Autonomous Indoor Mapping Robot using ROS

Nishad Milind Rajhans
Department of Mechanical engineering
MITSOES
MIT Art, Design & Technology University
Pune,Maharashtra
nishad07112000@gmail.com

Ayush Giri
Department of Mechanical engineering
MITSOES
MIT Art, Design & Technology University
Pune,Maharashtra
aayushg2001@gmail.com

Kalpesh Kolte
Department of Mechanical engineering
MITSOES
MIT Art, Design & Technology University
Pune,Maharashtra
kalpeshkolte1@gmail.com

Gufran Momin
Department of Mechanical engineering
MITSOES
MIT Art, Design & Technology University
Pune,Maharashtra
gufranmomin1234@gmail.com

Bhumeshwar Patle
Department of Mechanical engineering
MITSOES
MIT Art, Design & Technology University
Pune,Maharashtra
balu_patle@rediffmail.com

Praveen Kumar Bhojane
Department of Mechanical engineering
MESCOE
MES Wadia College of Engineering
Pune,Maharashtra
praveenkumar.bhojane@mituniversity.edu.in

***Abstract*—The concept of mobile robots has always been the prime topic of interest among the community of the roboticist However, the idea of engineering a mobile robots robot that can display indoor mapping and navigation, The current robots that we find are made for only a particular task orientation and are not user friendly as well as they also lack the ability of flexibility in the development of the robot, The purpose of this research thus involves developing and fabrication of the highly user-friendly and open source flexible interface of ROS(Robotic Operating System) which can integrate a wide range of sensors and can perform various operation as well as functions very efficiently and it takes the shortest time to cover (i.e.) Path planning, In the indoor environment, It also is the combination of A* algorithm into the robot automation firmware using obstacle avoidance, and thus every decision is unique and optimized. The experimental and simulation results are validated here to show the effectiveness of the A* algorithm.**

Keywords—UGVs, ROS (Robotic Operating System), Path planning, mobile robot navigation, obstacle avoidance, indoor mapping, A* algorithm

I. INTRODUCTION

Robotics constitutes a significant subfield within the discipline of mechatronics engineering. Robotics encompasses various aspects, including the strategic formulation, conceptualization, physical realisation, functional execution, and practical use of robotic systems. A robot is a machine, specifically one that can be programmed by a computer to execute a set of intricate activities autonomously. In the manufacturing sector, robots are commonly employed for various tasks such as the creation, finishing, transfer, and assembly of components. Various forms of robots exist, including drones. The six most prevalent categories of robots are autonomous mobile robots (AMRs) [1], automated guided vehicles (AGVs) [2], articulated robots, humanoids, industrial robots, and hybrids. Robotics is an engineering discipline concerned with the development, design, and operation of robots for the purpose of automation.

It is also associated with the study and implementation of robotics, encompassing the construction of robots through the integration of diverse technologies and their utilisation in our daily activities. The field of robotics presents significant prospects for future advancements. Robotic systems find extensive utilisation across diverse domains, serving a multitude of objectives. Presently, their deployment is particularly prominent in hazardous settings, encompassing tasks such as inspecting radioactive substances, detecting and neutralising explosive devices, as well as facilitating manufacturing operations. Additionally, robots are employed in environments that are inhospitable to human presence, such as outer space, underwater realms, extreme heat conditions, and the management and confinement of perilous materials and radiation. Robots possess the capacity to assume various physical manifestations, with certain models specifically designed to closely resemble the human form. This facilitates the integration of robots in specific tasks that typically involve human-like replication. These robots endeavour to imitate many human activities [2].

A. Challenges and Motivation

With the due research done, we came to know there is not much evidence of using the A* algorithm in the robot with ROS and implementing that in the mobile robot navigation the main objective is to optimize path length and navigational time and find the shortest distance, This was a challenging task to implement (Robotic operating system) with the A* algorithm because the method was unknown and we had to find many different innovative ways to accomplish the task and we are successful in making that, There are various parameter for the A* algorithm to inculcate in our research.

The 1st section contains information on robotics, its uses, and its application, the 2nd section contains and information about the navigation and mapping of the mobile robot including the self-localization and mapping. The 3rd section of the research paper consists of the path planning of the robot using the A* algorithm and integrated with the ROS (Robotic Operating System) and the detailed information on the robot construction with the software used in the simulation it also consists which path planning algorithm (A* algorithm) used in the indoor environment. 4th section consists of the simulation and analysis of the robot in the different indoor environments in using A* path planning and ROS architecture with different parameters are taken into consideration having graphical and statistical data. 5th section consists of the simulation results during the simulation of the robot in the indoor environment using ROS and the A* algorithm. 6th section consists of all the research papers that we have gone through and had some useful insights during our research.

