Improved A* Algorithms For DDMR With Static And Dynamic Constraints With Comparison Study Of DWA, Dijkstra, RRT, And Traditional A*

Basem Fares [*]
Emad Alrouh **
Ahmed Haj Darwish ***
Shaza Otri ****

(Received 17 / 11 / 2024. Accepted 16 / 1 / 2025)

□ ABSTRACT □

This paper presents a path planning method based on the traditional A* algorithm for a two-wheeled Non-Holonomic Differential Drive Mobile Robot (DDMR). The proposed method supports autonomous navigation and obstacle avoidance and also diminishes the drawbacks of A* algorithm, such as large turning angles in the robot's path, unsmooth trajectories, and applicability only to static environments. This method adopts A* algorithm and a weighted heuristic function incorporating curvature. The weighted heuristic function decreases the path length, while the curvature function smooths the path. Three different scenarios (maps) is simulated to evaluate the proposed method. Robot Operating System (ROS), Gazebo and RViz is used to simulate three different scenarios with consistent navigation parameters for the mentioned methods. The results obtained were compared with those of DWA, Dijkstra, RRT, and A* algorithm. The comparison proved the superiority of the proposed method, in removing redundant tipping points to smooth the planned path and shortening its length 60%. The effect of weights and curvature angles on the distance function, as well as the effects of local and global cost map parameters on the proposed method is studied and clarified.

Keywords: Path Planning, Avoiding Obstacles, Navigation Stack, Ros, Rviz, Gazebo, DDMR.

[*] **Dept. Control Engineering and Automation - Faculty of Electrical and Electronic Engineering - University of Aleppo - Aleppo - Syria.**
**** Dept. Mechatronic Engineering- Faculty of Electrical and Electronic Engineering - University of Aleppo - Aleppo - Syria.**
***** Dept. Artificial Intelligence and Natural Languages- Faculty of Informatics Engineering - University of Aleppo - Aleppo - Syria.**
******Postgraduate Student (Ph.D.), Dept. Control Engineering and Automation - Faculty of Electrical and Electronic Engineering - University of Aleppo- Aleppo- Syria. shaza.otri@tishreen.edu**

خوارزمية A* محسنة لروبوت DDMR ضمن قيود ستاتيكية وديناميكية مع دراسة مقارنة مع DWA وDijkstra وRRT وA* التقليدية

باسم فارس*

عماد الروح**

أحمد حاج درويش***

شذى عطري****

(تاريخ الإيداع 17 / 11 / 2024. قُبِل للنشر في 16/ 1 / 2025)

□ ملخّص □

هذه تقدم الورقة البحثية منهجية جديدة لتخطيط المسار تعتمد على خوارزمية A* التقليدية وتطبيقها على روبوت نقال تفاضلي ذو عجلتين من نوع DDMR. تدعم المنهجية المقترحة الملاحة المستقلة مع تجنب العوائق، كما تتعامل مع أوجه القصور في خوارزمية A* التقليدية، مثل زوايا الانعطاف الكبيرة في مسار الروبوت، والمسارات غير السلسة، وقابلية التطبيق فقط في البيئات الثابتة. تعتمد المنهجية المقترحة على تحسين خوارزمية A* بوجود دالة استدلالية مرجحة مدمجة بدالة انحناء، حيث تقلل الدالة التقديرية الموزونة طول المسار، بينما تعمل دالة الانحناء على جعل المسار أكثر سلاسة. تمت المحاكاة من خلال ثلاثة سيناريوهات مختلفة للخرائط مع ضبط بارامترات الملاحة للروبوت والبيئة المقترحة بنفس القيم لتقييم الطريقة المقترحة، في بيئة نظام التشغيل(noetic) ROS ، حيث تم استخدام Gazebo وRViz لمحاكاة ثلاثة سيناريوهات مختلفة مع معلمات تنقل متسقة للخوارزميات التي تمت دراستها لإجراء مقارنة بين الخوارزمية المقترحة مع DWA وDijkstra وRRT وخوارزميات A* التقليدية.

أثبتت المقارنة تفوق الخوارزمية المقترحة نجحت في تنعيم المسار من خلال إزالة نقاط التحول الزائدة، وتقليل طول المسار بنسبة 60%. كما تمت دراسة تأثير الأوزان وزوايا الانحناء على دالة المسافة، وكذلك تأثيرات بارامترات خريطة التكلفة المحلية والعالمية على المنهجية المقترحة.

الكلمات المفتاحية: تخطيط المسار، تجنب العوائق، بارامترات خريطة، Ros, Rviz, Gazebo, DDMR.

* قسم هندسة التحكم والأتمتة – كلية الهندسة الكهربائية والإلكترونية –جامعة حلب–حلب– سورية.

** قسم هندسة الميكاترونكس– كلية الهندسة الكهربائية والإلكترونية –جامعة حلب–حلب– سورية.

*** قسم الذكاء الصنعي واللغات الطبيعية –كلية الهندسة المعلوماتية–جامعة حلب–حلب– سورية.

****طالبة دكتوراه–قسم هندسة التحكم والأتمتة – كلية الهندسة الكهربائية والإلكترونية –جامعة حلب–حلب– سورية.shaza.otri@tishreen.edu

Introduction:

The rapid development of embedded intelligent systems and sensory technologies has speeded up developments of autonomous mobile robots, improving their productivity and efficiency in their various types, such as Humanoid, Industrial, Social, Medical, Service, Rescue, and Agricultural robots [1]. The autonomy of these robots depend on motion planning algorithms that facilitate safe navigation to their targets.

