ISSN No:-2456-2165

Path Planning of Mobile Robot Using ROS

Ayush Salunke
Department of Computer Engineering K J Somaiya Institute of Technology
Mumbai, India

Sanika Churi
Department of Computer Engineering K J Somaiya Institute of Technology
Mumbai, India

Mandar Bivalkar
Department of Electronics Engineering, K. J. Somaiya Institute of Technology
Mumbai, India

Mohd Tabish Khan
Department of Computer Engineering K J Somaiya Institute of Engineering and Information Technology
Mumbai, India

Abstract:- The objective of our project is to develop a mobile robot capable of traversing an arena from a starting point A to a destination point B by taking the shortest path possible and avoiding any obstacles that may be encountered. The robot will be designed to accept user inputs from various sources, such as mobile devices or computers. Based on the given inputs, the robot will create a path to follow and navigate accordingly. Our aim is to equip the robot with the capability to autonomously map its surroundings, generate the optimal path, and navigate through the arena while avoiding any barriers or obstacles. The goal is to create a robot that is intelligent and efficient in its ability to navigate through various environments.

***Keywords*:-** *ROS, Gazebo, OpenCV, Microcontroller Programming, Solid-Works 3D Modelling, Deep Learning, and Digital Image Processing.*

I INTRODUCTION

ROS- ROS, also known as the Robot Operating System, is an open-source tool designed to assist researchers, developers, and scientists in building and sharing code between various robotics applications. It is part of a worldwide community of engineers, developers, and scientists who work towards improving the accessibility and functionality of robots for everyone. Modern-day robot systems are equipped with numerous sensors and controllers, making them complex hardware devices, which require intricate distributed software management. In order to facilitate the development of such systems, it is essential to simulate and visualize the robot's environment before it is used in the real world. ROS serves as a framework for such simulations, allowing for the development and testing of robotics software in a controlled environment.

CAD- Using computers to help with the creation, modification, analysis, or optimization of a design is known as computer-aided design (CAD). By using this software, designers can work more productively, create better designs, communicate more effectively through documents, and build databases for manufacturing. A common CAD output format file for printing and other manufacturing processes. Mechatronics systems can be controlled and modeled using CAD tools. Since the robots are mechatronic systems, simulating their behavior is possible thanks to their model's accurate dimensions.

Gazebo- A free and open-source 3D robotics simulator is called Gazebo. There are many strong physics engines available to Gazebo. It assists you in simulating real-world phenomena like gravity, friction, torques, and any other components that might have an impact on the outcome of your simulation. You can connect a variety of sensors with the aid of a Gazebo. You can evaluate these sensors using Gazebo and create robot designs that make the most of them.

OpenCV- A significant open-source library for computer vision, machine learning, and image processing is called OpenCV. It makes a significant contribution to the real-time operation, which is crucial in modern systems. In pictures and movies, it can be used to look for people, things, and even human handwriting. The OpenCV array structure can be handled by Python for analysis when combined with other libraries, such as NumPy. We use vector space and apply mathematical operations to these features to recognize visual patterns and their numerous features.

In this project, we intend to create our robot entirely from scratch and program it using this open-source software. Solid works will be used to design the robot's base chassis, the electronics will be connected, the microcontroller will be programmed, and then a ROS-based control system feedback loop will be used to drive the robot autonomously along its input path.

Although there are many different robots on the market, we wanted to make this one with the fewest resources possible without sacrificing performance. We think there are a number of crucial applications for our user-defined path-planning function that fall outside the purview of this project.

ISSN No:-2456-2165

II LITERATURE REVIEW

A. *Review of Previous Work in Autonomous Path Planning for Mobile Robots*

In conducting our research, we came across numerous papers and research projects that experiment with and propose various path-planning systems and mechanisms. The works listed below are a few of the ones we found intriguing and used in our own work.

In [1] The article is a brief study on path planning for a robot with three Omni wheels. In order to implement the Bezier curve tracing algorithm for path planning, the paper suggests using the built-in encoders of motors and an IMU (Inertial Measurement Unit). The authors recommend building a three-wheeled chassis with a 120-degree angle between each wheel and attaching it to a built-in hall encoder for measuring the robots' coordinates along the X and Y axes. The drive train has three degrees of freedom: ordinate, yaw, and abscissa. A gyroscopic sensor is also fastened to the chassis along with the encoders.

