iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** IX **Month of publication:** September 2023

DOI: https://doi.org/10.22214/ijraset.2023.55790

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue IX Sep 2023- Available at www.ijraset.com

ROS-Based Food Delivery System

Rahul Gowlapalli[1], Deepesh Suranjandass[2]
Electronics and Communication Dept, VIT University(UGC), Hyderabad, India
Computer Science and Engineering Dept, VIT University(UGC), Bengaluru, India

Abstract: ***More people prefer to dine out nowadays, after having to spend many months in their homes due to the COVID-19 pandemic. The food and beverage industry has to revolutionise its way of serving customers in order to remain sustainable to the growing population. This project aims at the designing of a robot that delivers food to a table in a restaurant and returns back to the kitchen for the next order. Even though it is challenging to design a robot, doing so reduces the wastage of resources in restaurants. Multiple algorithms have been developed for the robot, which allow it to deliver food safely. The mobile robot follows a point to point algorithm and an obstacle avoidance algorithm in the fixed grid environment. The robot has been simulated in Gazebo, with the help of Construct Sim.***
Keywords: Gazebo, Robot, Obstacle avoidance.

I. INTRODUCTION

As we are aware there is a rapid spread of COVID-19 all over the world, the main reason for the spread of COVID is due to failure of maintaining social distance among the people.
This can be achieved by introducing robots in areas like restaurants, shopping malls, banks, transport stations etc. that means we need to decrease the physical interaction among people in crowded areas.
In this project we developed a robotic system which can deliver food to customers. The algorithm is similar to having robots in areas like medical assistance to COVID patients, in malls etc.
But we have chosen a restaurant because according to the report there is a rapid spread in restaurants, hotels as people remove their mask and spend more time in that area.

II. METHODOLOGY

This project has been developed in Constructsim and the robot has been simulated in Gazebo. Gazebo is a powerful robot simulator used to calculate physics, generates sensor data and provides convenient interfaces.

Steps involved in brief:

1) Explore the macros for URDF files using XACRO files
2) Insert a laser scan sensor to the robot
3) Read the values of the laser scanner
4) An obstacle avoidance algorithm
5) Create an algorithm to go from a point to another
6) Work with wall following robot algorithm

A. Explore the macros for URDF files using XACRO files: The Unified Robotic Description Format (URDF) is an XML file format used in ROS to describe all elements of a robot. To use a URDF file in Gazebo, some additional simulation-specific tags must be added to work properly with Gazebo.
In the xacro file we have defined chassis, caster wheel link, wheels and joint type for the left and right side of the robot. Visual, collision and inertia of the robot is included in this xacro file. Here visual property controls the appearance of the robot.Inertia and collision enable physical simulation.

1) World Launch file:
This launch file contains the environment where the robot has to move. The dimensions of the obstacles, where the obstacle has to place everything, is given in this launch file.

When we launch this launch file we get an environment as below-

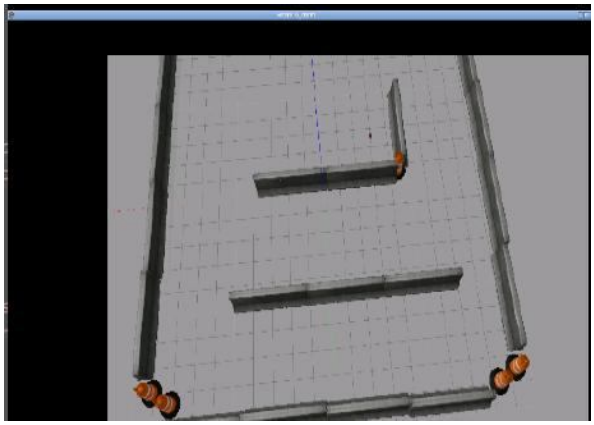


Fig 1. Launched environment

Then to place the robot in the environment we used spawn.launch file Once we run the file we get the output as below