Motion planning consists of two stages: the first one called **path planning**, which focuses on processing time, path length, and path smoothness; while the second stage is **trajectory planning**, that generates the kinetic commands for robot navigation [2].

Path planning algorithms are also classified of two main planners: **global planners**, which require complete knowledge of the environment to establish a collision-free path based on a static map, and **local planners**, which handle updated dynamic obstacles in real time. Moreover, path planning techniques have three primary groups: traditional methods (including graph-based, sampling-based, gradient-based, optimization-based, and interpolation curve algorithms), machine learning and deep learning approaches, and meta-heuristic optimization techniques [3].

Traditional path planning methods are mostly used due to their simplicity, low computational cost, and their online real time abilities. Dijkstra [4] [5] is one of those common and well-known planers. Dijkstra is considered a Graph Search and an Uninformed Search algorithm, it is also a kind of blind search, which is used with discrete maps like Occupancy Grid Map (OGM) [6]. Where it always finds the shortest feasible path. Dijkstra has been modified to make it faster, through A* algorithm. A* is a Graph Search and an Informed Search algorithm. A* also is a typical heuristic and deterministic algorithm based on the heuristic method of finding an optimal path solution [4] [7] [8] [9]. DWA (Dynamic-Window Approach) algorithm is another method that uses robot dynamics to plan the path in the shortest time, where admissible velocities are used. This algorithm is used in local planning [7] [9].

The RRT (Rapidly exploring Random Tree) algorithm is used as a competitor to the A* algorithm for high dimensional spaces. RRT is considered a kind of Monte-Carlo simulator with bias search to build a path tree [5] [10]. This makes the RRT algorithm more complex and difficult to implement.

This paper will focus on adopting A* algorithm characteristics to build a path planning method adaptable to static and dynamic environments.

The rest of the paper is organized as follows: section 2 discusses the A* algorithm and its heuristic functions thoroughly, section 3 illustrates the dynamic model of the simulated robot, section 4 shows simulation tools and the maps used, section 5 shows the results and section 6 concludes the paper.

2. A* algorithm:

A* algorithm is a Graph Search Based, and an Informed Search algorithm. A* is a typical heuristic and deterministic algorithm based on the heuristic method of finding an optimal path solution.

There are issues associated with the A* algorithm that limit its use. The system path planning has an excessive number of inflection points and turns, which makes it difficult for the robot to move in its actual surroundings [4]. A* algorithm is considered a complete, optimal path planning algorithm. It is heuristic deterministic and not scalable well as map size increases. The total cost $f(x)$ is defined as the total real traveled distance $g(x)$ and the heuristic value $h(x)$, as in equation (1).

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

By using acceptable admissible heuristic guidance $h(x)$. A* is both effective and optimal [4].

2.1 A* algorithm heuristic functions:

The heuristic function estimates the distance from the current node to the target node.

a. Euclidean distance:

The distance is the name of the imaginary straight line between the starting and target points, regardless of the obstacles. The Euclidean distance is given by equation (2) [8].

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (2)$$

b. Manhattan Distance:

The Manhattan distance between two points is measured along two axes at right angles. It calculated distance to allow movement (left or right) and (up or down) only. We find that the path includes horizontal or vertical blocks, and the angle between these two blocks is equal to 90 degrees. Given in equation (3). [8]

$$d = |x_1 - x_2| + |y_1 - y_2| \quad (3)$$

c. Octile distance

This function is part of the hierarchical clustering method. The Octile distance calculates the value from x to y. where x,y are a numeric vector or a matrix. The Octile heuristic is used when pathfinding on an 8-connected grid that can be written [11].

d. Quadratic Distance

This function also called Euclidean distance squared, this method recommended to avoid the expensive square root in the Euclidean distance by using distance-squared that given by equation (4) [11].

$$d = ((x_1 - x_2)^2 + (y_1 - y_2)^2)^2 \quad (4)$$

e. Diagonal distance

Diagonal distance can be used if the map allows diagonal movement, where you can move (4 northeast). The Manhattan distance for (4 east, 4 north) will be 8×D. However, diognal move (4 northeast) instead, so the heuristic should be 4×D2, where D2 is the cost of moving diagonally. The Diagonal distance is given by equation (5) [11].

$$d = \max(|x_1 - x_2|, |y_1 - y_2|) \quad (5)$$

2.2 Curvature

In many path-planning algorithms like A*, the raw path generated consists of straight-line segments connecting waypoints the Euclidean distance, as shown in equation (6).

$$d = \sqrt{\left(x_{current} - x_{last\_waypoint}\right)^2 + \left(y_{current} - y_{last\_waypoint}\right)^2} \quad (6)$$

This can lead to sharp turns which make the path slower and less efficient. The function applied is a method for achieving path planning with a smooth trajectory closes to the original path, avoiding abrupt changes in direction. If the angle between segments is small (indicating a straight path), intermediate waypoints can be removed. However, if the angle is large (indicating a sharp turn) the path will be smoothed, and more waypoints should be retained. [12]

The angle is calculated as shown in equation (7).

$$\theta = \arctan(y_{next} - y_{current}) / (x_{next} - x_{current}) \quad (7)$$

