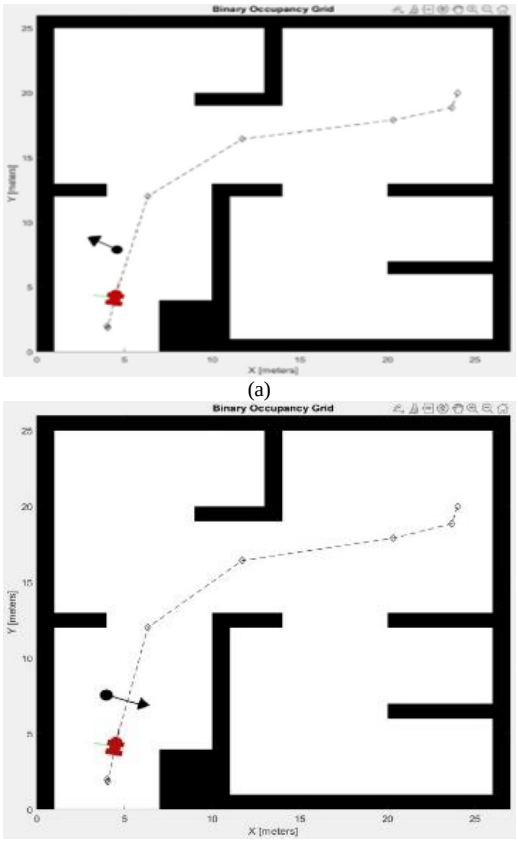

Fig. 4. (a) No collision Case (b) Collision case

The shortest path is guaranteed by a Minimum turning radius path segment. The near-optimal sub-path is returned by the proposed method NO-MOPP. The robot is instructed to pursue this new branch of the course. Because the Global-path-planning of the proposed dynamic path-planning system previously computed the costs of five neighbors corresponding to each steering angle for each location on the global optimum path, the cost computing in the proposed method NO-MOPP is no longer required. There is therefore no computational overhead. The time needed for this is likewise quite short because there are just five neighbour points with regard to five steering angles while looking forward and only those points for which the arc length is greater than the cell size Eq. (4) So, search time is also reduced. Therefore, the method has no consequence on the efficiency of time. The following Algorithm 2 summarizes the pseudo-code of the proposed dynamic path planning algorithm NO-MOPP:

Algorithm 2: NO_MOPP
Input: Start, Goal, Closed-list and Open-list
Output: The near-optimal-path for an autonomous mobile-robot fromthe given initial point to the goal-position in Path length and Execution time.
Algorithm NO_MOPP (Open_List, Closed_List, X_{start}, X_{goal})
1. $X_{next}=X_{start}$
2. index=0
3. optimal_path =Closed_List
4. Pathlength= length (Optimal-path)
5. Robot follows optimal_path
6. for each point in Closed_List ()
7. {
8. index=index+1
9. Robot move forward in the mentioned orientation
10. X_{next} = Closed_List(index)
11. if X_{next} ==Xgoal
12. Print ("Path detected successfully")
13. Print ("Found near Optimal path", Closed_List);
14. Sensor data= Sensor reading from its coverage area
15. If (Sensor data != Obstacle)
16. Robot moves forward to the X_{next}.
17. Else if (Collision_detection () == false)
18. Robot moves forward to the X_{next}
19. Else if
20. {
21. $X_{current} = X_{next}$
22. While (neighbours of $X_{current}$ from Open_List ==true)
23. {
24. $X_{new\_next}=X_{current}$'s neighbours with next minimum cost from Open_List ()
25. $X_{new\_next}=X_{current}$ having next minimum cost from Open_List ()
26. If Collision_detection (X_{new_next}) == false
27. Robot moves to X_{new_next} with the given orientation
28. Update Closed_List
29. Return
30. Else
31. Continue
32. }
33. update path length
34. If (no more neighbors of $X_{current}$ in Open_List)
35. Print ("There is no path exists")
36. }
37. Method: Collision_detection (X_{new_next})
38. {
39. d= distance between the robot's current point and collision point
40. if (d >= threshold value)
41. return false
42. else
43. return true
44. }