II. MOBILE ROBOT NAVIGATION

The field of autonomous mobile robotics places significant emphasis on control systems and navigation as primary areas of concern. The system has components for hardware circuit design, control software, and upper computer software. The velocity and current control of DC motors have been observed in lesser computer systems. The navigation pack holds significant importance and possesses considerable power inside the framework of the ROS (Robotic Operating architecture) architecture. The navigation system employed in the autonomous car [3]. As depicted in Figure 1, the comprehensive structure of the navigation and Path planning involves determining the most efficient route considering the surrounding environment and the presence of obstacles that the robot must navigate. There are two primary categories of obstacles: static obstacles, which remain stationary, and dynamic obstacles, which are subject to movement. Upper computer interaction primarily facilitates human interaction, machine interaction, remote control, and data communication. The navigation stack operates at a basic conceptual level by receiving input from the odometry and sensor streams.

In order to ensure accurate navigation, several prerequisites must be met for the mobile base to receive appropriate velocity commands. Firstly, the robot must be utilising the ROS software framework. Additionally, the tf transform tree must be properly established. Furthermore, the publisher data should be transmitting the correct messages. Lastly, the ROS configuration should be tailored to accommodate the shape and dynamics of the robot, enabling it to operate at an optimal level. The hardware requirements of the navigation stack are specifically designed for wheeled robots with differential drives. It implies that the mobile base may be controlled by transmitting velocity signals. Additionally, the navigation stack necessitates the presence of a planar laser, which should be positioned on the mobile base. The laser in question is utilised for the purposes of map construction and localization. The navigation stack was originally devised for implementation on a square robot, therefore yielding optimal performance on robots of all shapes and sizes. However, while employing this navigation stack on larger rectangular robots within confined areas such as entrances, certain limitations may arise [3].

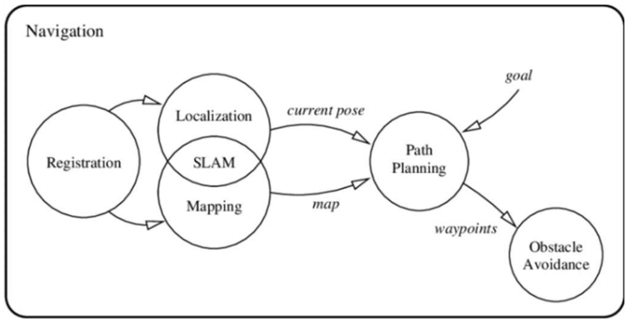


Fig 1 Navigation Stack

A. *Self-localization*

The expeditious execution of a comprehensive methodology for a robotic system to achieve self-localization within an indoor setting that may be represented as a basic polygon. As depicted in Figure 2, the robot's sole source of information consists of a polygon map and sensor data obtained from a range detecting device. It is assumed that in this manner, the robot possesses access to its localised visibility polygon. The iterative approach we employ involves consistently moving towards the nearest point where the robot can ascertain the elimination of at least one potential location in which it may be situated [4].

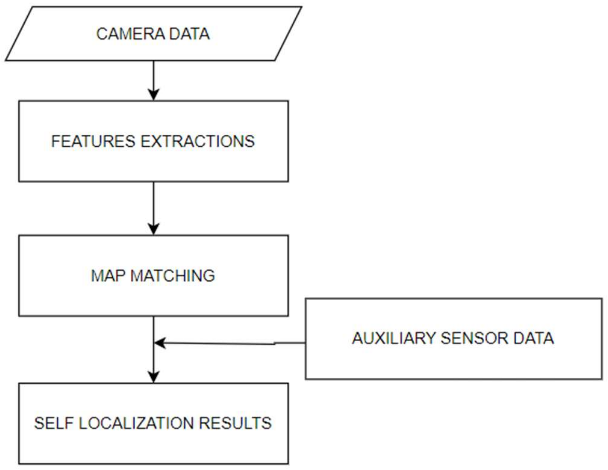


Fig 2 Self Localization

B. *Map Building and Map Interpretation*

The process of mapping the mobile robot is a fundamental aspect of achieving effective navigation in the field of mobile platform technology. Figure 3 illustrates that localization is a fundamental and crucial undertaking for attaining a high degree of autonomy in robot navigation and ensuring resilience in vehicle positioning. The field of robotic mapping and map interpretation is closely associated with cartography, employing techniques and computational methods to construct trajectory maps that accurately represent reality and effectively convey spatial information. [4].