3. The kinematic model

The linear velocity and angular velocity are sampled to simulate the robot path that planned with global and local planners with avoiding obstacles behavior in real time [13]. The kinematic model of the DDMR as shown in Figure (4). The linear velocity of DDMR

in the robot coordinates represents the average of linear velocities of the right and left wheels is calculated by

$$v = \frac{v_R + v_L}{2} \quad (8)$$

The angular velocity of DDMR, where the 2L is the distance between two wheels is calculated by.

$$w = \frac{v_R - v_L}{2L} \quad (9)$$

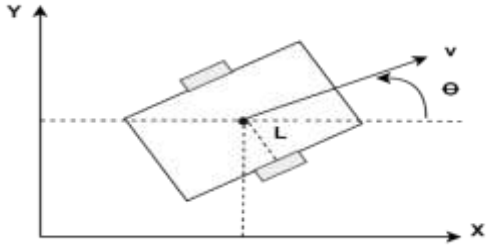


Figure (1) Dynamic model of a mobile robot

An alternative form [13] of the forward kinematic model of DDMR based on linear velocities and angular velocities represented in the robot coordinates is:

$$q^{\cdot I} = \begin{bmatrix} x_a^I \\ y_a^I \\ \theta^{\cdot I} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \quad (10)$$

The last equation is called the DDMR Navigation Equation.

4. The proposed methodology:

A study of A* algorithm heuristic functions such as Euclidean, Manhattan, Octile, Diagonal and Quadratic is made to compare path length, time travel, and iteration for all methods.

The A* algorithm is improved by applying a weighted heuristic function after calibration of weights to reach the optimal weight that generate the shortest path. Then the track is smoothed with curvature method to solve the problem of many turning points, large turning angles that are the traditional A* disadvantages.

The proposed algorithm Data flow diagram

A graphical representation for the proposed A* is generated by using the rqt_graph command to visualize the communication between different active nodes in a ROS system, showing how nodes (processes) are interconnected by topics.

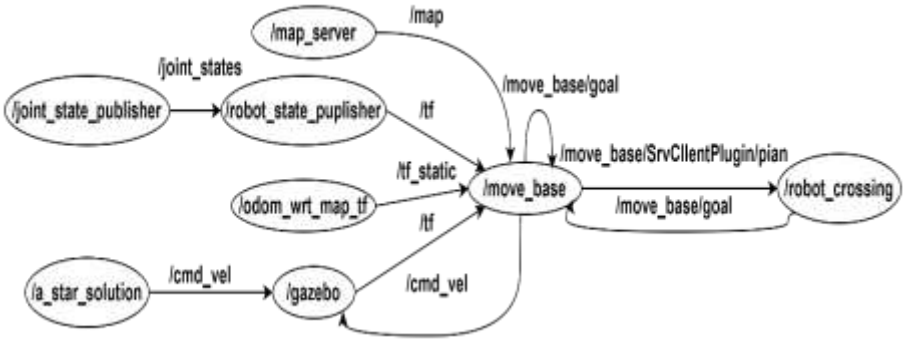


Figure (2): The data flow of the active nodes connection for the proposed A*

The proposed algorithm Pseudocode:

1. Start
2. Create open_list
3. Set closed_list to empty
4. Initialize g_cost = 0 for the start node
5. Determine h_cost = weighted_heuristic_distance from start to goal
6. Set the start node f_cost = h_cost
7. Add start_node to open list
8. Set shortest path = [], path_found = False
9. while (open_list is not empty)

- sort open_list according to the lowest f_cost
- extract the first lowest f_cost element to current_node
- close current_node to prevent from visiting again by adding to closed_list
- if current_node is the goal then
  - path_found = True
  - breack
- get neighbors of current_node
- Loop neighbors
  - if the neighbor is visited then continue
  - calculate g_cost of a neighbor
  - let h_cost = weighted heuristic distance of current neighbor

f_cost = g_cost + w * h_cost

- if the neighbor is in open_list

if f_cost for neighbor is bigger than f_cost then update the node's f_cost inside the open_list

else

add neighbor to open_list

- if path_found
  - append the goal to the closed_list
  - reconstruct the path backward from the target
  - **Generate the optimal smooth path**

7. End

Transformation Tree for robot system

A graphical representation of the robot coordinate frames (such as the robot's base, sensors, and other components) connections, that visualize the linear graph of all ROS nodes links when started, by using the tf_tree, it's ROS command tools, as shown in Figure (3).

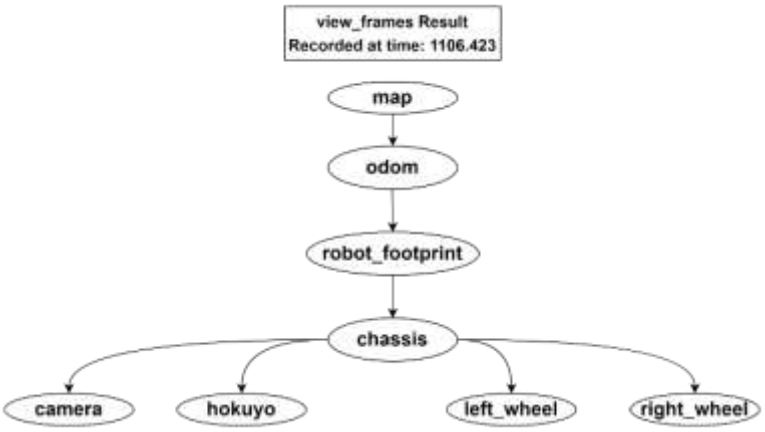


Figure (3): Transformation Tree for robot system showing all the sensors.