IV. Experimental Results

Using MATLAB 2021a on Windows 10 64bit-with an Intel-core i5 NIVIDIA G5-CPU, the performance of the proposed method, a novel Near-Optimal Multi-Objective Path Planner (NO-MOPP), was evaluated. For testing, sensor views from -45° to +45°, -90° to +90°, and -180° to +180° were captured. In ninety percent of the studies, it was discovered that the dynamic obstacles were successfully avoided. On a variety of dynamic maps, all simulations were executed with varying starting and ending points. Path-Length-Mean and Average-Execution-Time between the provided start and goal locations on the map are the performance metrics considered when evaluating the effectiveness of our proposed system. The execution time and length of the near-optimal-path was recorded.

1) Case-study-1: Complex Map: A more complex dynamic environment is considered; the environment is a complex maze with both static and dynamic obstacles that are stationary and moving. The proposed method was executed one hundred times in order to determine its average execution time. Fig. 5(a), (b) and (c) depicts a nearly optimal path generated by NO-MOPP for this complex environment on two levels known as Global path planning and Local-replanning paths and how robot (arrow) avoiding dynamic stationery(purple color) and moving obstacle(green color). In ninety percent of experiments, it successfully avoids dynamic obstacles.

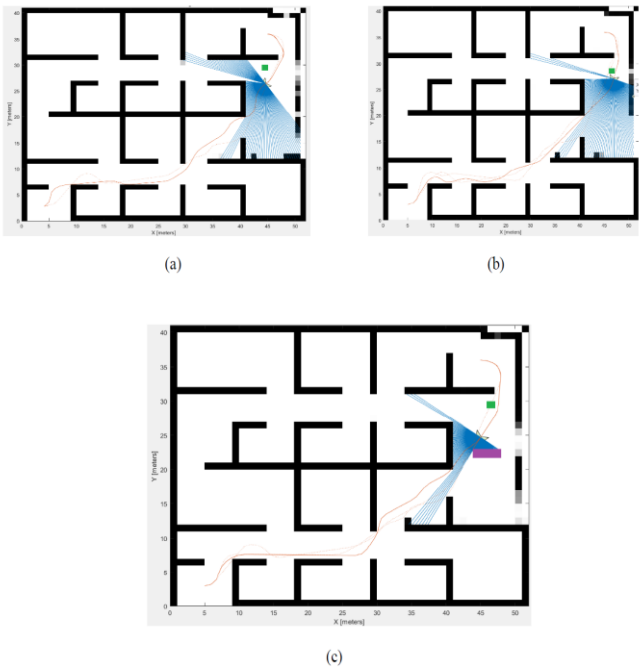

Fig. 5. (a) & (b): No collision and collision avoidance with a moving dynamic obstacle (green color) using NO-MOPP. (c) Collision avoidance with dynamic moving (green color) and stationary obstacles (purple color) using NO-MOPP

The findings for the Complex map's Path Length Mean and Average-Execution-Time are given in below Table I.

TABLE I. Type Results for Complex Map

	Offline path planning	After Local-replanning
Path length_mean in meters	64.5589	67.8374
Execution time in secs	0.457936	1.986376
Direct_Path_Length in meters	51.856	51.856

2) Case-study-2: Package pickup in Warehouse scenario: A package pickup in a warehouse dynamic-environment contains static and dynamic moving and stationary obstacles. Fig. 6(a), (b), and (c) illustrate, NO-MOPP devised a path that was close to optimal for avoiding static (purple color) and moving (green color) obstacles on the way to the package pickup site. In 90% of the experiments, the proposed strategy was effectively avoided.

The Path length mean and execution time outcomes for the Package Pickup scenario are summarized in Table II.