The model of the robot used in this work is described in [2] as a differential drive robot with two driving wheels and two caster wheels. The system architecture consists of the Gazebo, the ROS and Navigation Stack software, and sensor plugins. Tools for using the Build editor to create an environment are also available in Gazebo Simulator. Any robot can move independently in the environment once it has been mapped out. As a result, the suggested model uses less computational power while still producing comparable outcomes.

In [3], a ROS-based control application for a robotic platform employing the Gazebo 3D simulator will enable a user to observe and comprehend the simulation and movement of a mobile robot through a predefined map before being used in the real world. The dynamics of the robot must be put to the test, confirmed, and validated by developers in the field of robotics. These days, developers can make virtual workspaces and perform a focused simulation of robot dynamics thanks to the accessibility of virtual tools. The author of this paper discusses the methodology and experience of simulating a real mobile robot using Gazebo and ROS. The experiment was a success and gave information on how to improve the driving and orientation of the actual robot without exposing it to a dangerous environment, as well as its control, sensors, data acquisition, and potential danger tackling.

In [4], Omni wheels are wheels that can move in any direction and with or without changing the orientation. To enable motion parallel to the axis of the wheel, they have beads or rollers attached to the wheel. You can buy the wheels from any robotics store or online retailer. Speed and controllability must be balanced, so the right wheel size must be chosen. It is preferable to use a controller with an integrated hardware I2C circuit if the angle is taken using an Inter-Integrated Circuit (I2C).For the calculations necessary to drive and control the structure, any oscillator producing a clock above 1MHz will be sufficient. Due to its precise output and largely error-free data transfer, a gyroscope for angle feedback has proven to be very helpful. A robot with a holonomic drive can change its orientation without changing its motion, and vice versa. Due to the fact that doing so would require the robot to be aware of its surroundings even before it sets sail, it is challenging to calculate its coordinates with precision using laser sensors and other non-contact-based distance measurement techniques. Even though rotary encoders have the disadvantage of incremental error and the requirement for continuous operation, we have used them to completely avoid this scenario. However, these flaws are outweighed by its greatest asset, an absolute reference that is unaffected by changes in the environment.

In [5], Robotic Operating Software (ROS) is used to connect a Deep Learning model in the backend with a Raspberry pi based mobile robot. A faster R-CNN deep learning model is considered and robot capabilities in object detection are validated through experimental means. A kinetic sensor is used to click images which are then fed to the deep learning algorithm. The main aim of the project is to create service robots with advanced cognitive intelligence which can sense their surroundings and to implement this at as low a cost as possible. The proposed system consists of raspberry pi as the control kernel which runs ubuntu mate as the OS. The kinetic sensor captures photos and images from the surrounding environment which it then uploads to a SQL database. The saved images are then fed to a GPU-accelerated Faster R-CNN deep learning model running on a local computer. ROS, running on the local computer, supports a kinetic driver which connects to the kinetic sensor using Wi-Fi. Hence Using this process, the authors were able to achieve real-time object detection on a mobile robot with a detection probability of 99.7%.

In [6], we studied how to use CAD tools to use in our project. In this paper, the context is to compute the models required to simulate and control the six degrees-of-freedom GT6A robot arm. The author put forth the modeling, dynamic modeling, motion control, trajectory generation, and PID control interests in this. This study suggests using CAD designs of robots or other mechatronic systems to simulate them. The GT6A robot is the manipulator under study. By manipulating various objects, we can use the simulation technique to simulate the robot's movements in various settings and circumstances.

In [7], the author concentrates on an unsolved issue in the field of 3D object recognition. The third type of vision model building for the robot vision systems was made possible by this CAD system and robot vision system. Since a few years ago, using object representations of the kind that a CAD system might produce has been a popular research topic. The representation and identification of three-dimensional objects is challenging and a significant vision issue for robots. The author suggested shape features of objects, CAD models and PDES, feature recognition, and CAD representation of objects. The author attempts to automatically extract from a CAD database a high-level data structure form useful for robot vision in this approach to automated object interpretation. The feature recognizer's

ISSN No:-2456-2165

conceptual potential is limitless. A feature rule may use any measurable topological or geometrical property. The definition of primitive properties, which were then used to define holes, pockets, and straight and curved slots, required several hundred lines of utility rules. The feature definition rules are quite brief and specific once the utility rules have been developed. Benefits of applying production rules.