The control units of the software system produce appropriate velocity commands and recovery behavior when needed, this node helps the robot to escape dynamic obstacles by making a 360-degree rotation to perform obstacle avoidance. If the obstacles continue along the track even after rotation, the robot stops with error message [14] [15].

5. Simulation:

The experimental environment is Ubuntu 20.04 operating system, and the experimental platform is ROS (Noetic). A simulation environment with varying levels of complexity is created using the GAZEBO 3D simulation tool, as shown in Figure (4-b). This tool accurately models the dynamic characteristics of the robot and visualizes sensor data using RViz 2D tool. The map constructed with the Rviz simulator is depicted in Figure (4-a). Three different scenarios are simulated to evaluate the proposed method: (1) robot navigation within a room, (2) the robot movement passing through critical regions through a doorway, and (3) the navigation in environment with dynamic obstacles.

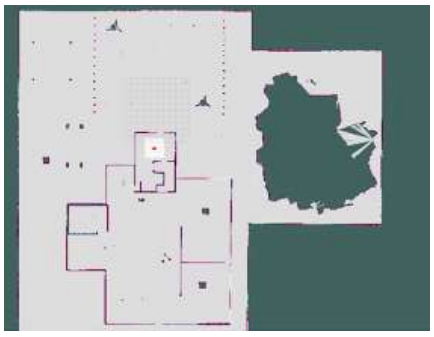


Figure (4-a): Map model of the proposed environment in the Rviz

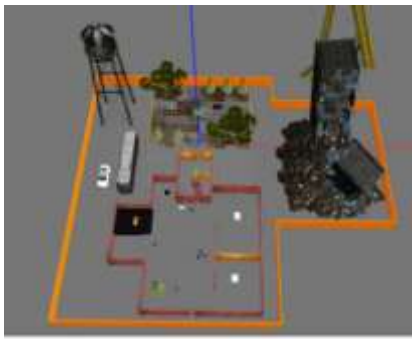


Figure (4-b): The Environment model in Gazebo simulator

A major benefit of mobile autonomous robots is the unique ability to move independently from the starting point to goal, through a dynamic constraint environment, without any collision. The navigation stack is used to move DDMR autonomously using SLAM (Simultaneous Localization And Mapping).

Results and Discussion:

6. Result Analysis:

Several experiments were conducted to analyze the performance of different Global Planner algorithms, with the same simulation environment, and the same robot parameters. Where the Translation velocity is 0.5 m/sec and the angular velocity is 0.5 rad/sec.

The study is done in three scenarios. The first scenario, the path planning algorithms is implemented on the DDMR robot in an internal static simulation environment with obstacles fixed on the map when the map is built. The start position of the robot and the goal position is inside the construction (indoor). As shown in figure (5).

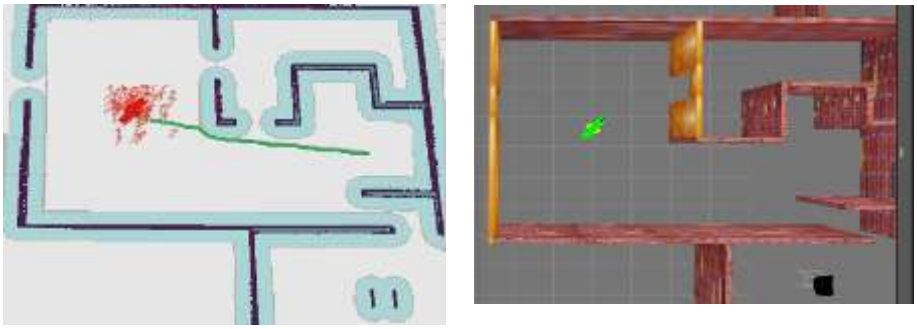


Figure (5): The first scenario in static environment

Path planning algorithms (DWA, Dijkstra, RRT, A* and the improved A*) are implemented, where C++ and Python codes are written to calculate the distance travelled between the initial and the goal, the time (Sec) and the number of iterations to reach the goal. The results shown in Table (1).

Table (1): Time, Iteration and path traveled in an static environment

scenario 1 - Initial position and goal indoor			
Algorithm	**Time**	**iteration**	**Path length**
DWA	1.69	106	3.87
Dijkstra	1.74	107	3.82
RRT	1.50	118	8.6
A*	1.68	108	3.81
improved A*	**1.48**	**117**	**1.62**

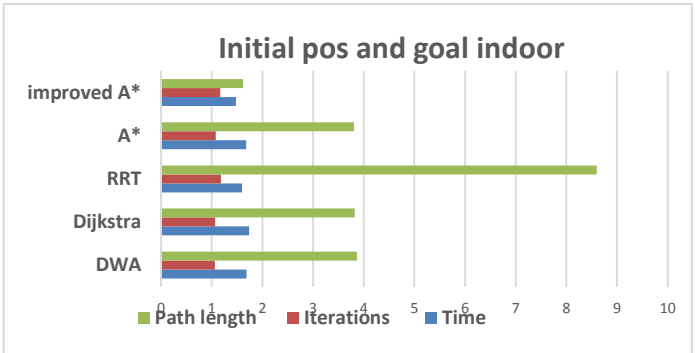


Figure (6): The path planning algorithms in static environment

The second scenario is implementing the path planning algorithms for an environment with static obstacles. The start position of the robot is inside the room while the goal is outside the room, passing through the door (critical area). As shown in figure (7).