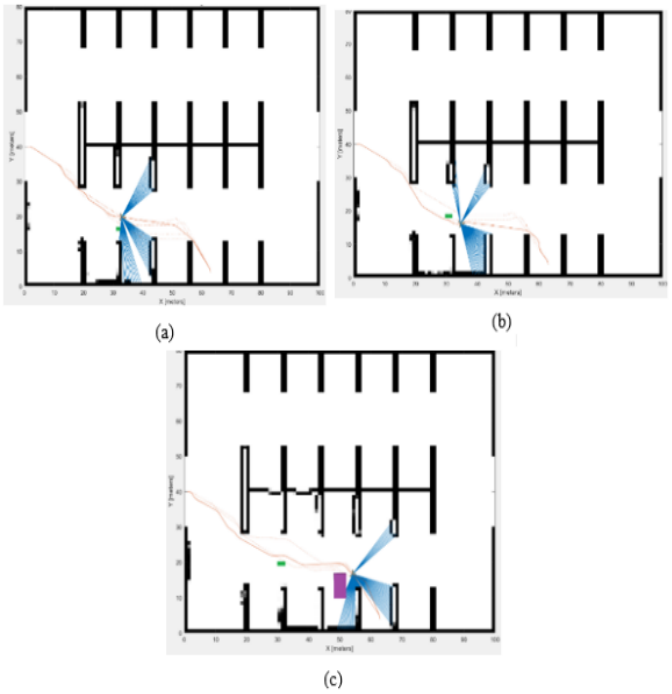

Fig. 6. (a) & (b): No collision, collision avoided with a moving dynamic obstacle (green color) using NO-MOPP. (c) The collision was avoided with the moving dynamic obstacle (green color), and stationery obstacle (purple color) using NO-MOPP

TABLE II. Results for Package Pickup in a Warehouse Scenario

	Global path planning	After Local-replanning
Path length_mean in meters	73.5541	91.5263
Execution time in secs	0.4951	2.953216
Direct_Path_Length in meters	62.1488	62.1488

Performance Evaluation: In the preceding two case studies, we describe the efficacy of the proposed method NO-MOPP and comparability it to the current techniques RRT and RRT*. In this section, we compare NO-MOPP to RRT and RRT*. The proposed method NO-MOPP has been rigorously examined. The comparison of performance is summarised here. In comparison to RRT and RRT*, the path length supplied by the Proposed Method NO-MOPP was superior. The execution time of No-MOPP is considerably shorter.

1) Case Study-1: Complex Dynamic map: Table III provides a visual representation of the results of a performance evaluation that compares the proposed technique NO-MOPP to the existing methods RRT and RRT* when applied to a complex dynamic map. The figures provide abundant evidence that the proposed technique is successful even when the level of map complexity increases.

TABLE III. PERFORMANCE EFFICIENCY COMPARISON IN A COMPLEX MAP

Planner/performance metric	Path length Mean in meters	Avg_Execution Time in secs
RRT	110.873295	4.136285
RRT Star	105.567284	4.513792
Proposed method NO-MOPP	67.8374	1.986376

As is evident from the Fig. 7 below, the proposed method NO-MOPP performed better than the existing methods RRT and RRT* after being run through 100 iterations. The blue line that depicts its performance shows that the proposed method developed the shortest length paths in contrast to other existing approaches in the Complex Dynamic Map. This is proved by the fact that the method created the shortest length paths in each iteration.

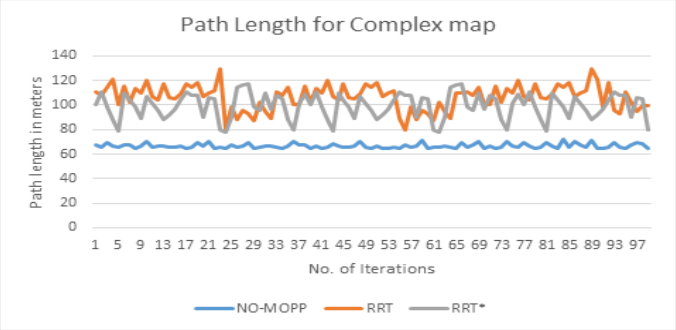

Fig. 7. Path length comparison in complex map

The Fig. 8 displays the efficiency and efficacy of the execution time. The proposed method NO-MOPP is represented by a blue line in virtually all 100 iterations of the Complex Dynamic Map. This method generates paths that require significantly less time consuming compared to the existing techniques RRT and RRT*.

