OPTIMIZATION of PATH in MOBILE ROBOTS USING PRM ALGORITHM

Sakshi Arali, Rutvik Deshpande, Vedant Pagar, Roshan Waghchaure, Sanjay Narayankar
Student,Mechanical, PCCOE&R, India
Student,Mechanical, PCCOE&R, India
Student,Mechanical, PCCOE&R, India
Student,Mechanical, PCCOE&R, India
Assistant Professor,Mechanical,PCCOE&R, India
E-mail:sakshi06092001@gmail.com, vedantpagar5@gmail.com , , rutvikdeshpande22@gmail.com, roshanwaghchaure98@gmail.com , sanjay.narayankar@pccoer.in

Abstract **- In the current study of mobile robots, the path planning algorithm and optimization in both static and dynamic situations are the key issues that are being addressed. Path planning can be used to resolve a wide range of issues in a variety of industries. It can direct the robot to arrive at a specific location or destination using very basic trajectory planning to the choice of an appropriate series of actions. The focus of this project is on the algorithms that are used to optimize the path of a mobile robot using MATLAB and other important resources. For static known obstacles, the A* and D* algorithms are described with a suitable justification and mathematical relationships. The PRM (probabilistic roadmap) technique is also covered in this. This algorithm creates a network graph of potential paths in a given map based on free and occupied regions. The robot is initially placed in an area with static known obstacles, and is then assigned to a start and target point. Additionally, waypoints generated by MATLAB are used to provide the robot with instructions so that it can navigate and follow an optimized course while avoiding potential hazards. With regard to time, obstacles, and distance, the optimal path is. Typically, Arduino is used to combine hardware components into software programmes, but in this project, we utilized MATLAB for better computation because it provides accurate optimized paths that are integrated with ESP. As a result, there are fewer hardware and software components, which decreases computing time**.

Keywords: **Machine learning, algorithm, PRM, Optimized path, MATLAB, Integration, known static obstacles, differential drive robot, waypoints**

I. Introduction

Making the best or most efficient use of a circumstance or resource is the definition of optimisation in its literal sense. In a number of sectors, including manufacturing, healthcare, transportation, and agriculture, mobile robots are becoming more and more prevalent. However, it can be challenging to optimize the path planning and navigation of mobile robots. Traditional path planning methods usually require a lot of physical labour and can be time-consuming. To overcome these challenges,researchers are examining how machine learning (ML) and artificial intelligence (AI) could be applied to enhance the path planning of mobile robots.

By merging AI and ML with mobile robots, it is possible for mobile robots to learn from their surroundings, adapt to changing conditions, and make decisions in real-time based on the data they collect. Path planning that takes advantage of AI and ML algorithms may find the fastest route, avoid obstacles, and use less energy. These algorithms may take into account a wide range of variables, including the layout of the environment, the robot's speed, and the available sensors. The benefits of using AI and machine learning to path planning for mobile robots are significant. By maximising path planning, mobile robots can boost safety, decrease expenses, and increase production.

In general, path optimisation in mobile robots is required to boost productivity, guarantee safety, effectively manage resources, meet time sensitive requirements, navigate complicated settings, and enable coordination in multi-robot systems. It is essential for enhancing the capabilities and effectiveness of mobile robots in a variety of applications.

A. Reasons why Path Optimization is Required in Mobile Robots-

Effectiveness, Collision-avoidance, Allocating resources, Complex environments, Multi-robot coordination, Optimization problem.

II. Main Problems & Objectives

- Since robots are never able to decide between which is the longest or shortest route, we analyse their beginning place and navigation to the end destination.
- Working on selection of elements that must be taken into account for the path to be ideal.

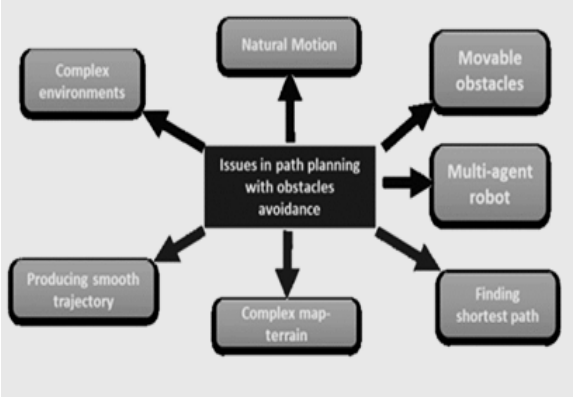

Fig.1. Difficulty with path planning and avoiding obstacles

Journal of Engineering Design and Computational Science(JEDCS) Volume 3, Issue 3, May 2024

Objectives-

- Mapping the environment into arrays to provide MR a better understanding of where it is in relation to its target.
- Pathfinding algorithm selection based on the shortest trip distance and time between the starting point and the destination.
- Making Robot Travel along the best route possible by using several PID controllers.

III. Path planning by algorithms and matlab

The process of choosing an ideal or practical route for a robot to go from its current location to a desired objective location in its surroundings is
referred to as path planning in the field of robotics. It entails assessing the robot's environment, taking into account barriers and limitations, and creating a trajectory or series of moves that the robot should do in order to get where it's going.