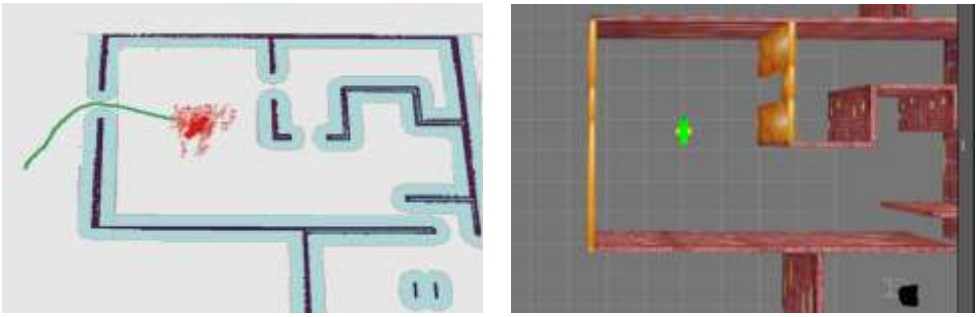


Figure (7): The second scenario (static environment), passing critical area

The results are shown in Table (2), the RRT algorithm was unable to perform properly, due to a critical area between the robot and the goal. Such areas lead to poor performance of algorithms that rely on artificial intelligence.

scenario 2 - Critical path			
Algorithm	**Time**	**iteration**	**Path length**
DWA	1.49	099	3.20
Dijkstra	1.33	086	2.67
RRT	1.88	132	5.03
A*	1.86	122	4.14
improved A*	**1.73**	**116**	**1.55**

Table (2): Time, Iteration and path in an outdoor static environment

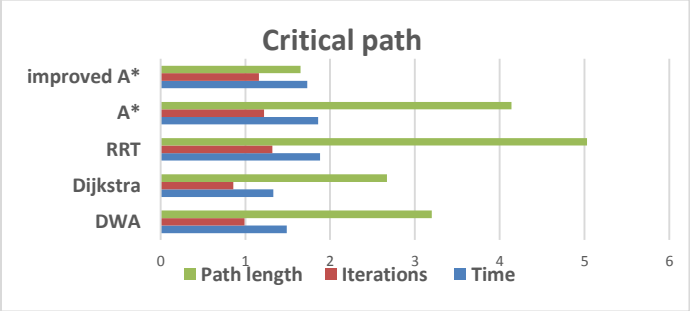


Figure (8): The path planning algorithms in critical path

Examples of critical areas include:
• Non-circular obstacles.
• Very large obstacles.
• Obstacles with L_ shape.
• Obstacles close to each other, which prevent the robot passing through them.
The third scenario, the path planning algorithms are implemented for DDMR robot with dynamic obstacles environment, detecting with sensors during movement in real time. The impact of dynamic obstacle is shown in Table (3). As shown in figure (9).

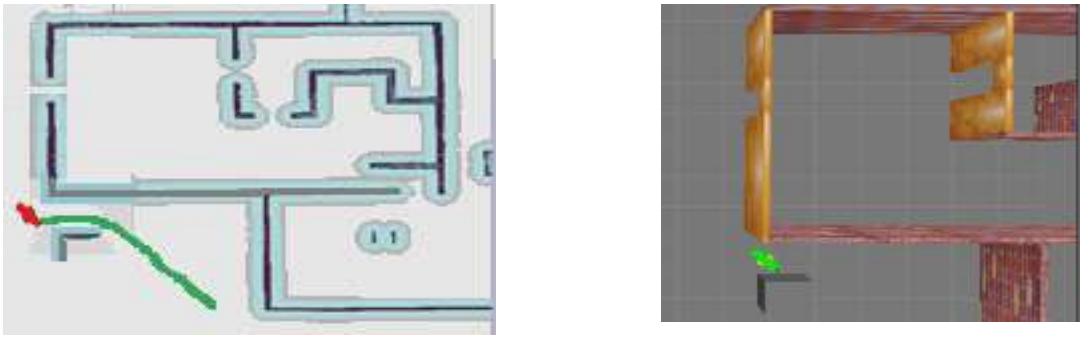


Figure (9): The third scenario in dynamic environment

scenario 3 - Dynamic Obstacle			
Algorithm	**Time**	**iteration**	**Path length**
DWA	2.49	116	7.01
Dijkstra	2.29	145	5.26
RRT	1.54	137	6.51
A*	2.12	165	6.50
improved A*	**1.67**	**140**	**1.73**