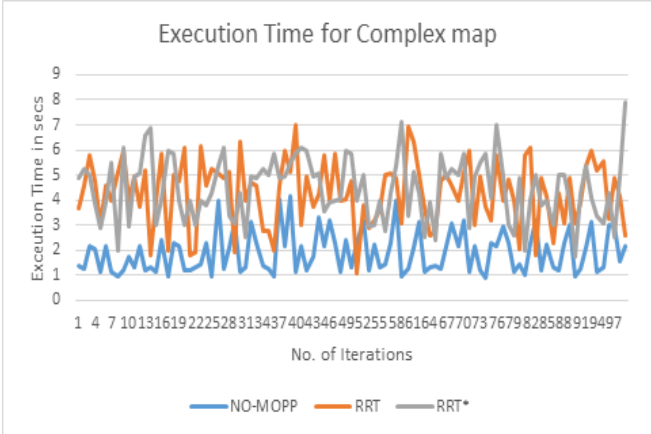

Fig. 8. Execution time comparison in complex map

2) Case Study-2: Dynamic map for Package Pickup: The RRT and RRT* approaches, as well as the proposed approach, were evaluated using the Warehouse scenario as a point of comparison. Table IV presents a comparison of various performance metrics for further consideration. Examining how the proposed solution comes up in path length as well as execution time in comparison to the other available options. The proposed method is carried out fairly well when applied to this difficult warehouse map.

TABLE IV. PERFORMANCE EFFICIENCY COMPARISON IN A COMPLEX MAP IN WAREHOUSE

Planner/performance metric	Path length Mean in meters	Avg_Execution Time in secs
RRT	170.8753	4.596832
RRT Star	150.3547	4.975649
Proposed method NO-MOPP	91.5263	2.953216

The Fig. 9 indicates how the performance of the recommended method NO-MOPP compares to that of the existing methods RRT and RRT* in 100. When compared to the other existing methods in the Warehouse dynamic map, the proposed method NO-MOPP consistently created paths that were the shortest in length. This can be seen from the blue line that depicts the performance of the proposed methodology.

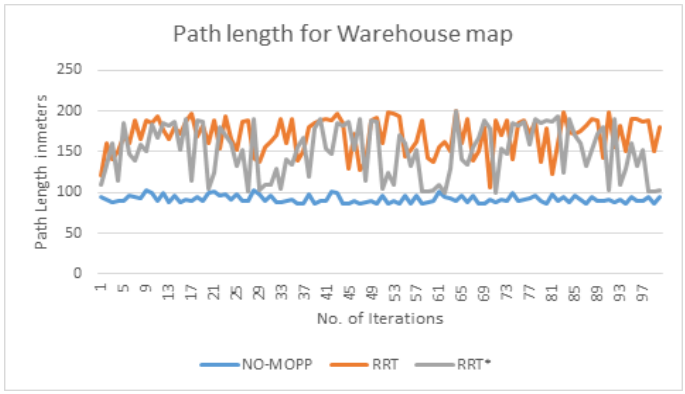

Fig. 9. Performance metric path length comparison in warehouse map

The results of a comparison between the efficiency of our method NO-MOPP and the existing techniques RRT and RRT* can be found in the following Fig. 10, which can be found below. The strategy that was suggested consistently produced the least time-consuming paths when compared to other approaches that were already being used for the dynamic map of the Warehouse.