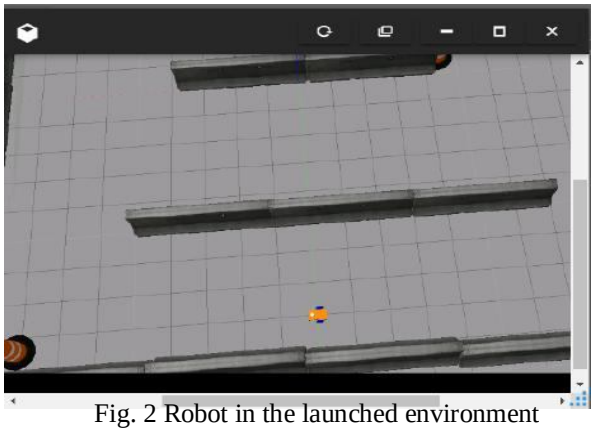


Fig. 2 Robot in the launched environment

B. Insert a Laser Scan Sensor to the Robot

A link is added which is in cylinder shape and this represents a laser sensor. To connect the sensor with the robot, we added a joint to the robot. For the sensor we also gave properties like radius, etc Once we give the minimum and maximum distance that the robot has to sense, the laser sensor starts sensing obstacles.

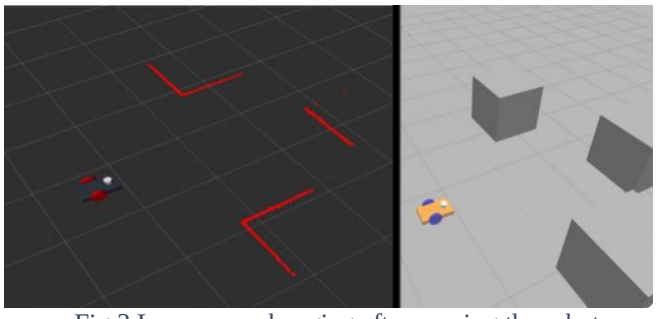


Fig.3 Laser scan changing after moving the robot

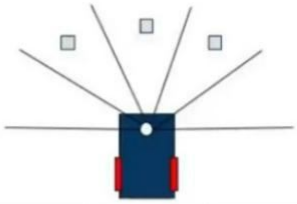
C. An Obstacle Avoidance Algorithm

Once we got the laser sensor values from the previous program, we used them in obstacle detection. In this we took values of the laser sensor from the left side, right side and front side of the robot.

The threshold value was given, which means when the robot finds a value which is less than 1 from any side, it means that there is an obstacle in that particular side. We used if-else statements to find in which direction the robot is having an obstacle.

Algorithm

1) If the front coordinate is greater than 1 , region of left is greater than 1 and region of right is greater than 1 then fix the state_description = 'case 1-nothing', linear_x = 0.6 , angular_z = 0
2) elseif the front coordinate is less than 1 , region of left is greater than 1 and region of right is greater than 1 then fix the state_description = 'case 2-front', linear_x = 0 , angular_z = 0.3
3) elseif the front coordinate is greater than 1 , region of left is greater than 1 and region of right is less than 1 then fix the state_description = 'case 3 - fright' , linear_x = 0 , angular_z = 0.3
4) elseif the front coordinate is greater than 1 , region of left is less than 1 and region of right is greater than 1 then fix the state_description = 'case 4 - fleft' , linear_x = 0 , angular_z = -0.3
5) elseif the front coordinate is less than 1 , region of left is greater than 1 and region of right is less than 1 then fix the state_description = 'case 5 – front and fright' , linear_x = 0 , angular_z = 0.3
6) elseif the front coordinate is less than 1 , region of left is less than 1 and region of right is greater than 1 then fix the state_description = 'case 6 – front and fleft' , linear_x = 0 , angular_z = -0.3
7) elseif the front coordinate is less than 1 , region of left is less than 1 and region of right is less than 1 then fix the state_description = 'case 7 – front and fleft and fright' , linear_x = 0 , angular_z = 0.3
8) elseif the front coordinate is greater than 1 , region of left is less than 1 and region of right is less than 1 then fix the state_description = 'case 8 – fleft and fright' , linear_x = 0.3 , angular_z = 0
9) else state_description = 'unknown case'



	Front Left	Front	Front Right
Case 1			
Case 2		x	
Case 3			x
Case 4	x		
Case 5		x	x
Case 6	x	x	
Case 7	x	x	x
Case 8	x		x

Fig. 4 Different cases of obstacle avoiding algorithm

D. Algorithm to go from a Point to Another

In this part implemented a simple navigation algorithm to move our robot from any point to a desired point. We used the concept of state machines to implement the navigation logic. In a state machine there are a finite number of states that represent the current situation (or behavior) of the system. In our case, we have three states--