Table (3): Time, Iteration and path in environment with dynamic obstacles

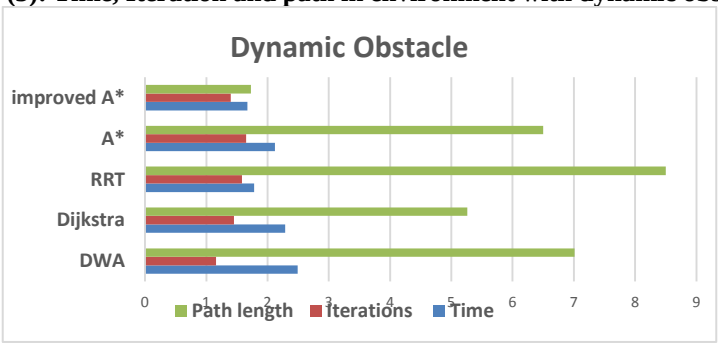
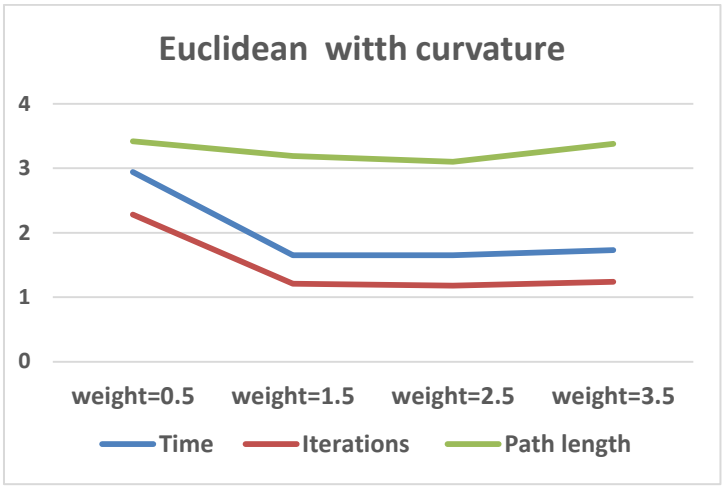


Figure (10): The path planning algorithms with dynamic obstacle

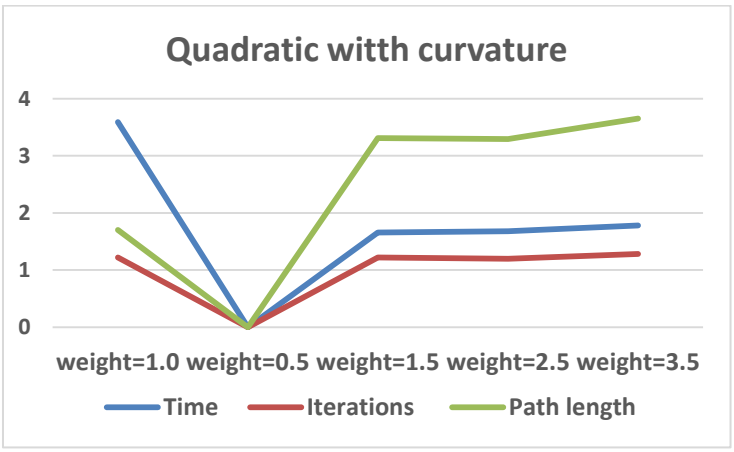
The case study, shows that the improved A* algorithm with weighted heuristic function, gives better results in path length and iteration compared to traditional A*. As the study shows that the path length is less 6% for Euclidean, 10% for Manhattan, 29% for quadratic, compared with the traditional algorithm and increasing the efficiency of robot movement. The proposed improved A* used weighted quadratic heuristic functions with a curvature function of pi/8 curvature angle, improved the performance leading to smooth shorter trajectory. The improved A* reduced path length by 60% and time travelled by 9% in most cases.

Figure (11) shows, reducing the weight of the heuristic function less than one make the algorithm work bad (big time travelled and path length values), in different cases. Which confirms the important role of the heuristic function.

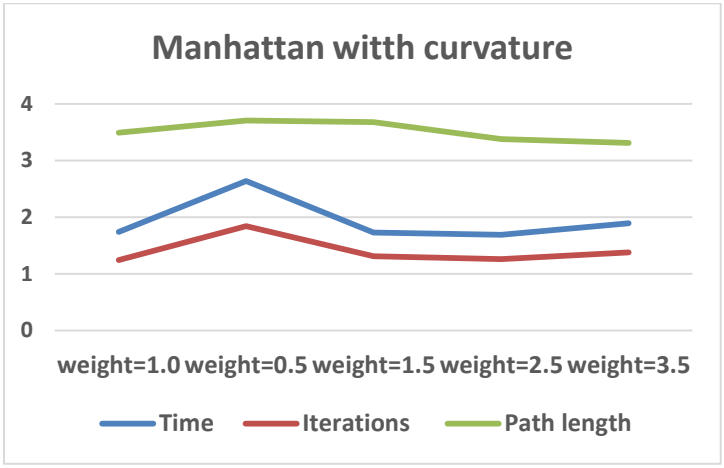
The proposed methodology with the weighted heuristic function with 0.5 has worse performance for the significantly increase in the time travelled and path length values for all heuristic functions studied, as in figures (11-a, c, d). This performance due to the weakness of the inference function, which caused by the rotation of the robot in path searching and sometimes it is stuck as the quadratic heuristic function, as shown in figure (11-b), and giving winding longer path as shown in figure (12-a, b, c).