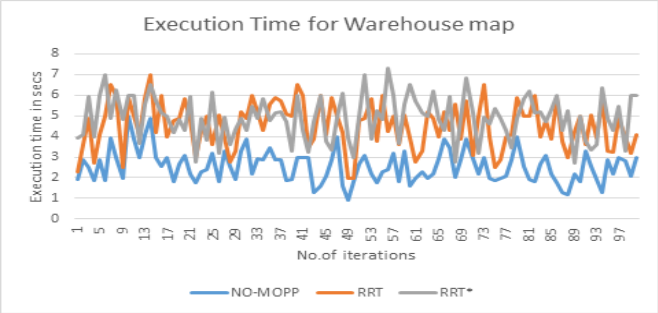

Fig. 10. Performance metric execution time comparison in warehouse map

V. Conclusion and Future Work

In this paper, A Near-Optimal Multi-Objective Path Planner (NO-MOPP) is used to determine the optimal path for mobile autonomous robots operating in dynamic environments. While complying with the robot's kinematic constraints, the robot is able to follow the determined path and avoid new obstacles. Detection of obstacles is dependent on the coverage area of the sensors of a robot and an assessment of the likelihood of a collision. Collision avoidance is achieved through re-planning if the collision check method indicates a collision with a newly introduced dynamic obstacle. Consequently, the development of smooth, drivable paths, which are required for realistic scenarios, is ensured. On average, smoother, collision-free, near-optimal paths may be discovered 90% of the time. NO-MOPP accomplishes Multiple Objective Optimization, which includes Path length, Execution Time, Cost function, and Path Smoothing. Based on the preceding experiments, it is obvious that applying the proposed technique reduced average 15% path length and execution time compared to the existing methods RRT and RRT*. Compared to these existing methods, the proposed method has exhibited superior performance efficiency in complex settings.

Future studies may extend the NO-MOPP method to dynamic situations with higher dimensions for real-time autonomous robots and autonomous vehicles.

References

[1] T. Lozano-Perez and M. A. Wesley, “An algorithm for planning collision-free paths among polyhedral obstacles,” Communications of the ACM., vol. 22, no. 10, pp. 560-570, Oct. 1979.

[2] Lozano-Perez, T., “Spatial Planning: A Configuration Space Approach”, IEEE Transactions on Computers, Vol. C-32, No. 2, February 1983.

[3] Hart, P.; Nilsson, N.; Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics 4(2), pp. 100-107, 1968

[4] Dudi, T.; Singhal, R.; Kumar, R. Shortest Path Evaluation with Enhanced Linear Graph and Dijkstra Algorithm. In Proceedings of the **2020** 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), **2020;** pp. 451–456.

[5] Ammar, A.; Bennaceur, H.; Chaari, I.; Koubaa, A.; Alajlan, M. Relaxed Dijkstra and A* with linear complexity for robot path planning problems in large-scale grid environments. Soft Computing - A Fusion of Foundations, Methodologies, and Applications Volume 20Issue 10Oct **2016** pp 4149–4171 https://doi.org/10.1007/s00500-015-1750-1.

[6] Qing, G.; Zheng, Z.; Yue, X. Path-planning of the automated guided vehicle based on improved Dijkstra algorithm. In Proceedings of **the 2017** 29th Chinese control and decision conference (CCDC), Chongqing, China, 28–30 May 2017; pp. 7138–7143.

[7] Syed Abdullah, F.; Iyal, S.; Makhtar, M.; Jamal, A.A. Robotic Indoor Path Planning Using Dijkstra’s Algorithm with Multi-Laye Dictionaries. In Proceedings of the 2nd International Conference on Information Science and Security (ICISS), Seoul, Korea, 14–16 December 2015.

[8] Sariff, N.; Buniyamin, N. Ant colony system for robot path planning in a global static environment. In Proceedings of the 9th WSEAS International Conference on System Science and Simulation in Engineering (ICOSSSE’10), Takizawa, Japan, 4–6 October 2010; pp. 192–197.

[9] Dang, C.V.; Ahn, H.; Lee, D.S.; Lee, S.C. Improved Analytic Expansions in Hybrid A-Star Path Planning for Non-Holonomic Robots. Appl. Sci. **2022**, 12, 5999. https://doi.org/10.3390/app12125999.

[10] I. Martin, S. Parkes, and M. Dunstan, “Modeling cratered surfaces with real and synthetic terrain for testing planetary landers,” IEEE Trans. Aerospace Electronics Systems, vol. 50, no. 4, pp. 2916–2928, Oct 2014.

[11] X. Ning and L. Liu, “A two-mode INS/CNS navigation method for lunar rovers,” IEEE Trans. Instrumentation and measurement, vol. 63, no. 9, pp. 2170–2179, Sep. 2014.

[12] Miao, H. Robot path planning in dynamic environments using a simulated annealing-based approach. Brisbane: Queensland University of Technology. 10th International Conference on Control, Automation, Robotics and Vision 2008.