For autonomous robots to move autonomously in changing and uncharted situations, path planning is crucial.The properties of the environment, robot kinematics, impediments, and dynamic limitations are all taken into account by path planning algorithms. Based on the particular demands of the robot's task and application, they seek to strike a balance between effectiveness, safety and optimality.

A. Concepts to be considered in the path planning process:

1) *Perception*: Using sensors the robot senses its surroundings in order to learn about barriers, the
terrain, or other important elements.

2) *Environment modelling*: A representation or model of the robot's environment is created using the perceived data. Techniques like occupancy grids, point clouds, or feature maps can be used for this.

3) *Path generation*: A number of algorithms are used to create a path or trajectory for the robot using the environment model. These algorithms might be as basic as potential fields or bug algorithms or as complex as A* (A-star), Dijkstra's algorithm, or Rapidly Exploring Random Trees (RRT).

4) *Path length, safety, smoothness, and energy economy* are some of the parameters used to analyse and optimize the created path. The path can be improved and refined as needed by using optimisation strategies.

5) *Execution and control*: By directing its actuators, such as wheels or joints, to follow the desired trajectory, the robot executes the planned
path. To monitor the robot's actual position and make modifications as necessary, sensors provide feedback.

B. Key factors for path planning-

1) *Type*:

i) Global planning, for example, assumes that all environmental data is known. So, before the robot travels, the full course is plotted.

ii) Local planning, sometimes referred to as sensor-based planning, is the use of sensors by a robot to gather data when the data is either unknown or only partially understood. The robot is moving as the course is being carefully planned.

2) *Time:*

i) Online: Using data from the robot's sensors, the path from the start to the goal points is planned during the robot's movement.

ii) Off-line: Before the robot moves, the entire course is planned.

3)*Environment*:

i) dynamic environment means that there are both stationary and
moving obstacles around the robot.

ii) Static Environment: The only barriers in the robot's environment are static.(an acknowledged static setting/unknown static configuration.)

Based on the results of the aforementioned investigation, the following approach was adopted to plan the robot's route:

- Time dependable,
- internet, and the type of plan- Local.
- Environmentally speaking: Static Barrier.
- Obstacle Type: Known or Unknown Static

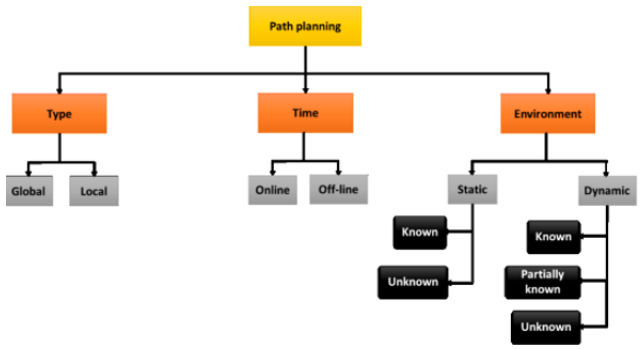

Fig. 2 . Path planning approach

No.	Global path planning	Local path planning
1	Work off-line	Work on-line
2	Robot Map-based	Robot Sensor-based
3	Deliberative navigation	Reactive navigation
4	workspace area is fully known and the terrain should be fixed.	workspace is not necessarily fully known or unknown
5	The algorithm produces a whole path from the initial point to the target point before the robot begins its movement.	The algorithm produces a new pathway in echo to environmental moveable.
6	Approximately slower response	Fast response

Fig.3. Difference between global and local path planning

C. Algorithm Study-

The two primary algorithm techniques were the focus of the algorithm study.

1) *Classical Approach:* In the context of robotic route planning, the term ”classical approach”
refers to established, conventional techniques for creating paths. The following are some significant old algorithms:

i) *Dijkstra's algorithm*: This calculates the shortest route from the beginning node to every subsequent node by iteratively navigating the graph.

ii) *A* (A-star) algorithm*: It is an extension of Dijkstra's algorithm and uses heuristics to more effectively direct the search towards the objective. It incorporates the projected cost from the beginning node to a node (g-cost) as well as the cost of travelling from that node to the target (h-cost).

iii) *Rapidly-exploring Random Trees (RRT):* They are a frequently employed probabilistically complete approach for path planning in high dimensional domains.

iv) *Probabilistic Roadmap (PRM)*: It works by selecting valid configurations at random and connecting them along collision-free lines, it includes creating a map of the environment. Path planning is done by locating a path inside the roadmap using graph search algorithms like Dijkstra's or A*.

2)*Heuristic Approach:* Heuristic techniques use heuristics or approximated measures to direct the search for an ideal or workable path in the context of path planning in robotics. These algorithms try to find a decent compromise between finding solutions quickly and being computationally efficient.

i) *Theta* algorithm:* Theta* is an enhancement of the A* method that increases path smoothness by taking neighbouring node visibility into account. Theta* looks for line-of-sight visibility between nodes rather than traversing the full grid.

ii) *D* (D-Star) algorithm:* The D* method is made for dynamic contexts where the configuration of the map or obstacles can vary while the algorithm is being executed. When the environment changes, it incrementally updates the path.

iii) *Genetic Algorithms:* Inspired by evolutionary ideas, genetic algorithms (GAs) employ a population-based strategy to look for the best
pathways. Chromosomes are used to represent paths, and genetic operators such as crossover and mutation are used to create new paths.

D. PRM Algorithm Applied-

Robotics motion planning algorithms like the Probabilistic Roadmap (PRM) algorithm are used to plan a robot's movement inside a specific environment.The PRM algorithm creates a roadmap of the environment, which is a graph with nodes that stand in for legitimate robot configurations and edges that stand in for practical routes connecting them. Construction and search are the two fundamental components of the PRM algorithm. The programme generates a collection of random robot configurations in the environment and determines if they are collision free during the construction phase. A configuration becomes a node on the roadmap if it is collision-free. A configuration becomes a node on the roadmap if it is collision-free. The programme then creates edges connecting the roadmap's nodes, which represent viable routes between them. This procedure is carried out again and again until the roadmap has enough nodes and edges. The PRM algorithm searches the roadmap during the search phase to identify a route between the starting configuration and the desired robot configuration. An established graph search algorithm, such as A* search, is used to do this. The algorithm looks for a route that connects the starting configuration and the desired configuration while avoiding environmental impediments. If a path is discovered, the robot can advance from the starting configuration to the goal configuration by following it. A popular path planning approach called Probabilistic Roadmap (PRM) builds a roadmap of the surrounding area to quickly identify viable paths.

E. Description of the PRM Algorithm's Operation-

1) *Initialization:*

•In the workspace, create a number of randomly sampled configurations (called nodes). These setups must to be reliable and devoid of collisions.
•Check for collision-free pathways between close configurations to connect them.The roadmap's edges are produced by these connections.
•Creating the road map.

2) *For every set configuration:*

•Choose a portion of the neighbouring configurations.
•Between the current configuration and its neighbours, look for collision free pathways.
•Add an edge to the roadmap connecting the current configuration with its neighbours if there is a collision-free path.

3) *Improving connections:*

•Make further attempts at connecting configurations that weren't initially connected. As a result, the roadmap's connectedness is enhanced and a wider range of paths are covered.

4) *Phase of query:*

•Check to see if the start and goal configurations are included in the roadmap when given a start configuration and a goal configuration. If not, locate the roadmap configurations that are closest to the start and objective and add them.
•Use a path planning algorithm to determine the shortest route between the roadmap's start and goal settings (for example, Dijkstra's algorithm or A*). A series of roadmap layouts joined by edges make up this journey.

5) *Smoothing the path:*
•Apply a route smoothing technique to the path to enhance it and make it more efficient or natural after getting the path from the roadmap.
• This may entail eliminating pointless waypoints, using optimisation methods to reduce path length, or enhance other criteria.

6) *Execution of the path:*
•Use motion control methods to direct the robot down the straight path.
• To precisely follow the course, this may need organising and carrying out small motions, such as straight-line segments or manoeuvres around obstructions.

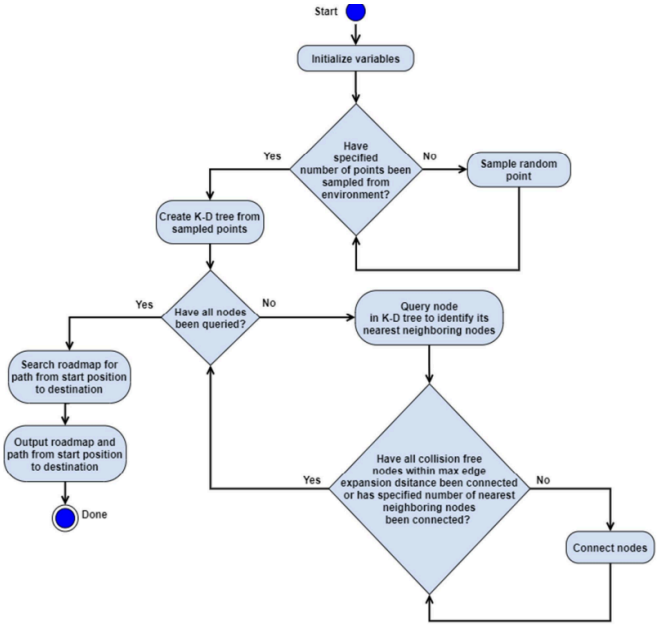

Fig.4. Working of PRM algorithm

By sampling several configurations and creating a roadmap that reflects the connection of the surrounding environment, the PRM algorithm efficiently explores the configuration space. The algorithm decreases the computational work necessary during query time by computing the roadmap in advance.

As a result, PRM is especially advantageous in scenarios involving complicated surroundings, high-dimensional configuration spaces, or dynamic environments where quick planning is required. The quality and density of the sampled configurations, as well as the roadmap's connection, all affect PRM's effectiveness. The effectiveness and calibre of the created pathways can be affected by changing parameters like the amount of samples or connection radius. Numerous robotic applications, such as autonomous vehicles, industrial automation, and humanoid robots, have successfully exploited the PRM algorithm.

F. Creation of Maze/surrounding in MATLAB:

- Construct a scenario to represent a moving robot moving around a space. The example shows how to generate a scenario, model a robot platform using a rigid body tree object, extract a binary occupancy grid map from the scenario, and use the mobileRobotPRM route planning algorithm to plan a path for the mobile robot to follow.
- A machinePlatforms that can move and a collection of stationary obstacles make up the scenario object. The mobile robot in the scenario can be modelled using the robotPlatform object. In this example, a ground plane and box meshes are used to construct a room.
- The room's walls are created as box meshes. The binary occupancy map used for path planning includes these barriers since the static meshes are inserted with the IsBinaryOccupied value set to true.
- For path planning, get an occupancy map as a binaryOccupancyMap object from the scenario. The occupied areas on the map should be 0.3 metres larger.

```
scenario = robotScenario(UpdateRate=5);
```

Add a plane mesh as ground plane in the scenario.

```
floorColor = [0.5882 0.2941 0];
addMesh(scenario,"Plane",Position=[2 2 0],Size=[4 4],Color=floorColor);
```

The walls of the room are modeled as box meshes. The static meshes are added with the `IsBinaryOccupied` value set to `true`, so these obstacles are incorporated into the binary occupancy map used for path planning.

```
wallHeight = 1;
```

```
wallHeight = 0.5;
wallWidth = 0.01;
wallLength = 4;
wallColor = [1 1 0.8157];
addMesh(scenario,"Box",Position=[wallWidth/2, wallLength/2, wallHeight/2],...
    Size=[wallWidth, wallLength, wallHeight],Color=wallColor,IsBinaryOccupied=true);
addMesh(scenario,"Box",Position=[wallLength-wallWidth/2, wallLength/2, wallHeight/2],...
    Size=[wallWidth, wallLength, wallHeight],Color=wallColor,IsBinaryOccupied=true);
addMesh(scenario,"Box",Position=[wallLength/2, wallLength-wallWidth/2, wallHeight/2],...
    Size=[wallLength, wallWidth, wallHeight],Color=wallColor,IsBinaryOccupied=true);
addMesh(scenario,"Box",Position=[wallLength/2, wallWidth/2, wallHeight/2],...
    Size=[wallLength, wallWidth, wallHeight],Color=wallColor,IsBinaryOccupied=true);
addMesh(scenario,"Box",Position=[1.1, 1.3, 0.25],...
    Size=[2, wallWidth, wallHeight],Color=wallColor,IsBinaryOccupied=true);
addMesh(scenario,"Box",Position=[3.1, 2.6, 0.25],...
    Size=[1.6, wallWidth, wallHeight],Color=wallColor,IsBinaryOccupied=true);
```

```
show3D(scenario);
lightangle(-45,30);
view(60,50);
```

Obtain Binary Occupancy Map from Scenario

Obtain an occupancy map as a binaryOccupancyMap object from the scenario for path planning. Inflate the occupied spaces on the map by 0.3m.

```
map = binaryOccupancyMap(scenario,GridOriginInLocal=[-2 -2],...
                                  MapSize=[15 15],...
                                  MapHeightLimits=[0 3]);
inflate(map,0.3);
```

Visualize the 2-D occupancy map.

Fig. 5. Creation of maze and inputs for static obstacle with visualization of maze

G. Path Planning in MATLAB

In the context of robotics, the term "surrounding" typically refers to the environment or the immediate surroundings of a robot. It encompasses the physical space in which the robot operates, including objects, obstacles, terrain, and other elements that are present in its

vicinity.Understanding and perceiving the surrounding environment is crucial for a robot to perform tasks effectively and navigate safely.

To perceive the surrounding environment, robots often employ a combination of sensors such as cameras, LIDAR (Light Detection and Ranging), ultrasonic sensors, infrared sensors, and others. These sensors provide the robot with information about the surrounding objects, their positions, distances, shapes, and other relevant features.By analyzing the data from these sensors, the robot's perception system can create a representation of the surrounding environment, commonly referred to as a "map" or a "scene." This representation allows the robot to make informed decisions, plan its actions, and interact with the objects or navigate through the environment.

For example, a robot in a warehouse might use its sensors to detect and avoid obstacles, locate specific items, or navigate around a complex layout of shelves and pallets.Overall, path planning plays a crucial role in enabling robots to autonomously navigate their environment, avoid obstacles, and reach their desired destinations efficiently and safely.

1) *Steps in MATLAB:*

- Find a path between the start and goal coordinates on the acquired map using the mobileRobotPRM path planner.
- Set the mobile robot's start and end locations.
- With the binary occupancy map and the maximum number of nodes, create a mobileRobotPRM object. The maximum distance between the two connected nodes should be specified.
- Set the rng speed for repeatability.
- Identify a route that connects the start and goal positions.
- Use the waypointTrajectory System object to create a trajectory for the mobile robot to follow, complete with waypoints from the intended route.
- Load the Clearpath Husky mobile robot as a rigidBodyTree object from the robot library.
- Build a robotPlatform object containing a waypoint for the robot's trajectory and a rigidBodyTree object for the robot's modelSystem of Trajectories object.
- Imagine the situation involving the robot.
- Imagine the intended course.
- Make the simulation ready. Simply step through the simulation and update the visualisation at each step as the robot's positions are all known in advance.
- Reset the simulation in the scenario to run it one more and see the results.

By this method we were able to create the maze and the robot was able to find the optimised via. waypoints.

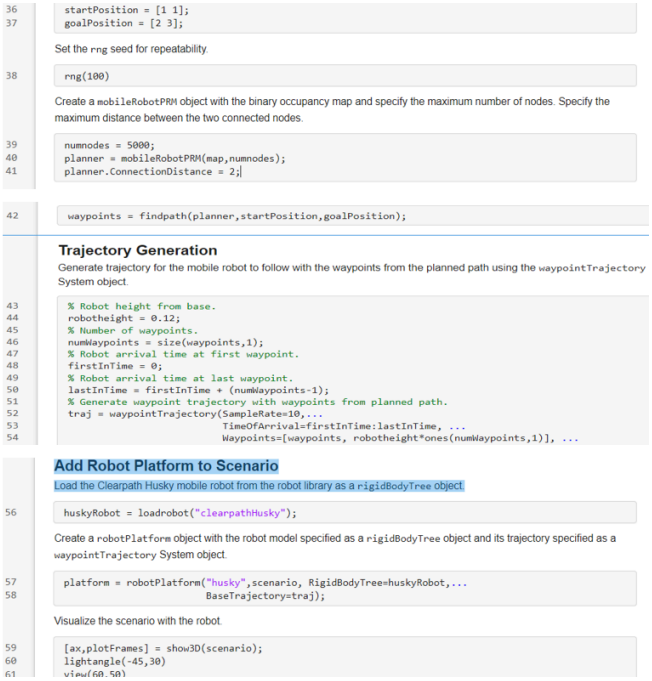

Fig. 6. Illustration of working of code

2) *Libraries used in MATLAB:*

- Robotics System Toolbox: This toolbox includes features for creating, modelling, and operating robotic systems. It contains algorithms for motion planning, perception, and robot dynamics. Additionally, it permits tethering to widely used robot hardware platforms.
- Robotics Toolbox for MATLAB (Peter Corke's Toolbox): This is a well-liked MATLAB toolbox for study and instruction in robotics. It provides kinematics, dynamics, trajectory generation, and visualisation features for robots. Additionally, it supports a variety of mobile robots and manipulators as well as multiple robot models.
- BinaryOccupancyMap: Robotics System Toolbox's "BinaryOccupancyMap" class is available in MATLAB. It represents a grid map of 2D occupancy where each cell can either be occupied (1) or empty (0). In robotics, it is frequently used for path planning, obstacle representation, and environment modelling.
- mobileRobotPRM: For the environment map supplied in the Map property, the mobileRobotPRM object serves as a roadmap path planner object. A roadmap, which is a network graph of potential routes in the map based on free and occupied places, is created by the object using the map. In order to identify an unobstructed route from one point to another, you can adjust the ConnectionDistance and NumNodes parameters to reflect the complexity of the map.

- WayPointTrajectory: The"waypointTrajectory" class from the Robotics System Toolbox is available in MATLAB. It represents a trajectory in a 2D or 3D space that is determined by a series of waypoints.In robotics applications, it is frequently used for path planning, motion planning, and trajectory generating.
- rigidBodyTree: The "rigidBodyTree" class from the Robotics System Toolbox is available in MATLAB. For modelling and simulating robotic systems, it represents a tree structure of connected rigid bodies.

You can provide the kinematic structure, joint types, joint limitations, and other characteristics of the robot using the rigidBodyTree class.

H. Differential Drive Robot Principle:

The movement of a differential drive robot is controlled by two different wheels or sets of wheels. Due to the autonomous driving of each wheel, the robot may turn and move in different directions.The robot may move in a variety of ways by changing the wheel's speeds and directions. For instance, both wheels must rotate at the same speed and in the same direction in order to drive forward.

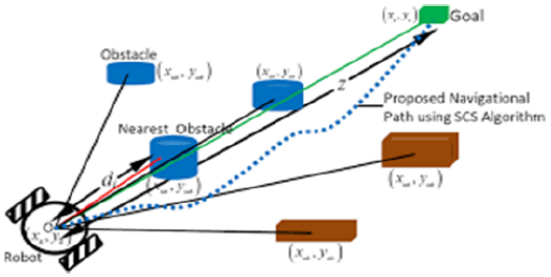

Fig. 7. Illustration of differential drive robot principle.

The wheels rotate in opposite directions or at different speeds to turn the robot, which pivots about a central axis.Due to its simplicity and manoeuvrability, differential drive robots are frequently utilised in applications including robotics contests, mobile robots, and tiny vehicles. Different techniques, such as manual input, sensor feedback, or self-running algorithms, can be used to control them. Some of the crucial calculations pertaining to the principle are listed
below:

1) Wheel Speed:
Speed of each wheel: *(Robot Speed * 2) / Wheel Diameter*.In this case, "robot speed" stands for the intended speed of the robot, and "wheel diameter" refers to the size of the wheels.

2) Wheelbase: The wheelbase of a robot is the separation between its two wheels. It is essential for figuring out the robot's manoeuvrability and turning radius where,
turning radius = (wheelbase/2)

3) Angular Velocity: *(2 * robot speed) / wheelbase*

4) Arc Length:arc length = *radius * angle.*

Here, "radius" refers to the robot's turning radius and "angle" to the angle at which it turned.

5) Odometry: Based on the robot's wheel motions, odometry calculates the robot's position and orientation. You can use the following equations to determine the change in position:
*delta x = (left wheel distance + right wheel distance) / 2 * cos(theta)*
*delta y = (left wheel distance + right wheel distance) / 2 * sin(theta)*
delta theta = (right wheel distance - left wheel distance) / wheelbase

To achieve desired behaviours or activities, further calculations may be required, depending on the particular requirements and control algorithms.

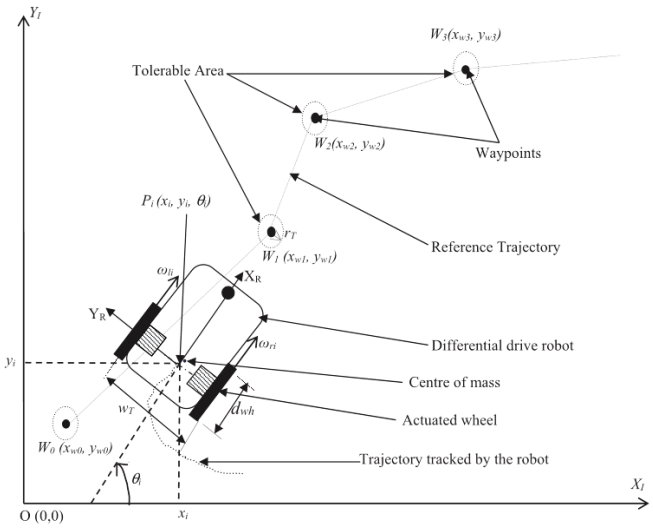

Fig. 8. Mathematical model of trajectory path

I. Components Used:

1) A 60RPM DC motor with gears
2) A caster wheel
3) 70X40 White Double Wheel 2
4) ESP8266-32 - The ESP8266-32 is a microcontroller module built on Espressif Systems' ESP32 system-on-a-chip (SoC).C++, MicroPython, and the Arduino IDE are just a few of the programming languages that can be used to create programmes for the ESP8266-32 module.
5) 13A DC MOTOR DRIVER
6) BATTERY, 3.7V
7) 3 CELL BATTERY HOLDER (4)
8)PINWIRE
9) ROCKER SWITCH IRON 2

J. Algorithm Validation:

The start and goal locations on this test case's simple square map are on two opposing corners, hence the diagonal is the only feasible way. However, the path appears to be slightly bent, therefore we must now adjust our algorithm to choose the best possible route. We just adjusted the number of nodes from 2500 to 10000, and the output shows that we

have improved the perfect path.Adjust the Nodes' NumberThe number of nodes, or points, that are placed on the map and used by the algorithm to produce a roadmap is specified by NumNodes. The method connects all places that do not have obstructions in the straight path between them using the ConnectionDistance attribute as a criterion for distance.The efficiency of the path can be improved by adding additional nodes by providing more viable routes. The intricacy, however, lengthens processing time. You might need a lot of nodes to have adequate map coverage. Some regions of the map might not have enough nodes to connect to the rest of the map because of the nodes' haphazard placement. In this illustration, you create a mix of many and few nodes in a roadmap Algorithm verification using a single barrierSince our algorithm has been improved, we can now verify it for a straightforward obstacle between the maps. Here is the straightforward map that we used for the test, complete with a single barrier.

Now that the algorithm has been fine-tuned, we can use it to navigate increasingly complicated mazes because it is capable of quickly determining the best route across the gaps. The start and destination locations are now 1, 1, and 2, and we will now determine the ideal path points between this difficult path and those between it.

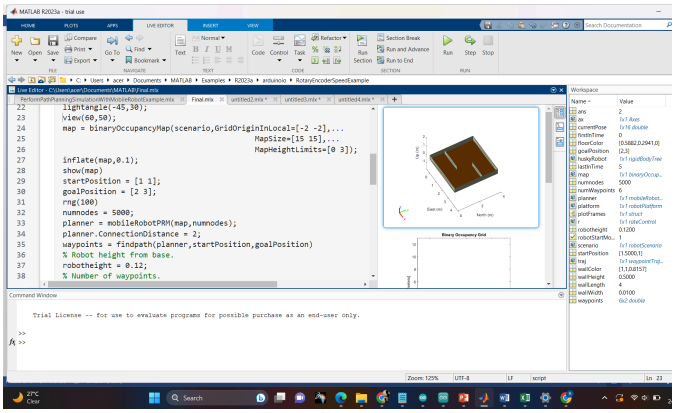

Fig. 9. Maze is created

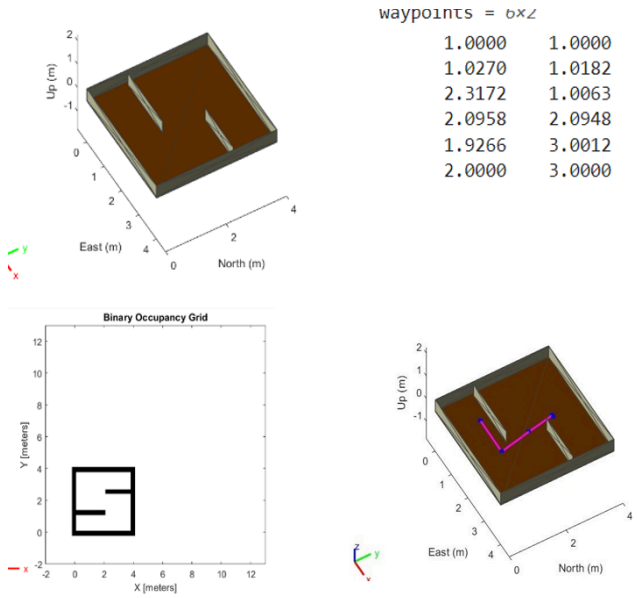

Fig.10. Final waypoints and optimized path given by algorithm for maze with two obstacles.

K. Robot Validation:

Tracking the physical robot's movement and comparing it to the path predicted by the MATLAB simulations allowed us to thoroughly examine the robot throughout the robot validation phase. Our goal was to verify that the real robot's trajectory closely mirrored the simulation's progression.Through careful observation and data collecting, we discovered that the robot consistently followed a route that matched the path suggested by MATLAB simulations. This verification process verified the accuracy and dependability of our algorithm as well as its successful use for real-world robotic mobility.Our method's effectiveness in carrying out the necessary path planning and control is demonstrated by how well the path taken by the actual robot lines up with the MATLAB-simulated path. Our robotic system's dependability and performance are verified through this certification process.

L. Working of Model:

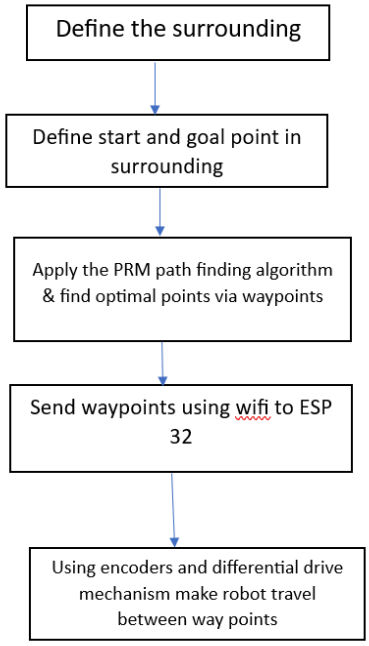

Fig. 11. Flowchart of working of robot in actual.

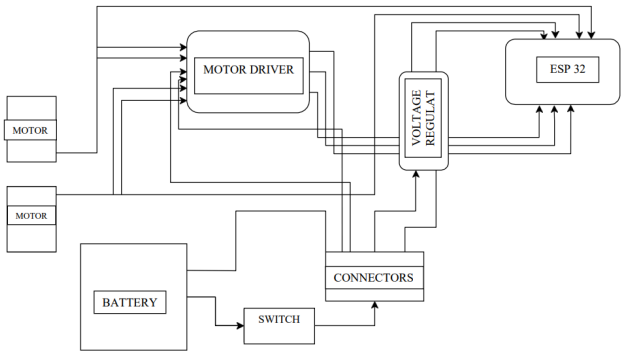

Fig.12. Schematic wire diagram of robot

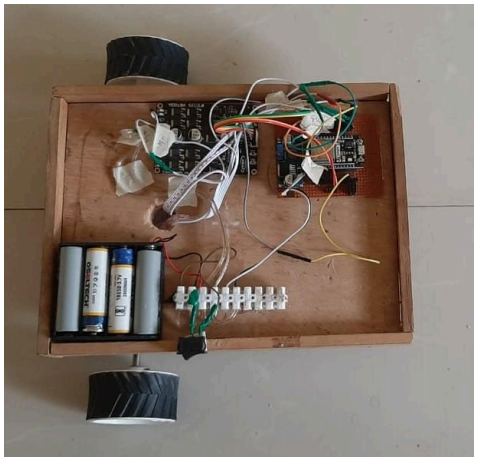

Fig. 13. Top view of actual robot.

M. Results

We first thoroughly assessed our algorithm while considering fundamental conditions. We found several problems with the path created throughout this operation. We then adjusted our algorithm to overcome these problems, leading to the creation of an ideal path. We carried out multiple test cases incorporating obstacles and various environmental factors to confirm the efficacy of our system. Every time, we were successful in finding the best course of action.

During the robot validation step, we followed the robot's path to make sure it matched the path discovered during the algorithm testing. Surprisingly,we discovered that the robot continually took the same optimal path, demonstrating the precision and dependability of our system.

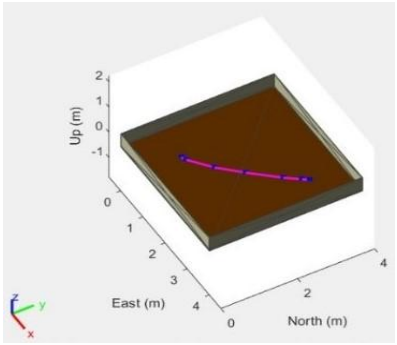

Fig.14. Case1- Maze with no obstacles

Fig. 15. Actual Result of Case1.

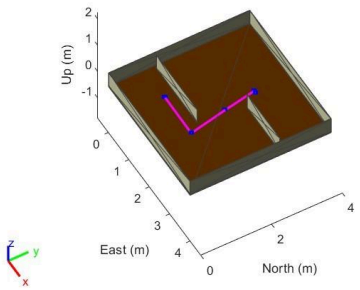

Fig. 16. Case2- Maze with known static obstacles.
(Blue points- obstacles, Pink- waypoints)

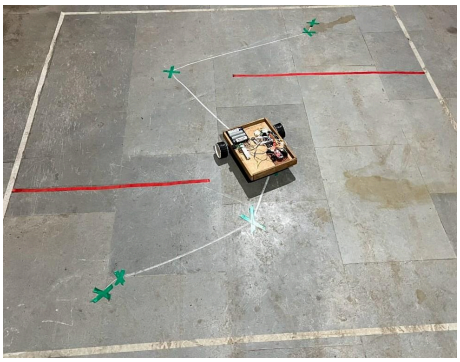

Fig.17. Actual Result of Case2.

IV. Conclusion

For MATLAB case1 - Based on the information provided, we were able to obtain the optimal path between two diagonal points. Initially, there was a slight deviation in the path taken by the robot. However, after tuning the angle settings in the ESP code, the robot was able to trace the optimal path accurately. Overall, the conclusion drawn is that by fine-tuning the angle settings in the ESP code, the robot's movement was improved, and it successfully followed the optimal path.

For MATLAB case2 - Based on the information provided,more complexity was added to the scenario, specifically involving multiple waypoints.The challenge was to ensure that the robot correctly followed all the waypoints in order to reach the goal point. In this case, the robot successfully followed 7 waypoints using the optimal path obtained in MATLAB.

This conclusion highlights the successful navigation of the robot through a more complex scenario. By incorporating multiple waypoints into the robot's path planning algorithm, the robot was able to follow the optimal path generated in MATLAB. It successfully visited all 7 waypoints in the correct order, ultimately reaching the goal point. The ability of the robot to accurately navigate through a series of waypoints demonstrates the effectiveness of the path planning algorithm and the implementation of the optimal path obtained from MATLAB.It indicates that the robot's control system and navigation capabilities were capable of handling the added complexity and accurately following the desired trajectory.

Acknowledgment

References

[1] Changgeng Li , Xia Huang * , Jun Ding , Kun Song , Shiqing Lu, 2022 "Global path planning based on a bidirectional alternating search A* algorithm for mobile robots"

[2] Bing Fu, Lin Chen, Yuntao Zhou, Dong Zheng, Zhiqi Wei, Jun Dai, Haihong Pan, 2018
"An improved A* algorithm for the industrial robot path planning with high success rate and short length"

[3] Mohd Nadhir Ab Wahab , Samia Nefti-Meziani , Adham Atyabi, 2020 "A comparative review on mobile robot path planning: Classical or meta-heuristic methods?"

[4] B.K. Patle , Ganesh Babu L , Anish Pandey , D.R.K. Parhi , A. Jagadeesh "A review: On path planning strategies for navigation of mobile robot"

[5] Mustafa Salah Abed, Omaar Farouq, Qusay F .Al-Doori, 2021 "A Review on Path Planning Algorithms for Mobile Robots"

[6] Mohd. Nayab Zafara,*, J. C. Mohantab, 2018
"Methodology for Path Planning and Optimization of Mobile Robots: A Review"

[7] Tahseen Fadhel Abaas1 and Alaa Hassan Shabeeb1, 2020
"Path Planning Optimization of a Mobile Robot based on Intelligence Algorithm"

[8] HyeokSoo Lee and Jongpil Jeong, 2021
"Mobile Robot Path Optimization Technique Based on Reinforcement Learning Algorithm in Warehouse Environment"

[9] Ahmed Hussein, Heba Mostafa, Mohamed Badrel-din, Osama Sultan and Alaa Khamis, 2012
"Metaheuristic Optimization Approach to Mobile Robot Path Planning"

[10] Martha Nohemi Acosta Montalvol.2020.
"Introduction to Interfacing Arduino Hardware And MATLAB®-Simulink®"

[11] Robins Mathew, Somashekhar S Hiremath, 2016
"Trajectory tracking and control of differential drive robot for predefined regular geometrical path"

[12] Ayorkor Mills-Tettey, Vincent Lee-Shue Jr. Prasad Narendra Atkar, Kevin Tantisevi

"Robotic Motion Planning: A* and D* Search"
[13] https://in.mathworks.com/help/robotics/ug/perform-pathplanning-simulation-with-mobile-robot.html

[14] Jacqueline Jermyn , 2021
"A Comparison of the Effectiveness of the RRT, PRM and Novel Hybrid RRT-PRM Path Planners"