Euclidean witth curvature
4
3
2
1
0
weight=0.5
weight=1.5
weight=2.5
weight=3.5
Time
Iterations
Path length

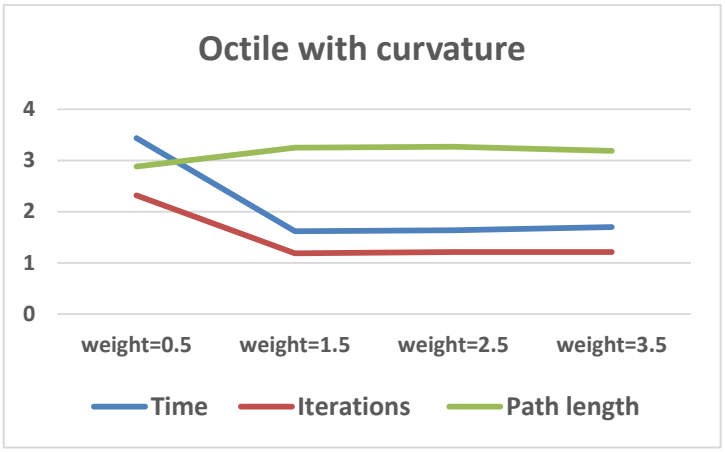
(a)


Quadratic witth curvature
4
3
2
1
0
weight=1.0
weight=0.5
weight=1.5
weight=2.5
weight=3.5
Time
Iterations
Path length

(b)


Manhattan witth curvature
4
3
2
1
0
weight=1.0
weight=0.5
weight=1.5
weight=2.5
weight=3.5
Time
Iterations
Path length

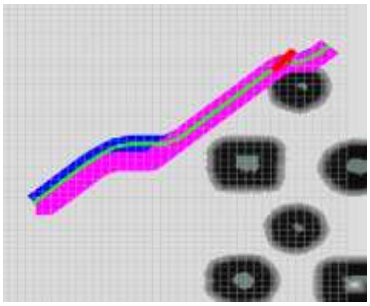
(c)



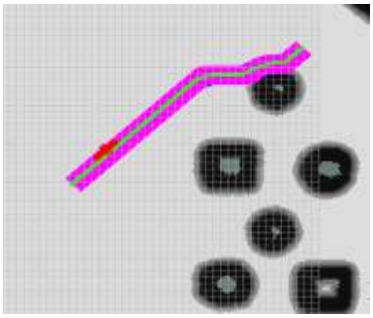
(d)

Figure (11). Weight consequence comparison of different method.

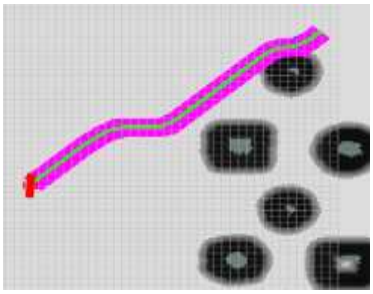
The optimal weighted heuristic function which is weighted quadratic function with weight by (1.6) with curvature by angle (π /8) shown in figure (12-d). The proposed methodology converts the Piecewise linear path into smooth curvature path removing redundant tipping points in the planned path and shortening its length.



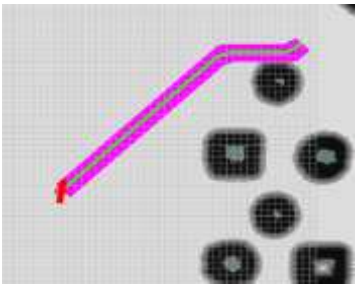
a.Euclidean_curvature_0.5_weight



b. quadratic_curvature_0.5_weight



c. Octile_curvature_0.5_weight



d. Optimal weighted Quadratic function with weight 1.6

Figure (12). Comparison of weighted heuristic functions with curvature on path track

Cost map Parameters:

The global planner uses a variety of algorithms like the proposed algorithm to compute a high-level, optimal path for a robot to move from its current position to a target location over a global map of the environment. The Local planner receives the updated Sensors and Odometry data and the path provided by global_planner, produces appropriate linear and angular velocities along the path, publishing to the move_base node to control the robot.

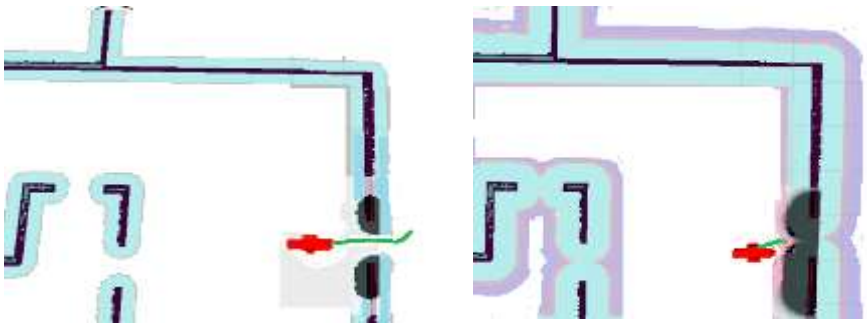
Once the path provided by global_planner fails with dynamic obstacles, the local_planner requests the global_planner to calculate another.

A launch file is created to execute multiple nodes such as the map server, AMCL, move_base, Gazebo and Rviz packages in the same time and publish the parameters of the global and local planners that stored in the parameter servers.

The global cost map consists of three layers: the Static Map Layer, the Inflation Layer and the Obstacle Map Layer. The Inflation Layer adjusted by the parameters inflation_dist, cost_scaling_factor and inflation_radius. The higher inflation_dist, the higher the obstacle surrounding area. The Obstacle Map Layer used to track dynamic obstacles, adjusted by obstacle_range and raytrace_range parameters. The local cost map consists of Inflation and Obstacle Map Layers only.

The global planner package provide nodes to plan the optimal path from current position to goal before moving. The global_planner publish the velocity commands on /plan topic [15]. Global_planner uses algorithms such as DWA, A*, Dijkstra or RRT etc.