1) *Fix Heading:* Denotes the state when robot heading differs from the desired heading by more than a threshold (represented by yaw_precision_ in code)
2) *Go Straight:* Denotes the state when robot has correct heading but is away from the desired point by a distance greater than some threshold (represented by dist_precision_ in code)
3) *Done:* Denotes the state when the robot has the correct heading and has reached the destination.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue IX Sep 2023- Available at www.ijraset.com

The robot can be in any one state at a time and can switch to other states as different conditions arise. This is depicted by the following state transition diagram --

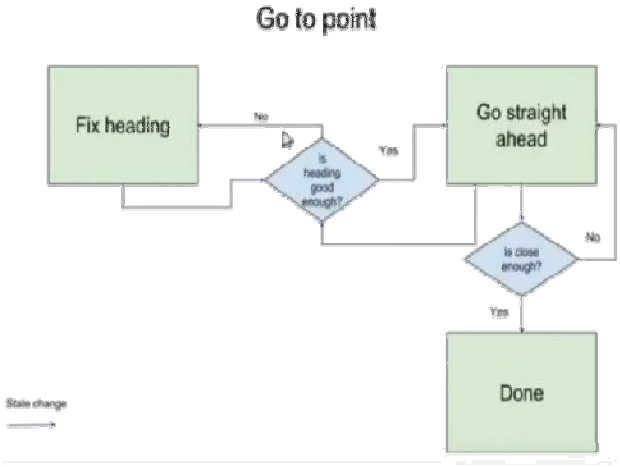


Fig.5 Summary of Go to point algorithm

Algorithm
Step1 - START
Step2 - Fix heading
Step3 - Repeat while heading is not good enough
1. Fix heading

Step4 - Go straight ahead
Step5 - Repeat while it is not close enough
1. Go straight ahead
2. Repeat Step-3

Step6 - Done

E. Wall following robot algorithm:

There are a few global variables in the code like
pub_ : this variable is a reference to the publisher that will be initialized inside the main function
regions_ : this is a dictionary variable that holds the distances to five directions
state_ : this variable stores the current state of the robot
state_dict_ : this variable holds the possible states
The functions defined are:

- Main : This is the entry point for the algorithm, it initializes a node, a publisher and a subscriber.Depending on the value of the state_ variable, a suitable control action is taken (by calling other functions). This function also configures the frequency of execution of control action using the Rate function.
- clbk_laser : This function is passed to the Subscriber method and it executes when a new laser
- data is made available. This function writes distance values in the global variable regions_ and calls the function take_actions
- take_action : This function manipulates the state of the robot. The various distances stored in regions_ variable help in determining the state of the robot. The three main stages in wall following robot is as below:
- find_wall : This function defines the action to be taken by the robot when it is not surrounded by any obstacle. This method essentially makes the robot move in an anti-clockwise circle (until it finds a wall).
- turn_left : When the robot detects an obstacle it executes the turn left action
- follow_the_wall : Once the robot is positioned such that its front and front-left path is clear while its front-right is obstructed the robot goes into the follow wall state. In this state this function is executed and this function makes the robot follow a straight line.

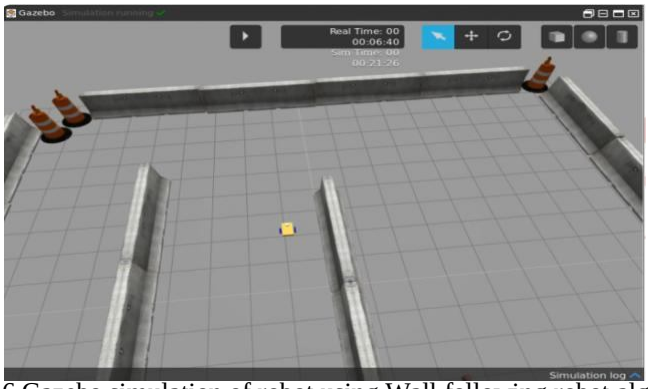


Fig.6 Gazebo simulation of robot using Wall following robot algorithm

III. RESULTS AND DISCUSSIONS

The robot was initially at (0,8,0) position. The laser sensor in the robot started sensing the obstacles. If the distance between the obstacle and robot is less than 1.5m then the robot considers that as an obstacle and starts moving in another direction. In this way it detects the obstacle and moves accordingly. When we ran the wall_follow.py program, the robot started moving along the wall. It followed the wall and reached the destination.