In [8], the author looks for more efficient ways to program Arduino microcontrollers for high-speed controls and operations. Although the conventional approach to using the high-level C language is user-friendly, it can occasionally be ineffective for critical applications that require high-speed control. Because of this, the author suggested a different approach that directly communicates with registers using more effective code, such as Bitwise C language. By using this technique, instructions are executed more quickly. To calculate the delay required to carry out the input/output commands, a straightforward setup was constructed. The Arduino Uno module's ATMega328P was programmed using two different methods. The C language is the first one. But it results in a protracted delay.

In extremely sensitive applications or high-speed controls, this could lead to serious issues. The Arduino Uno's port registers using bitwise operations coding was the second one. Although there is a small amount of time delay with this programming style compared to C-Language code for the same instruction, there is still some time delay. The Proposed demonstrated that programming in Bitwise operations is significantly more efficient than doing so in C. More pins can be programmed simultaneously, and the program size is also reduced. This study demonstrates that bitwise operations coding is a faster and more effective way to program Arduino. The Serial Wire Debug (SWD) interface on ARM microprocessors can be used to debug many microcontrollers that are based on them today.

There are many microcontrollers nowadays which are based on ARM microprocessors that can be debugged through the ARM's Serial Wire Debug (SWD) interface. Flashing targets is also done through it. In the paper [9], the author provided us with insights into the software and hardware underlying a multiplexer that would enable programming multiple targets of the same model and type without the need to create intricate configuration scripts. A hardware tool for SWD multiplexing, firmware for the hardware tool, support for Makefiles for multi-target programming, and a prototype system for testing were all suggested by the author. It might not be easy to program multiple microcontrollers using the SWD interface. This paper presents an easy-to-implement hardware circuit that aims to give developers a quick solution to get them started.

In [10], the researchers used Python 3.7 and OpenCV 4 to build the virtual simulation of Khepera IV as a line-following robot. The image of the path line is taken by the robot's camera during image pre-processing, and it is then cropped to a particular width and length of the frame. The computer program Kephera IV is capable of identifying the edge contour of any shapes or lines present in an image. Equation with a resolution of 375752 for determining the central or centroid line These data will be processed to produce the centerline parameter, which will dictate the action or movement of the robot.

The paper [11] outlines the steps for setting up an autonomous system with ROS and a TurtleBot 3. Early on in the project, a simulated environment was built using Gazebo, Ubuntu, and ROS. The actual construction and configuration of the robot followed. The project is currently in the phase of writing test code to make sure that the robot and the software are communicating properly. Following this phase, the project enters a phase of path planning and the development and/or testing of autonomous driving algorithms. In the following stage, sensor fusion will be used to enhance object avoidance and environmental stimulation. The project's final stage involved testing various algorithms on the robot and using machine learning to determine how well each approach performed given the project's current hardware requirements. The advantages of open-source software like ROS and simulators like Gazebo, which have facilitated entry into the field of robotics, are discussed in the paper. The adoption of autonomous robotic systems has been robotics' greatest accomplishment, and this paper describes how industrial autonomy has moved from factories to more advanced robots. The project's objective is to create a flexible academic base that can accommodate any testing requirement. It can be assumed that the project is still in progress and that the authors want to compare different path-planning algorithms and sensor reading methods even though no specific research gap is mentioned in the report. The research gap is probably in determining the best sensor fusion method and path planning algorithm for the TurtleBot 3. The authors also mention the use of machine learning to verify each algorithm's optimization with the project's current hardware needs. This subject may merit more research.

In paper [12], the suggested path planning algorithm combines map-based path planning and reactive path planning to design a safe and efficient path for mobile robots in changing environments. After the initial path is generated using map-based planning, reactive planning is used to change the path in real-time to avoid obstacles. The proposed algorithm provides mobile robots with a safe and efficient path while navigating dynamic obstacles. The algorithm is put to the test using simulations and actual experiments, and the results show that it works well. In the study, no particular research gap is identified. However, the ability of the proposed method to handle complex and dense situations might indicate a need for further study.

In [13] The authors simulated a mobile robot in a mapping and path-planning scenario using the ROS (Robot Operating System) framework. They used the g-mapping program to perform SLAM (Simultaneous Localization and Mapping) and create an environment map. For robot localization and path planning, the ROS navigation stack, which employs the Adaptive Monte Carlo Localization (AMCL) technique, was used. In the simulation study, a

laser range finder sensor attached to the robot was used to scan the surroundings and produce a map, which was then used for course planning. The paper provides a comprehensive analysis of the use of ROS packages for SLAM implementation and path planning.

It highlights the benefits of utilizing ROS for simulation, which enables efficient testing and validation of robot behavior in a variety of contexts. The authors also discuss the challenges of putting such systems into practice and the need for trustworthy and precise sensors for mapping and localization. The proposed method is not evaluated in a real-world setting; the study primarily focuses on simulation-based studies. Additionally, the authors do not quantitatively compare the proposed approach to other approaches, which could have improved the study's findings. More study is needed to validate the approach in practical settings and compare its performance to other approaches.

In the paper [14], the authors recommended plotting a dual-arm robot's trajectory using the ROS (Robot Operating System) framework. The dual-arm robot's streamlined kinematic model and the RRT (Rapidly-Exploring Random Tree) method are combined to create the trajectory planning. The ROS moveit! the package is used for motion planning and collision detection, and Gazebo, a 3D robot simulation environment, is used for simulation research.

The paper goes into great detail about the trajectory planning method for a dual-arm robot using the ROS framework. Effective and optimal motion planning for the dual-arm robot is crucial for performing complex manipulation tasks, and this is made possible by the RRT algorithm's integration with the moveit package. The advantages of using a Gazebo for simulation, which offers a realistic simulation environment and enables the testing of diverse robot behaviors, are also covered by the authors. The suggested trajectory planning technique is based on a dual-arm robot's reduced kinematic model, which might not be accurate enough for more intricate manipulation tasks. To determine whether the suggested strategy can be used for more complex cases, more study is required. Moreover, the study omits to compare the suggested strategy with current practices, which would have improved the study's conclusions.

In [15], The authors proposed a method for path planning for a tracked mobile robot. Path design, path optimization, and environment modeling are the three steps in the procedure. The authors used MATLAB for environment modeling and path planning; ROS (Robot Operating System) was used for path optimization. The optimization is done using the ROS navigation stack, which plans local paths using the Dynamic Window Approach (DWA) method. The A* algorithm was employed in this work as the path planning algorithm. The study on path planning for a tracked mobile robot in the paper is in-depth, and the suggested method works in a range of settings. It is possible to simulate the surroundings accurately and plan the path effectively by using MATLAB. The advantages of using ROS for path optimization are also covered by the authors because it permits the testing of various robot behaviors and the evaluation of the suggested strategy in various settings. More study is required to see whether more sophisticated path-planning algorithms can be applied because the proposed path-planning method is based on a simple approach employing the A* algorithm. The study's conclusions could have been strengthened if it had included a comparison of the suggested strategy with current approaches. Furthermore, the study is restricted to simulation-based experiments, and the suggested strategy has not been tested in a real-world setting.

In [16], in this paper, the author proposes the optimization of the heuristic function in the A* algorithm by using the Manhattan distance to reduce the number of turning points and extended nodes. In addition, the author also proposes the use of a cubic Bezier curve to smooth the total path obtained hence finally resulting in a totally improved A* algorithm. The Improved A* Algorithm has the advantage of providing a shorter and smoother optimized path when compared to the traditional approach. It also reduces the number of search nodes and transition points between the start and end point thus providing a much more effective path. The time taken by the improved path is more than the time taken for the traditional approach by almost 27%. This can be improved upon to provide a faster algorithm.

In [17], to achieve efficient path planning for robots the author proposes a 3-stage implementation method. The first step is to create a rasterized cost map of the entire surrounding area. The map is divided into grids (each grid is the size of the robot) and all the obstacles and boundary walls are given very high integer values. As the robot detects any new obstacle in its path the map is updated. The second step is a vision system on the robot for obstacle detection. A Naive Bayes classifier is used to locate the obstacles and map them, which is then used to update the cost map. Finally, the paper proposes the use of a neural network to calculate the best path for the robot. The actual cost weight from the current node to the state space is calculated as a third variable using CNN. For robot pose and posture transformation, the authors advise using a gyroscope. The proposed methodology reduces the error from calculations and the time lost due to the complexity of the environment. It uses about 70% fewer waypoints as compared to the original algorithm. The gait update frequency is also reduced by 30% which results in improved speed of path planning. The obstacles detected from the vision system of the robots should be a known category and characteristics. Hence obstacles of unknown categories cannot be detected.