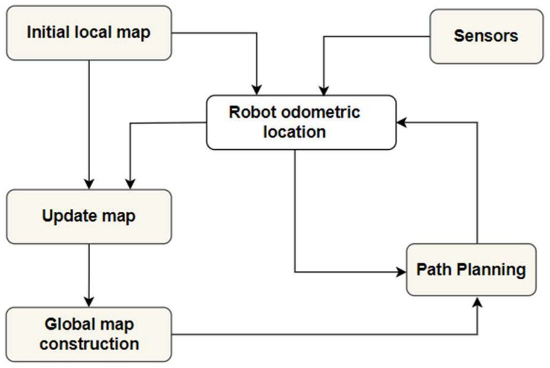


Fig 3 Robot Mapping

III. PATH PLANNING OF MOBILE ROBOTS USING A* AND ROS

Path planning, also called motion planning, is a computational problem that involves determining a set of feasible configurations to move an object between two locations. The objective of a path-planning algorithm is to determine a geometric trajectory that links the robot's present position to the desired destination, utilising a given map. Moreover, mobile robots working in unorganised settings or service and companion machines often lack comprehensive or complete prior understanding of the setting. Additionally, the context in which these robots operate is not static, meaning that while in motion, the robot may come across other robots, human beings, or companion animals.

Consequently, its performance of tasks is frequently influenced by unpredictability. Local obstacle handling, which includes obstacle detection and avoidance, is also required to achieve collision-free path planning. Robots may now detour around barriers utilizing modern approaches by quantitative measurement of the dimensions of obstacles [5]. In order to simulate the robot, the proposed algorithms were implemented in the Robot Operation System (ROS) as shown in the (Fig.4) the An free to use, meta-operating system designed for robotic platforms. The operating system, or OS, offers a range of services that are typically anticipated, such as hardware conceptualization, control over low-level devices, implementation of frequently utilised functionalities, inter-process communication through message-passing, and managing packages. More importantly, ROS (Robotic Operating System) has plenty of open-source packages including sensor drivers, navigation tools, environment mapping tools, path planning tools, communication and visualization tools, and many others that ultimately rigidifies the robot software network. The robot has a 360 Lidar module which can sense and give proper instructions to the micro-controller of that of the obstacle within the alarming range of the robot it also has a Bluetooth module for the communication of the instructions that the robot needs to accomplish and it has the encoder. The unmanned vehicle autopilot software suite in the Gazebo environment [5], the robot receives its position from the LiDAR module that connects to ROS(robotic operating system) for continuous movement along its x, y axes. In a SITL simulation, the desired path runs on the computer (either on the same computer or another computer on the same network). Sensor data is observed on the computer from the vehicle dynamics model in the simulator during SITL operations [6].

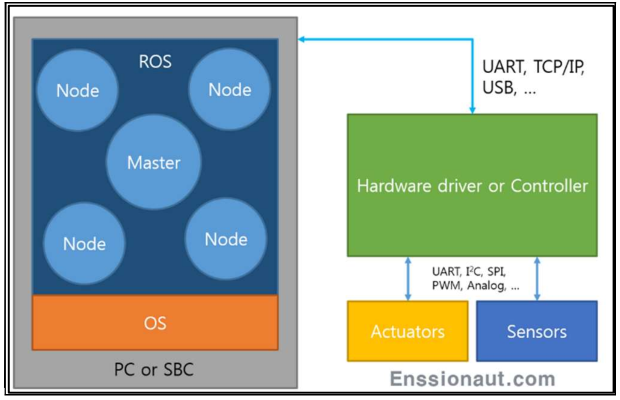


Fig 4 ROS-Robot Framework

A. *Robot Construction*

Mobile robots are required to know their locations within the environment as well as their surroundings so that they can perform assigned tasks. These issues are investigated within the context of localization and mapping, a phenomenon in robotics that analyses the world around a mobile robot. As shown in (Fig.5) The method is implemented in software that runs on the(robotic operating system) ROS platform. by using the stereo depth sensor on the robot, a point cloud about the obstacles is obtained for simulation and its applications the robot is equipped with various sensors for obstacle identification. the robot is equipped with 360 LiDAR shown in (Fig.6). This method involves the utilisation of remote sensing techniques, wherein the surrounding environment is subjected to scanning through the emission of a pulsed laser beam. Subsequently, the time taken for the reflected signal from the object to reach the detector is measured. Lidar sensors possess the ability to identify obstacles throughout a broad range of visual perception, rendering them highly suitable for integration into a comprehensive sense and avoid framework. Additionally, ultrasonic sonic sensors and a depth camera are also used to determine the presence of obstacles and its data will be used to plan the optimal path. A number of filter operations are used to convert this data. The robot receives the user's intended destination information, which is used in conjunction with the drone's position and map information to construct the intended flight path [6].

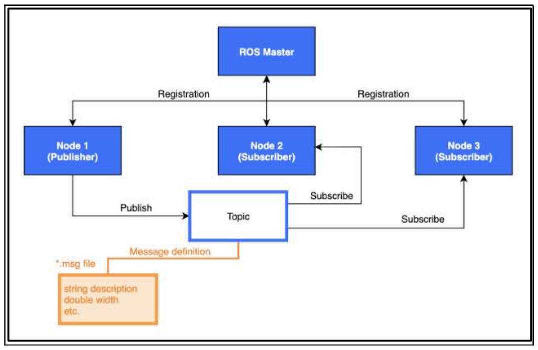


Fig. 5 Robot Sensors Publish and Subscribe pattern

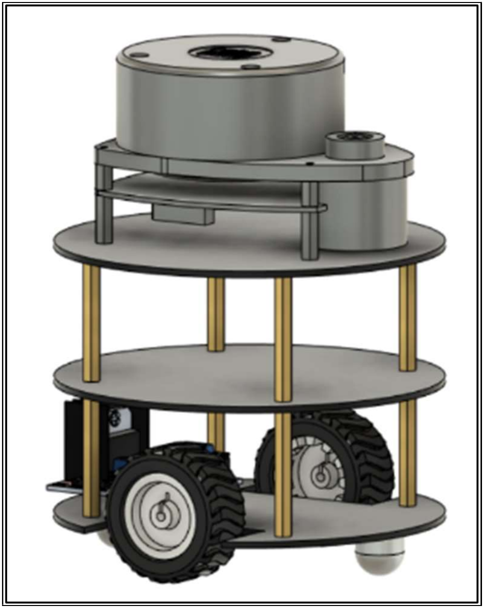


Fig 6 Robot Design

B. *ROS (robotic operating system)*

The Robot Operating System (ROS) comprises a collection of software libraries and tools that facilitate the development of robot applications. ROS is an open-source, meta-operating system designed for robots, encompassing a wide range of components like as drivers, state-of-the-art algorithms, and a robust development community. The operating system offers a range of services and resources, which encompass hardware abstraction as illustrated in the block diagram (Figure 7), implementation of frequently utilised functionalities, inter-process communication through message-passing, and package management. Additionally, it provides a diverse array of tools and libraries for the acquisition, construction, composition, and execution of code across many computational devices. The Robot Operating System (ROS) incorporates various modes of communication, encompassing synchronous Remote Procedure Call (RPC)-style communication through its service offerings, asynchronous streaming of data via topics, and data storage on a Parameter Server. The major objective of the Robot Operating System (ROS) is to facilitate the reuse of code in the field of robotics research and development. The Robot Operating System (ROS) is a distributed framework consisting of a collection of processes referred to as Nodes. These Nodes allow for the creation of executables that can be created independently and have loose coupling during runtime. The aforementioned processes can be categorised into Packages and Stacks, facilitating their seamless sharing and distribution. Currently, the ROS framework is exclusively accessible on operating systems that are Unix-based. The software designed for the Robot Operating System (ROS) is consistently subjected to testing on Ubuntu and Mac OS X operating systems. However, it is worth noting that the ROS community has actively contributed to extending support for more Linux platforms like as Fedora, Gentoo, Arch Linux, and others. [6].

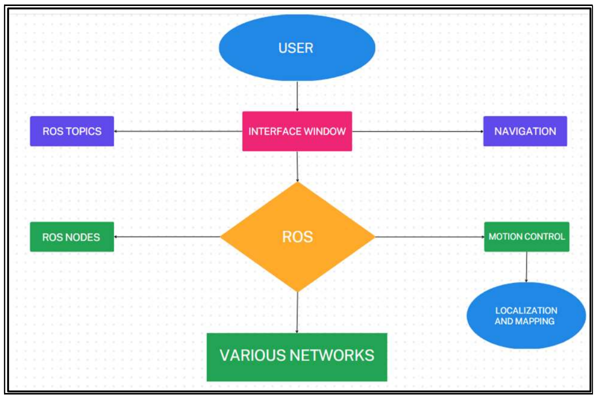


Fig 7 ROS Interface and Architecture

1) *Packages:* Packages play a crucial role in the organisation and structuring of software within the Robot Operating System (ROS). A package encompasses ROS runtime processes (referred to as nodes), a library that is dependent on ROS, datasets, configuration files, or any other components that are effectively organised in conjunction. Packages are the most desirable build item and release item present in ROS framework.

2) *Nodes*: ROS nodes are essentially just processes that are communicating with the robot through the use of ROS application programming interfaces (APIs). It's possible for a robot to have a lot of nodes to help it with its computations. ROS client libraries, such as roscpp and raspy, which will be covered in the next sections, allow us to establish ROS nodes. In the following parts, we will discuss:.

3) *Topics:* ROS topics are one of the ways in which two ROS Department of Mechanical Engineering nodes can communicate with one another and exchange ROS messages with one another. ROS messages are used to communicate between participants on designated buses known as topics.

In the search for the optimal course, the academic literature presents a great deal of different approaches. Every obstacle in the planning process requires a series of decisions that must be carried out in sequence throughout the course of time. In addition, it is absolutely necessary to specify in the planning formulation how the state shifts as activities are carried out. Each step of the path-planning procedure takes into account both the initial state and the destination state [7]. In many cases, there are two distinct categories of planning issues. The first factor to consider is whether or not the goal can actually be accomplished. To do this, you must devise a plan that, regardless of how effectively it works, will bring the robot to the desired destination. The second objective is to devise a plan that is both practicable and effective in enhancing performance in a particular manner. On the other hand, the effectiveness of these algorithms was evaluated according to the following ideal criteria: time required for computation and distance travelled [8].

In this work, we have used A* algorithm below is the general working and the basics of the A* algorithm and this is one of the best path planning algorithms that produced very satisfactory results.

a) *A* Search Algorithm:* There is a well-known and fundamental heuristic method called a star search (A*, A-star, or A* search [8]. Methodically, (Fig.8) the algorithm that the A* uses for its optimization. It is attempted to minimize the function, formalized as $f(n) = g(n) + h(n)$ taking into account the links between nodes and edges. In mathematical terms, $g(n)$ is the cost of the beginning point or node, while $h(n)$ is the cost of the remaining journey. $h(n)$ hereby constitutes the heuristic base of the algorithm [8].

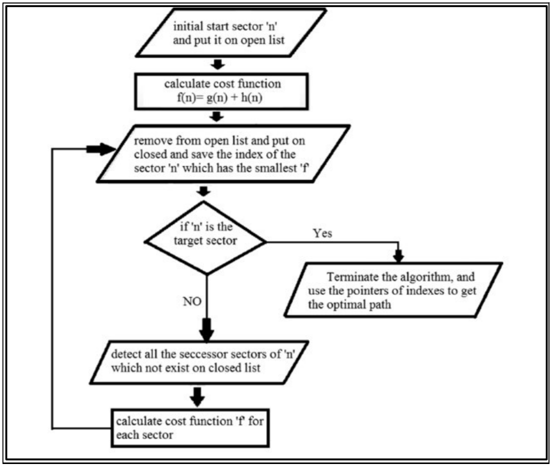

initial start sector 'n' and put it on open list
calculate cost function f(n)= g(n) + h(n)
remove from open list and put on closed and save the index of the sector 'n' which has the smallest 'f'
if 'n' is the target sector
Yes
Terminate the algorithm, and use the pointers of indexes to get the optimal path
NO
detect all the seccessor sectors of 'n' which not exist on closed list
calculate cost function 'f' for each sector

Fig. 8 The A* Algorithm flow chart

TABLE I. A* ALGORITHM PARAMETERS IN SIMULATION

Sr No	ALGORITHM PARAMETERS	
1	α - Safety Distance	0.8m
2	β - Safety Distance	0.7m
3	Velocity safety Distance	1.2m
4	Critical Safety Distance	0.07m
5	Max velocity	0.4m/s
6	Octomap resolution	0.1m
7	Spherical matrix resolution	6 degrees

The simulation that is performed in the indoor environment parameter used for the A* parameter is shown in (Table No 1). The UGV parameters of the performed simulation are shown in (Table No 2), The results of the simulation of the Special Case 1 are shown in (Table No 3), The results of the simulation of the Special Case 2 are shown in (table No 4).

TABLE 2 UGV PARAMETERS FOR THE SIMULATION

Sr No	UGV PARAMETERS	
1	Mass	2kg
2	Radius	35 cm
3	Inertia, ixx	0.0348
5	Inertia, iyy	0.0459

IV. SIMULATION AND ANALYSIS

After getting a broad grasp of how these algorithms function theoretically, it's crucial to put them to the test on a real robot to determine their effectiveness and usefulness. As shown in (Fig.9) the top view of the navigation and mapping sensors ROS (robotic operating system) includes a real-time physics simulator environment called Gazebo[9], allowing a robot model to be thoroughly evaluated before a prototype is built. In addition, the robot model is integrated into the Gazebo using a technique known as 'Software in the Loop,' or SITL[10], which feeds real-life robot data to the physical environment for simulation.