[13] Damle, Vikas P., and Seba Susan. "Dynamic Algorithm for Path Planning using A-Star with Distance Constraint." In **2022** 2nd International Conference on Intelligent Technologies (CONIT), pp. 1-5. IEEE, 2022. DOI: 10.1109/CONIT55038.2022.9847869

[14] Siegwart, R., & Nourbakhsh, I. (2004). Introduction to autonomous mobile robots (1st ed.). London: MIT Press.

[15] LaValle, S.M. Planning Algorithms; Cambridge University Press: Cambridge, UK, 2006.

[16] Wang Siming; Zhao Tiantian; Li Weijie, Mobile Robot Path Planning Based on Improved Artificial Potential Field Method, IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE), 2018.

[17] LaValle, S.M. Rapidly-Exploring Random Trees: A New Tool for Path Planning. 1998. Available online: http://lavalle.pl/papers/ Lav98c.pdf.

[18] Karaman S, Walter M, Perez A, et al. Anytime motion planning using the RRT∗. IEEE International Conference on Robotics and Automation (ICRA); 2011

[19] L. Kavraki, P. Svestka, J-C. Latombe, M.H. Overmars, Probabilistic roadmaps ˇ for path planning in high-dimensional configuration spaces, IEEE Trans. on Robotics and Automation 12 (1996), pp. 566–580.

[20] A. Stentz. The focussed D* algorithm for real-time replanning. IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2, August 1995 Pages 1652–1659.

[21] Arora, T., Gigras, Y., & Arora, V. (2014). Robotic path planning using genetic algorithm in a dynamic environment. International Journal of Computer Applications, 89(11), 9–12.

[22] Zhang, X., Zhao, Y., Deng, N., & Guo, K. Dynamic path planning algorithm for a mobile robot based on visible space and an improved genetic algorithm. International Journal of Advanced Robotic Systems, **2016**. 1, 1–17.

[23] Khulna. Karami, A., & Hasanzadeh, M. An adaptive genetic algorithm for robot motion planning in 2D complex environments. *Computers and Electrical Engineering*, 2015, *43*, 317–329. https://doi.org/10.1016/j.compeleceng.2014.12.014.

[24] Islam, R., Muftee, H., & Mahfuzul Hossain, S. (2014). Autonomous robot path planning using particle swarm optimization in a dynamic

environment with mobile obstacles & multiple targets. In International conference on mechanical, industrial and energy engineering (pp.1 – 6).

[25] Ou, Y.; Fan, Y.; Zhang, X.; Lin, Y.; Yang,W. Improved A* Path Planning Method Based on the Grid Map. Sensors **2022,** 22, 6198. https:// doi.org/10.3390/s22166198.

[26] Akshay Kumar Guruji, Himansh Agarwal, D. K. Parsediya, Time-Efficient A* Algorithm for Robot Path Planning, 3rd International Conference on Innovations in Automation and Mechatronics Engineering, **2016**. doi: 10.1016/j.protcy.2016.03.010.

[27] Hong-Mei Zhang, Ming-Long Li and Le Yang, Safe Path Planning of Mobile Robot Based on Improved A* Algorithm in Complex Terrains, Algorithms **2018,** MDPI, Sensors doi:10.3390/a11040044

[28] Khulna. Karami, A., & Hasanzadeh, M. An adaptive genetic algorithm for robot motion planning in 2D complex environments. Computers and Electrical Engineering, 2015, 43, 317–329. https://doi.org/10.1016/j.compeleceng.2014.12.014.

[29] Pham, D. T., & Castellani, M. (2015). A comparative study of the Bees Algorithm as a tool for function optimization. Cognet Engineering, 1–28.

[30] Choset, H.M. Principles of Robot Motion: Theory, Algorithms, and Implementation; The MIT Press: Cambridge, MA, USA, 2005.

[31] Noreen, I.; Khan, A.; Habib, Z. Optimal Path Planning using Memory Efficient A*. In Proceedings of the IEEE International Conference on Frontiers of Information Technology, Islamabad, Pakistan, 19–21 December **2016**; pp. 142–146.

[32] Warren, C.W. Fast path planning using modified A* method. In Proceedings of the IEEE International Conference on Robotics and Automation, Atlanta, GA, USA, 2–6 May 1993.