In [18], The paper identifies the problem that in the conventional method of path planning, due to the noise in measurements from Laser Range finders and numerous small obstacles, the cost map of the planned path is filled with many blind spots. In the proposed method, the author uses RGB-D cameras to calculate blind spots. In this method, the RGB-D cameras first capture the surrounding,

ISSN No:-2456-2165

then apply a voxel grid filter, removes irrelevant filters like the point cloud filter passing through the ground, and finally it clusters the point cloud based on the distance between the points which is then used to calculate the boundary position of the blind spots. The robot uses this information to draw a circular area around the blind spot so as they maintain a safe distance and avoid a collision from any obstacle or human passing through its blind spot.

The author proves the effectiveness of the proposed methodology with the help of simulations which show that the robot using RGB-D camera along with the proposed velocity term in the Dynamic-Window formula was about 27.7% faster than the robot using just Laser finders and the conventional formula for path planning. No research gap is mentioned however due to the high cost of RGB-D cameras the total cost of the project should be very high and hence not viable for small-scale production.

In [19], To conduct experiments, the ROS package and the simulator Rviz are both used. The same methods were applied to nine laser-scanned maps of the same environment in order to test the effectiveness of optimal path planning on pre-built maps. Path length and planning time can be compared between path planning on prebuilt maps and G mapping maps. Currently, large maps need to be split up into more manageable sections.

In [20], This study proposes a laser SLAM-based path-planning algorithm for ROS autonomous robots. The positioning system chooses AMCL (Particle Filter) as the positioning method. The choice is made to create the first laser SLAM map using the Karto-SLAM algorithm. As the first planning outcome, navigation is carried out using the complete coverage path planning algorithm (Complete Coverage Path Planning). The proposed path full coverage algorithm in this paper can effectively cover the traverse area, according to experimental results. The precise make-up of the unknowable environment is utilized. Map services, robot communication, and simulation visualization are all completed by the design of the main ROS nodes.

In [21], The path planning problem for mobile robots and the SLAM problem are both examined in this paper, and a mathematical model of the SLAM problem is developed. A particle filter-based FastSLAM algorithm is used. The ROS node base link serves as a representation of the robot base. Update the laser scanner information and time information about the data using the sensor message that was sent by the sensor. Update the robot's position estimation, speed estimation, and covariance matrix using the Odom message the odometer sends. The issue of local path planning based on sensor data was researched in terms of path planning. The artificial potential field method was used to realize the dynamic obstacle avoidance of mobile robots. Dynamic obstacle path is accomplished by avoidance along the map's shortest route.

In [22], As a prototype, a differential robot with two caster wheels and a laser scanner was built. The created robot model was then managed by the ROS navigation system. The experiments demonstrate that the robot is capable of navigating the simulated environment on its own without the need to modify the mapping, localization, or navigation package settings in any particular way. Additional systems, such as a speed controller and an inertial sensor, are required in order to achieve accurate path tracking, fluid motion, and precise positioning.

III METHODOLOGY OF RESEARCH AND PROPOSAL

For the project we propose to build a robot chassis. The robot will have a holonomic drive, for more mobility and ease of movement. To create a holonomic drive the wheels of the robot will be kept at 120 degrees with each other. This ensures that the robot is free to move in any direction possible without having to turn its head toward the direction. the chassis will be triangular with each side half a meter long.

For the base motors, we plan to use high rpm and low torque motors for faster movement of the robot. These motors will be attached to rotary encoders to count steps and calculate the position of the robot. The encoders also help in the accurate positioning of the motors using the PID algorithm.

The robot will have a 360 Degree RPLiDAR with a ~6m radius range. The sensor will be used for mapping, localization, and object/environment modeling.

To drive the motors, we are planning to use the L293D dual H-Bridge motor driver which allows speed and direction control of two DC motors at the same time and a 12V Lithium Polymer Battery. We can use an Arduino UNO Microcontroller to control the motor outputs. We are also planning on using a Raspberry Pi micro-processing board with the ubuntu mate operating system that can connect with a local computer for input reading.

The local computer will be running ROS and use the positional inputs from the onboard Raspberry Pi to calculate the position of the robot. ROS will also estimate the best positional coordinates that the robot should move to complete its goal. ROS will accordingly give the new set of motor inputs to move the robot in that particular direction.

Before implementing the Python code on the robot using ROS, we are planning on simulating the results using the Gazebo simulator. Gazebo simulation requires a CAD file of the robot and it can simulate the input code in a real-time environment to help us get a better understanding of any shortcomings and mistakes.

IV METHOD OF IMPLEMENTATION

Here are the steps for implementing path planning for mobile robots using ROS and its simulation in the Gazebo environment that we are planning to implement:

ISSN No:-2456-2165

- Install ROS and Gazebo: Firstly, you need to install ROS and Gazebo on your system if they are not already installed.
- Create a workspace: Create a new ROS workspace where you will create your ROS packages.
- Create a package: Create a new ROS package for your project. You can use the command catkin_create_pkg to create a new package.
- Add dependencies: Add the required ROS and Gazebo dependencies to your package. This includes the rospy, std_msgs, nav_msgs, tf, and gazebo_ros packages.
- Create a robot model: Create a robot model for your mobile robot in Gazebo. We are planning to use the Gazebo Model Editor to create the robot model.
- Write launch files: Write launch files for your project. A launch file is a file that launches multiple ROS nodes simultaneously. You will need to launch the Gazebo simulation, the robot control node, and the path planning node.
- Write a robot control node: Write a ROS node that controls the movement of the robot. This node will subscribe to the robot's current position and orientation and publish new velocity commands to move the robot.
- Write a path planning node: Write a ROS node that performs path planning for the robot. This node will take the current position of the robot and the goal position as inputs and output a sequence of waypoints that the robot should follow to reach the goal.
- We are planning to implement A* algorithm to find the optimal shortest path between two nodes. Since A* algorithm uses a heuristic approach to estimate the best route through that node we trust that A* will provide the most complete and optimal solution. The final cost "f(n)" for travelling nodes is denoted as:
- $f(n) = g(n) + h(n)$
- where,
- $g(n)$ = cost of travelling from one node to another,
- $h(n)$ = heuristic approximation of the node's value.
- Test the simulation: Launch the Gazebo simulation and test the path planning algorithm by setting a goal position for the robot and observing its movement.
- Refine the path planning algorithm: Refine the path planning algorithm based on the results of the simulation.

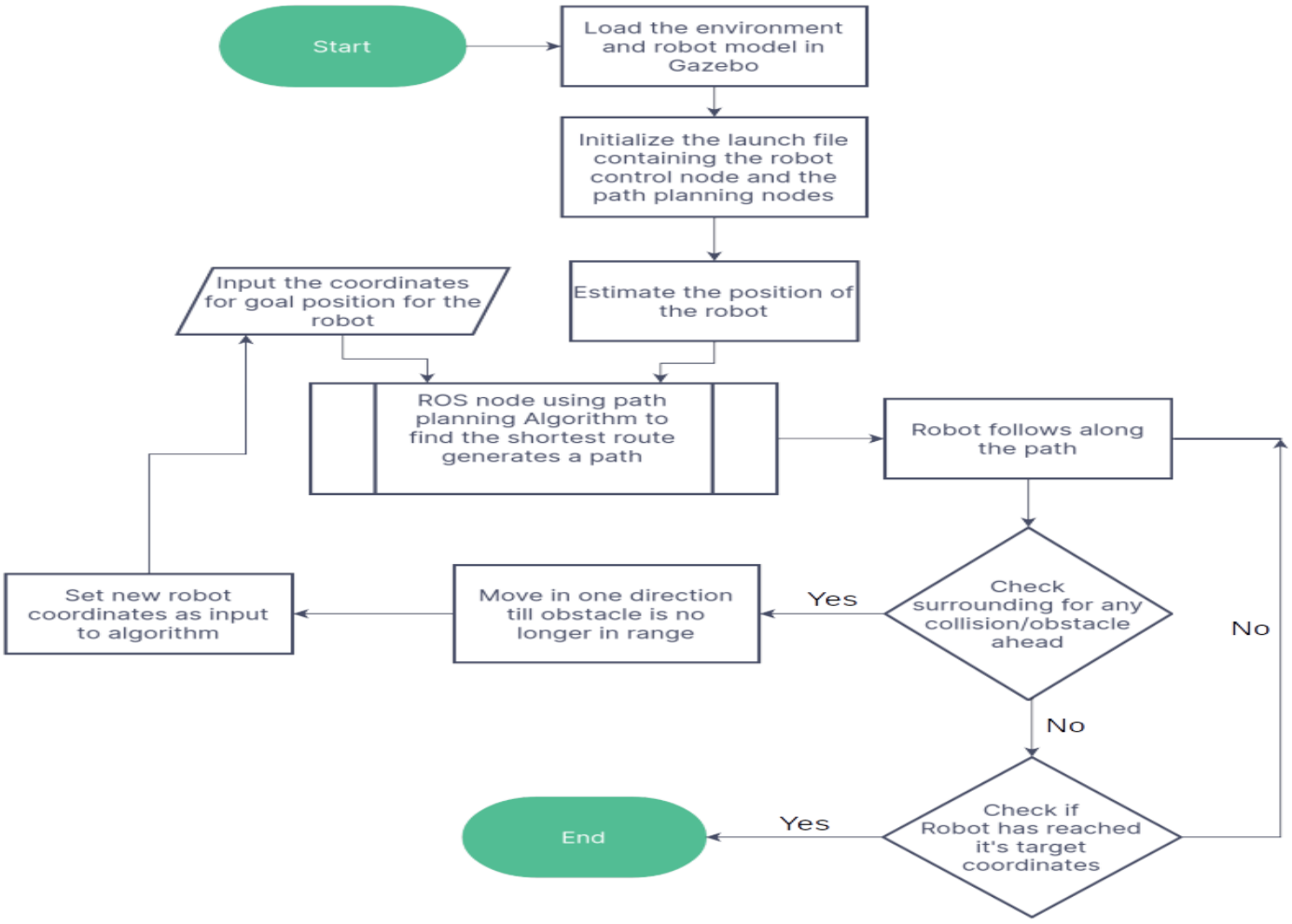

Fig 1 Flowchart for the process

V CONCLUSION AND FUTURE WORK

In this study, we proposed to build an autonomous robot that can travel and given path and draw any given user input on the arena. We hope this technology and our research work will be used for the betterment of the different classes of robots available in medicine, search and rescue, delivery, and many more fields.

In the future, we will study more machine learning algorithms and deep learning algorithms so that we can produce output with more accuracy.

➢ *Acknowledgment*

We would like to thank Mr. Mandar Bivalkar, our project guide for her help with the project and the paper.

REFERENCES

[1]. M. Gavani, D. Tanpure and P. Falake, "Path Planning of Three Wheeled Omni-Directional Robot Using Bezier Curve Tracing Technique and PID control Algorithm," 2019 IEEE Pune Section International Conference (PuneCon), 2019, pp. 1-6, doi: 10.1109/PuneCon46936.2019.9105899.

[2]. R. K. Megalingam, A. Rajendraprasad and S. K. Manoharan, "Comparison of Planned Path and Travelled Path Using ROS Navigation Stack," 2020 International Conference for Emerging Technology (INCET), 2020, pp. 1-6, doi: 10.1109/INCET49848.2020.9154132.

[3]. M. Marian, F. Stîngă, M. -T. Georgescu, H. Roibu, D. Popescu and F. Manta, "A ROS-based Control Application for a Robotic Platform Using the Gazebo 3D Simulator," 2020 21th International Carpathian Control Conference (ICCC), 2020, pp. 1-5, doi: 10.1109/ICCC49264.2020.9257256.

[4]. D. Tak, A. Jain, P. S. Savnani and D. Akash Mecwan, "Path Tracing in Holonomic Drive System with Reduced Overshoot using Rotary Encoders," 2020 7th International Conference on Signal Processing and Integrated Networks (SPIN), 2020, pp. 343-348, doi: 10.1109/SPIN48934.2020.9071196.

[5]. Y. -H. Chang, P. -L. Chung and H. -W. Lin, "Deep learning for object identification in ROS-based mobile robots," 2018 IEEE International Conference on Applied System Invention (ICASI), 2018, pp. 66-69, doi: 10.1109/ICASI.2018.8394348.

[6]. F. Boujnah and J. Knani, "Motion simulation of a manipulator robot modeled by a CAD software," 2015 7th International Conference on Modelling, Identification and Control (ICMIC), 2015, pp. 1-6, doi: 10.1109/ICMIC.2015.7409442.

[7]. Tian Jie, Tai Juwei and Gao Zhengyu, "CAD-based robot vision: from CAD models to vision models," [1992] Proceedings of the IEEE International Symposium on Industrial Electronics, 1992, pp. 362-365 vol.1, doi: 10.1109/ISIE.1992.279552.

[8]. M. M. M. Asad, I. A. Marouf and H. M. Enshasy, "An effective way to program microcontrollers for high-speed control operations," 2017 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS), 2017, pp. 1-4, doi: 10.1109/ITCOSP.2017.8303166.

[9]. L. Bogdanov, "Multiple Microcontroller Programming Using the SWD Interface," 2020 XXIX International Scientific Conference Electronics (ET), 2020, pp. 1-4, doi: 10.1109/ET50336.2020.9238282.

[10]. A. Ma'arif, A. A. Nuryono and Iswanto, "Vision-Based Line Following Robot in Webots," 2020 FORTEI-International Conference on Electrical Engineering (FORTEI-ICEE), 2020, pp. 24-28, doi: 10.1109/FORTEI-ICEE50915.2020.9249943

[11]. S. Pietrzik and B. Chandrasekaran, "Testing Autonomous Path Planning Algorithms and Setup for Robotic Vehicle Navigation," 2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 2018, pp. 485-488, doi: 10.1109/UEMCON.2018.8796525.

[12]. Z. Cheng, B. Li, and B. Liu, "Research on Path Planning of Mobile Robot Based on Dynamic Environment," 2022 IEEE International Conference on Mechatronics and Automation (ICMA), Guilin, Guangxi, China, 2022, pp. 140-145, doi: 10.1109/ICMA54519.2022.9856220.

[13]. J. Cheng, L. Zhu, X. Cai, and H. Wu, "Mapping and Path Planning Simulation of Mobile Robot Slam Based on Ros," 2022 International Seminar on Computer Science and Engineering Technology (SCSET), Indianapolis, IN, USA, 2022, pp. 10-14, doi: 10.1109/SCSET55041.2022.00012.

[14]. Y. Cong, C. Jiang, H. Liu, H. Du, Y. Gan and C. Jiang, "Research on trajectory planning method of dual-arm robot based on ROS," 2020 Chinese Automation Congress (CAC), Shanghai, China, 2020, pp. 2616-2621, doi: 10.1109/CAC51589.2020.9327351.

[15]. S. Fu, C. Zhang, W. Zhang and X. Niu, "Design and Simulation of Tracked Mobile Robot Path Planning," 2021 IEEE 4th International Conference on Big Data and Artificial Intelligence (BDAI), Qingdao, China, 2021, pp. 86-90, doi: 10.1109/BDAI52447.2021.9515251.

[16]. J. Gao, X. Xu, X. Zhang, S. Xu, and Q. Pu, "Path Planning of Mobile Robot Based on Improved A* Algorithm," 2022 5th World Conference on Mechanical Engineering and Intelligent Manufacturing (WCMEIM), Ma'anshan, China, 2022, pp. 1098-1102, doi: 10.1109/WCMEIM56910.2022.10021353.

[17]. W. Jingyao and Y. Naigong, "Universal Path Planning Based on Naive Bayes Classifier and Improved A* Algorithm using CNN," 2022 41st Chinese Control Conference (CCC), Hefei, China, 2022, pp. 7030-7035, doi: 10.23919/CCC55666.2022.9902424.

[18]. M. Kobayashi and N. Motoi, "Path Planning Method Considering Blind Spots Based on ROS Navigation Stack and Dynamic Window Approach for Wheeled Mobile Robot," 2022 International Power Electronics Conference (IPEC-Himeji 2022- ECCE Asia), Himeji, Japan, 2022, pp. 274-279, doi: 10.23919/IPEC-Himeji2022-ECCE53331.2022.9807203.

[19]. H. Mukhtar, M. A. Hasan, and M. U. G. Khan, "ROS-Based Global Path Planning for Autonomous Ground Robot Using the Pre-Built Map of the Environment," 2021 International Conference on Robotics and Automation in Industry (ICRAI), Rawalpindi, Pakistan, 2021, pp. 1-8, doi: 10.1109/ICRAI54018.2021.9651374.

[20]. S. Zhao and S. -H. Hwang, "Path planning of ROS autonomous robot based on 2D lidar-based SLAM," 2021 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Korea, Republic of, 2021, pp. 1870-1872, doi: 10.1109/ICTC52510.2021.9620783.

[21]. Z. Liu, "Implementation of SLAM and path planning for mobile robots under ROS framework," 2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP), Xi'an, China, 2021, pp. 1096-1100, doi: 10.1109/ICSP51882.2021.9408882.

[22]. D. Chikurtev, "Mobile Robot Simulation and Navigation in ROS and Gazebo," 2020 International Conference Automatics and Informatics (ICAI), Varna, Bulgaria, 2020, pp. 1-6, doi: 10.1109/ICAI50593.2020.9311330.