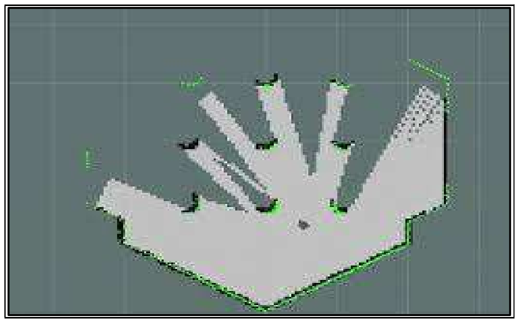
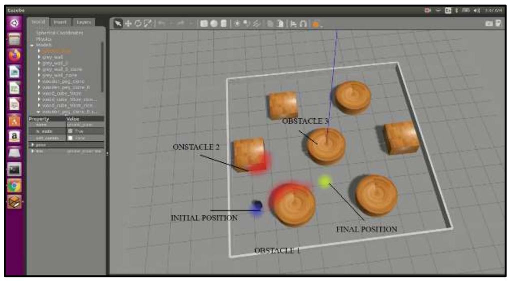
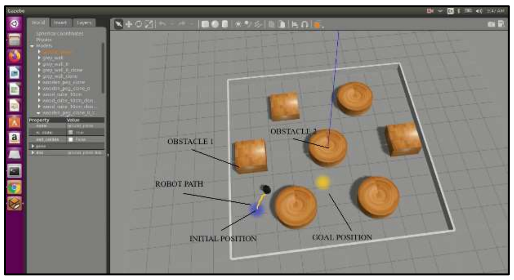


Fig 9 Robot sensors Navigation and Mapping

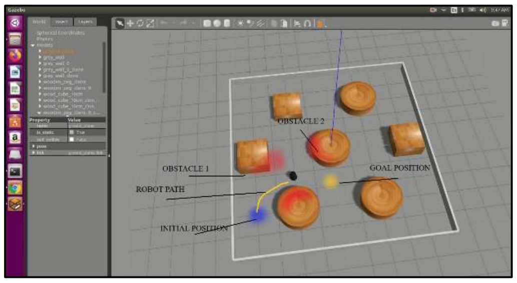


(a)

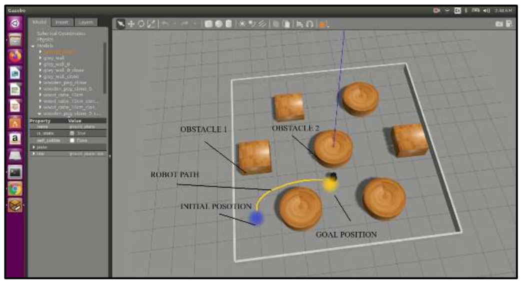
Special Case 1: The robot in the room with the Gazebo closed environment and with some static obstacles and the robot is situated in a particular position here the robot navigates through the obstacles and intelligent path planning is used to navigate the robot to find the most efficient path which has less cost in covering the distance, The (Fig.10) depicts the navigation mapping and the path taken by the robot. Here we give the robot a particular target position and the robot first determines the initial position of the robot in the room and analyses the target position given to the robot through which it navigates and reaches its given destination in the most efficient way possible to reach the point of target. (Table.3) gives the coordinates of the initial position and goal position of the robot with its navigational time taken by the robot and the path length achieved by the robot. The graphical representation of the statical data of particular case 1 is shown in (Graph.1) performed in the indoor test environment.



(b)



(c)



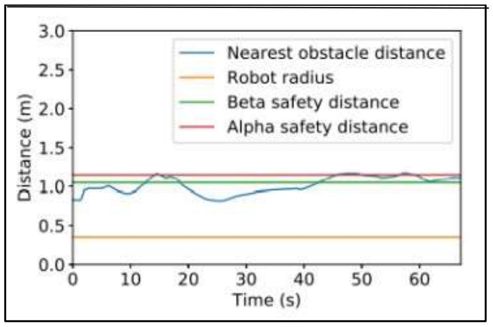
(d)

Fig 10 Test scenarios created in Gazebo simulator

Fig (a) The robot initiates the action by analyzing the initial position of the robot. Fig.(b) The robot starts the path planning and acts according to the set path planned by the software. Fig.(c) The robot is in the given path and detects the object and avoids the object and moves around. Fig. (d) The robot reaches the final destination.