A. Robot Reaching the Destination

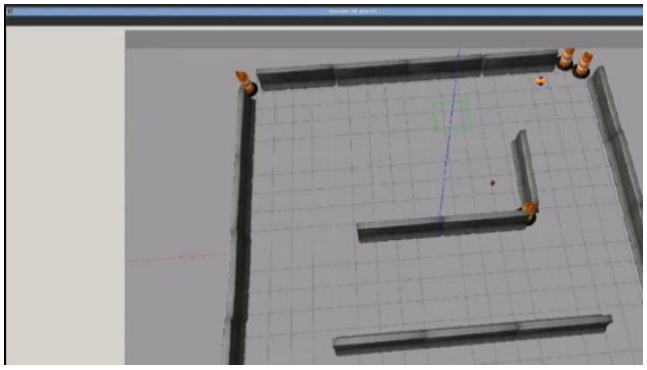


Fig.7 Robot reaching the destination

Now after we launched all the files, the robot started moving to the destination point (12,1,0). Finally the robot avoided all the obstacles in between and reached the destination by following the wall.

B. Robot Reached the Destination

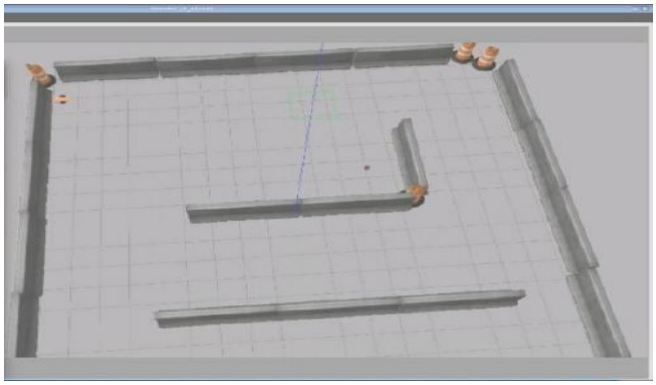


Fig.8 Robot reached the destination

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue IX Sep 2023- Available at www.ijraset.com

IV. APPLICATIONS

1) *Hospital Delivery:* Delivery robots can perform several tasks in hospital settings to reduce operational costs. The first set of tasks are for food, medical specimens, and medicine deliveries. With multiple sensors, the delivery robots can navigate the interior layout of the hospitals.
2) *Package Delivery:* Many E-commerce companies can use these delivery robots to deliver the packages to the respective addresses and can save a lot of manpower.
3) *Food Delivery:* Restaurants and hotels can use delivery robots to serve or deliver food to the consumers and during this pandemic period it will be great to have non contact delivery of food.