The costmap parameters set to specific values due to the map size and the robot Dimensions. Where the obstacle_range is set to 0.35 to prevent the robot from stuck the obstacle. The inflation_radius is set to 0.2 to allow the proposed robot to exit critical areas like doors to reach the goal, regarding to its dimension, Otherwise inflating obstacles too much preventing the robot from passing through the door, as in the figure (13). To select the path planning algorithm. The use_dijkstra parameter is set to default when using Dijkstra algorithm and to false when using A*.



a- tuned parameter **b- default parameters**

Figure (13). The impact of costmap parameters tuning

To use the navigation stack properly, parameters is tuned as Table (4).

Table 4. Parameters modified for the proposed DDMR

Parameters	Definition	Values
obstacle_range	The default maximum distance from the robot at which an obstacle will be inserted into the cost map in meters.	0.35
inflation_radius	The radius in meters to which the map inflates obstacle cost values.	0.25
raytrace_range	The default range in meters at which to ray trace out obstacles from the map using sensor data.	2.0
max_vel_x	Max value of forward translate velocity	0.5

max_vel_theta	Max value of angular velocity	0.5
min_obstacle_dist	The allowed minimum Euclidean distance to the obstacle	0.5
goal_tol	Acceptable limit of distance to goal	0.5
K_in	Max value of iterrations	4000
D	Max value of each branch length	0.4

Conclusions and Recommendations:

This research, presents a proposed methodology to address the issues of the traditional A∗ such as redundant tipping points, long search time, unsmooth paths, close proximity to obstacles, and applicability only to static maps. The proposed algorithm presents A∗ algorithm improved by using an optimal weighted heuristic function to decrease the path length, with curvature formula to deal with large inflection points and turning angles in the search path.

The simulations-based results show that the algorithm significantly improves path smoothness, path length, path planning time compared to traditional algorithms, by reducing path length by 60% in all scenarios, and time travelled in critical path scenario by 6% whereas in dynamic obstacle scenario reduced by 28%, which increasing the robot movement efficiency.

The simulation results prove the ability of the improved A∗ algorithm to increase the path smoothness by removing redundant tipping points, and decreasing path length, with more iterations, which leads to shorter and smoother path, leading to better handling complex environment conditions by avoiding dynamic obstacles effectively.

References:

[1] B. Popoola, ROBOTICS AND MACHINE LEARNING, 2024.

[2] A. Gasparetto, P. Boscariol, A. Lanzutti and R. Vidon, "Path Planning and Trajectory Planning Algorithms: A General Overview," in *Motion and Operation Planning of Robotic Systems. Mechanisms and Machine Science*,, G. Carbone and F. Gomez-Bravo, Eds., Springer, Cham, 2015.

[3] M. Reda, A. Onsy, A. Haikal and A. Ghanbari, "Path planning algorithms in the autonomous driving system: A comprehensive review," no. 174, 2024.

[4] Q. Hongwei, S. Shiliang, W. Ting, X. Yu, J. Yi and C. Zonghan, "Review of Autonomous Path Planning Algorithms for Mobile Robots," vol. 7, no. 3, 2023.

[5] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavarki and S. Thurn, Principles of Robot Motion: Theory, Algorithms, and Implementations, MIT Press, 2005.

[6] T. Grebner, P. Schoeder, V. Janoudi and C. Waldsch, "Radar-Based Mapping of the Environment: Occupancy Grid-Map Versus SAR," *IEEE Microwave and Wireless Components Letters*, pp. T. Grebner, P. Schoeder, V. Janoudi and C. Waldschmid, 2022.

[7] K. Li, X. Gong, M. Tahir, T. Wang and R. Kumar , "Towards Path Planning Algorithm Combining with A-Star Algorithm and Dynamic Window Approach Algorithm," vol. 6, no. 14, 2023.

[8] X. Li, X. Hu, Z. Wang and Z. Du, "Path Planning Based on Combination of Improved

A-STAR Algorithm and DWA Algorithm," in *2nd International Conference on Artificial Intelligence and Advanced Manufacture (AIAM)*, Manchester, United Kingdom, 2020.

[9] Y. Li, R. Jin, X. Xu, Y. Qian, Z. Wang, S. Xu and H. Wang, "A Mobile Robot Path Planning Algorithm Based on Improved A* Algorithm and Dynamic Window Approach," 2022.

[10] Z. Mi, H. Xiao and C. Huang, "Path planning of indoor mobile robot based on improved A* algorithm incorporating RRT and JPS," *AIP Advances*, 2023.

[11] A. Patel, "Amit's A* Pages," [Online]. Available: https://theory.stanford.edu/~amitp/GameProgramming/index.html. [Accessed 10 11 2024].

[12] A. G. P. E., "Introducing Calculus to the High School Curriculum: Curves, Branches and Functions," *ATLANTA*, p. Page 23.815.3, 2013.

[13] R. Dhaouadi and A. Abu Hatab, "Dynamic Modelling of Differential-Drive Mobile Robots using Lagrange and Newton-Euler Methodologies: A Unified Framework," *Advances in Robotics & Automation*, vol. 2, no. 2, 2013.

[14] A. Patil, "Recovery Behaviours used in Navigation Stack - ROS," Medium, [Online]. Available: https://medium.com/@patiladitya1309/recovery-behaviours-used-in-navigation-stack-ros-5df991f408c7. [Accessed 10 11 2024].

[15] K. Zheng, ROS Navigation Tuning Guide, researchgate, [Online]. Available: https://www.researchgate.net/publication/318011822, 2020.