TABLE NO 3-PATH LENGTH AND NAVIGATIONAL TIME USING A*ALGORITHM

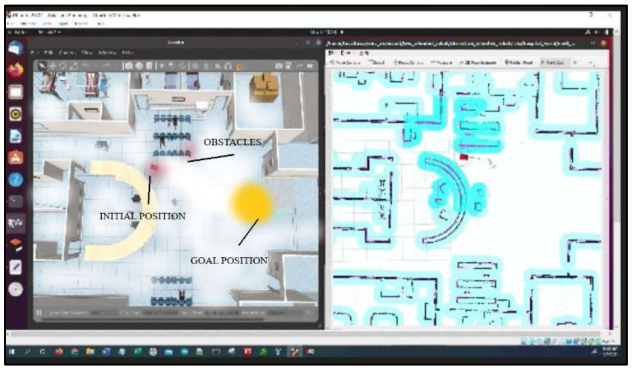
Sn No :	**Initial position (x, y)**	**goal position (x, y)**	**Navigational time (s)**	**Average path Length (m)**
1	-10; -10	-2.1; -1.35	1.72	8.5950
2	-10; -10	-2.3; -1.96	1.70	9.5950
3	-10; -10	-2.2; -1.21	1.69	8.1056
4	-10; -10	-2.2; -1.36	1.66	8.2369
5	-10; -10	-2.0; -1.54	1.74	9.1256
		∑	1.702	8.73162



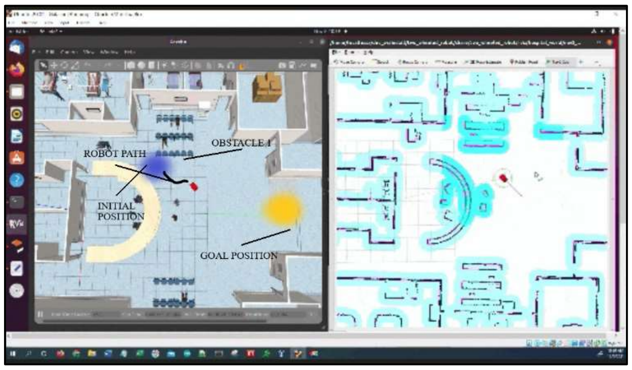
Graph 1 of Path Planning using A* algorithm in case1

TABLE NO 4 PATH LENGTH AND NAVIGATIONAL TIME USING A* ALGORITHM

Sr No:	Initial Position (x, y)	Goal Position (x, y)	Navigational time (s)	Average path Length (m)
1	-5, 10	-3; -2.5	2.73	15.5873
2	-5, 10	-2.9; -2.5	2.78	15.6669
3	-5, 10	-3; -2.72	2.69	15.6988
4	-5, 10	-3; -2.61	1.88	15.6524
5	-5, 10	-3; -2.65	2.88	15.2546
		∑	2.592	15.5720

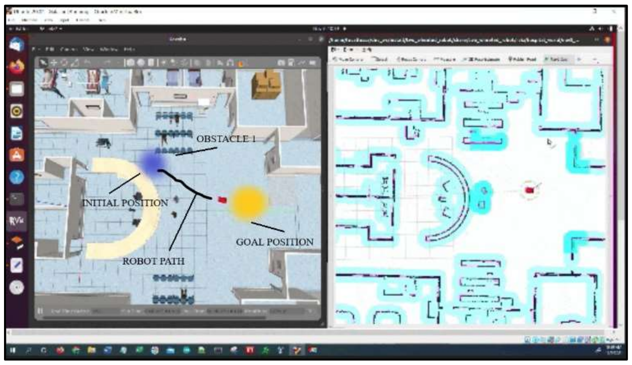


(a)



(b)

Special Case 2: As shown below (Fig.11) that the robot performs path planning with many obstacles as well as obstacle avoidance, In this simulation environment the robot is placed between many obstacles that are stationary as well as some of them are dynamic obstacles the robots need to analyze the data from the sensors and make the necessary decision and decide the most efficient path through the obstacles it helps the robot to reach its target decision in the best possible way. (Graph.2) below depicts the statistical information of case 2 of the indoor mapping environment performed by the robot. (Table.4) shows the results of the simulation performed by the robot in case 2.



(c)

Fig 11 Test scenario 2 created in Gazebo simulator

Fig (a) The robot initiates the action by analyzing the initial position of the robot. Fig (b) The robot starts the path planning and acts according to the set path planned by the software. Fig (c) The robot is in the given path and detects the object and avoids the object and moves around. These are some of the navigation codes used in the ROS environment to plan the path to be executed to reach from given position of the robot to the target destination. These codes include various parameters that the robot has to behave accordingly.

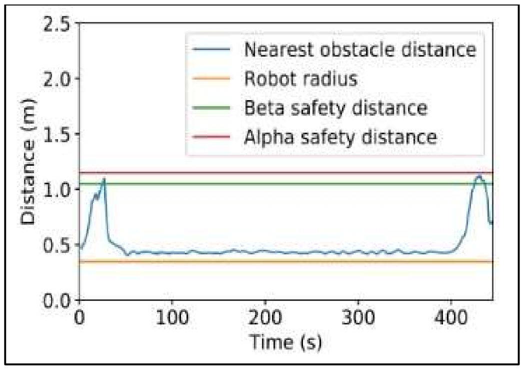


Fig. 12 Path planning using A* algorithm in Case-2

CONCLUSION

The experiments reveal that the A* algorithm takes the shortest path, however, it uses significantly more computing power than the other algorithms. The proposed research was in the view to develop a autonomous mobile robot system that is capable enough to smartly navigate to the given goal position, and efficiently follow the path provided by an optimized path planning algorithm. This study included that the A* algorithm is more efficient and better in performance with the robot in the Gazebo Simulator using the ROS (robotic operating system) framework, and it was discovered that the A* algorithm produced better results and computes the path more quickly in the particular environment in which the robot was situated in the Gazebo environment.

REFERENCES

[1] Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." ICRA workshop on open source software. Vol. 3. No. 3.2. 2009.

[2] Jain, Nitin, Amit Kumar Gupta, and Priya Mathur. "Autonomous drone using ROS for surveillance and 3D mapping using satellite map." Proceedings of the Second International Conference on Information Management and Machine Intelligence: ICIMMI 2020. Springer Singapore, 2021.

[3] Marin-Plaza, Pablo, et al. "Global and local path planning study in a ROS-based research platform for autonomous vehicles." Journal of Advanced Transportation 2018 (2018): 1-10.

[4] Quiñonez, Yadira, et al. "Simulation and path planning for quadcopter obstacle avoidance in indoor environments using the ROS framework." Trends and Applications in Software Engineering: Proceedings of the 6th International Conference on Software Process Improvement (CIMPS 2017) 6. Springer International Publishing, 2018.

[5] Post, Mark A., Alessandro Bianco, and Xiu T. Yan. "Autonomous navigation with ROS for a mobile robot in agricultural fields." 14th International Conference on Informatics in Control, Automation and Robotics (ICINCO). 2017.

[6] Korkmaz, Mehmet, and Akif Durdu. "Comparison of optimal path planning algorithms." 2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET). IEEE, 2018.

[7] Gawande, S. H. "A combined numerical and experimental study on metal expansion bellows for STHE." Journal of the Brazilian Society of Mechanical Sciences and Engineering 40 (2018): 1-14.

[8] Patle, B. K., et al. "Hybrid FA-GA Controller for Path Planning of Mobile Robot." 2022 International Conference on Intelligent Controller and Computing for Smart Power (ICICCSP). IEEE, 2022.

[9] Batik Garip, Z., et al. "Path planning for multiple mobile robots using A* algorithm." Acta Physica Polonica A 132.3 (2017): 685-688.

[10] Zidane, Issa Mtanos, and Khalil Ibrahim. "Wavefront and a-star algorithms for mobile robot path planning." Proceedings Of The International Conference On Advanced Intelligent Systems And Informatics 2017. Springer International Publishing, 2018.

[11] Pagar, N. D., S. S. Gawde, and S. B. Sanap. "Online condition monitoring system for rotating machine elements using edge computing." Australian Journal of Mechanical Engineering (2023): 1-14.

[12] Haldar, Arijit I., and Nitin D. Pagar. "Predictive control of zero moment point (ZMP) for terrain robot kinematics." Materials Today: Proceedings 80 (2023): 122-127.

[13] Gawande, S. H., and N. D. Pagar. "A combined numerical and experimental investigation on the effect of dynamics characteristics of metal expansion bellows." Journal of Vibration Engineering & Technologies 6 (2018): 401-416.

[14] Darade, Santosh A., M. Akkalakshmi, and Dr Nitin Pagar. "SDN based load balancing technique in internet of vehicle using integrated whale optimization method." AIP Conference Proceedings. Vol. 2469. No. 1. AIP Publishing, 2022.

[15] Sanap, Sudarshan B., and Nitin D. Pagar. "Structural Integrity Assessment of the Compensators Used in the Heat Exchangers Under Combined Angular Movement and Lateral Offset." ASME International Mechanical Engineering Congress and Exposition. Vol. 86717. American Society of Mechanical Engineers, 2022.