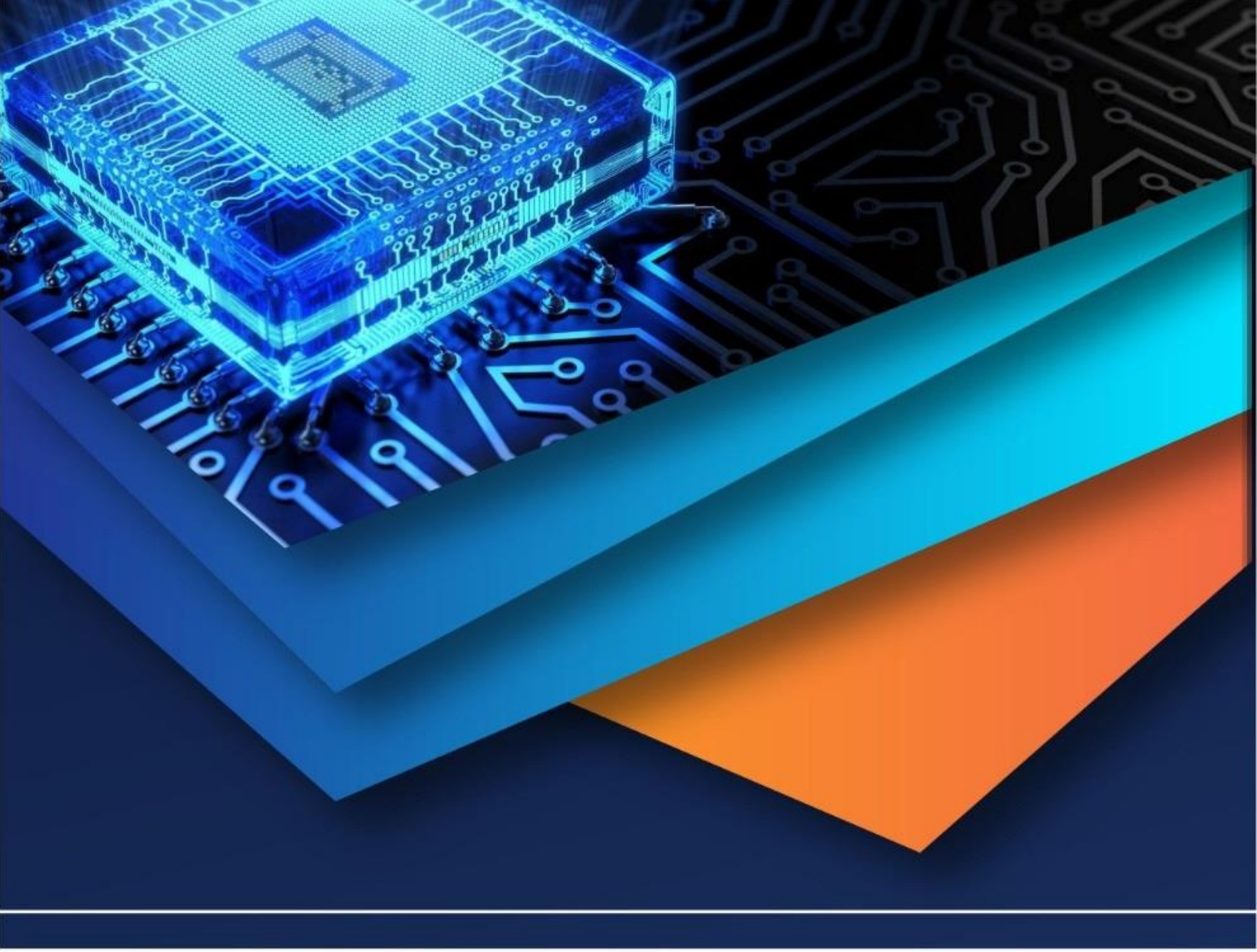
V. CONCLUSION

COVID 19 has changed our lives as we know it. This robot solves the problem of person to person contact in public areas like restaurants thereby reducing the risk of spreading to more population. Robots can work at a rate faster than humans, and thereby reduce the time taken to deliver the orders. The applications of this robot are numerous. This robot can be further developed and modified for various other applications such as delivering medicines to the elderly, holding sterilized medical equipment during surgeries and also to deliver groceries from a local market to a nearby house.

As it is a simulation process there is no requirement and usage of hardware in building the prototype. This helps us decrease the waste age of material which happens when we try to build a prototype with our calculations.

REFERENCES

[1] Raza, Sayyed Jaffar Ali & Gupta, Nitish & Chitaliya, Nisarg & Sukthankar, Gita. (2018). Real-World Modeling of a Pathfinding Robot Using Robot Operating System (ROS).
[2] J. BačÍK, F. ĎUrovský, M. Biroš, K. Kyslan, D. Perduková and S. Padmanaban, "Pathfinder–Development of Automated Guided Vehicle for Hospital Logistics," in IEEE Access, vol. 5, pp. 26892-26900, 2017, doi: 10.1109/ACCESS.2017.2767899.
[3] Pablo Marin-Plaza, Ahmed Hussein, David Martin, and Arturo de la Escalera , "Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles"
[4] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 34, no. 1, pp. 718–725, 2004.
[5] C. Rosmann, W. Feiten, T. Wosch, F. Hoffmann, and T. Bertram, "Efficient trajectory optimization using a sparse model," in Proceedings of the 2013 6th European Conference on Mobile Robots, ECMR 2013, pp. 138–143, IEEE, September 2013.
[6] P. Vadakkepat, K. C. Tan, and W. Ming-Liang, "Evolutionary artificial potential fields and their application in real time robot path planning," in Proceedings of the Congress on Evolutionary Computation (CEC 00), vol. 1, pp. 256–263, IEEE, July 2000.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY