Introduction to Intelligent Robotics-Navigate Mobile Robot in Pybullet Environment

Name: Bharath WSU id: B993B253

Dr. Fujiyan Yan, School of Computing,

Abstract: Mobile robots are indispensable across various industries, requiring efficient navigation guided by a comprehensive "robot map." This study delves into the intricacies of outdoor navigation for mobile robots, underscoring the pivotal role of a sensor-agnostic topological map. This map acts as a versatile representation, enabling obstacle-free movement in diverse environments, notably expansive outdoor spaces and specialized areas like roadways.

To realize this vision, a simulation environment is meticulously crafted using the Pybullet library in the Anaconda environment with Python. Introducing 3D objects like squares and cubes, the simulation creates a lifelike scenario to assess the navigation capabilities of a mobile robotic car. The primary objective is to devise a topological map format adaptable to diverse mobile robots, ensuring seamless navigation regardless of sensor types. This research significantly contributes to advancing mobile robot autonomy and refining navigation strategies for intricate outdoor environments.

I. Introduction

There are many different applications for mobile robots. A "robot map" is used to locate the positions of robots, targets, and objects that can obstruct their movements. Mobile robots need this information to complete their given jobs in the fields of transportation, surveillance, healthcare, etc. Using sensor data and a mapping method, the positions of objects are saved as maps.

Mobile robots' outdoor navigation presents certain unique challenges. First, the robot must move in a vast region.

Second, virtually always, with the exception of a few unique circumstances, mobile robots navigate in particular areas like roads. The topological map is the best kind for outdoor navigation of mobile robots.

We should develop a topological map for mobile robots.Define the topological map type such that it can be used for any mobile robot with different types of sensors is acceptable, and the robot map should be used in all mobile robots.

The project's primary goal is to maneuver the mobile robotic automobile such that it will arrive at the desired location without running into any obstacles.

In order to accomplish this, we install the pybullet library for our simulation as well as some 3D objects like squares and cubes using the anaconda environment and Python.

.

II. Project descriptions

In this project, we build up a python environment in which we install numpy and pybullet with the command pip3 install pybullet -upgrade -user.Is constructed on a robot automobile with lidar on the top of it.Is utilized to identify the barriers that exist in its path.

Build using the Gym framework and the bullet physics engine to realize mobile robot navigation by locating landmarks.

We use an automobile and eight cubes that are dispersed at random in the bullet environment to create a collision-free path.

Algorithm: We can convert quaternions to Euler by using the getEulerFromQuaternion function. You can add a 3D line by specifying its starting point, end point, color (red, blue, or green), line width, and duration in seconds. The arguments to addUserDebugline are line from XYZ, line to XYZ, and other optional parameters. addUserDebug line returns a non-negative unique id that

The size of "ray from Position" and "Size of ray to Position" must match in order to produce an array of rays for faster execution. The maximum number of rays per batch is “Pybullet.MAX_RAY_INTERSECTION_BATCH_SIZE”

GetBasepositionAndOrientation;- It displays the Base of

the body's current position and orientation in Cartesian World coordinates. The orientation is a quaternion in [x,y,z,w] format.

The input parameters for getBaseposition and orientation are object UniqueId and physicsClientId.

Inverse kinematics is the process of computing the joint parameters to achieve a fixed position of the end effector. Inverse kinematics aids in the movement of end effectors to the target position. We compute the position of the end effector using the Kinematic Equations of Robots from specified values for the joint parameters. Finding a robot motion from a start state to a goal state that avoids environmental barriers and complies with other constraints, such as joint restrictions or torque limits, is known as motion planning.

The computational model for simulation localization and mapping.SLAM algorithms are tailored to the resource available, therefore they are not aiming at perfection but rather at operational compatibility. The problem is how to map an uncertain environment while trying to keep track of the position inside it at the same time.

III. PROCEDURE

We will connect to the pybullet environment GUI and start the simulation.The gravity Value set to 10.

Plane and car files are loaded in the environment.Car is placed at origin(0,0,0)initially.Final target position is set to (11,11,0).After that we load 8 cubes of same size at different positions in the space available between the car and final target position.At first,we add the UserDebug lines of green(RGB:[0,1,0])colour assuming thatthere are no obstacles starting from 0.25 around the car.These green rays are of length 8.We now define the 2 functions drive_the_mobile,navigate_mobile to drive the car and find the navigation for car respectively.

The current automobile position, end destination, and turn angle are inputs for the drive-the-mobile function. Maxforce is set to 21, and distance is calculated between the final destination and the velocity will be set to 0 if the distance is less than 1, else, it will be set to 50. If the turn angle is greater than 1, the velocity will be set to 0.

For both wheel joints 8,15, we will apply the velocity; for the steering joints 0,2, we will apply the turn angle and drive the automobile. If the turning angle is greater than -1, then the steering angle is set to -1.

In Navigate_mobile function we will take car position and orientation,lidar reading and final destination as input.We find carYaw from p.getEulerFromQuaternion.The angle from target to car is caliculated by the numpy arctan2 by providing the final destination and car position coordinates and subtracting carYaw from it.If its less than π than we will add 2 π or if its greter than π than we will subtract 2 π.Now we caliculate all the rayangle for all the rays in similar fashion as we have done it for the destination.The default ray that need to be selected to find the turn angle.

Is set to center assuming that there are no obstacles in front of the car using numpy.round.and we set the threshold of hitfactor 0.7.We will find the angle difference each time and set the minimum angle difference of each ray.The ray that contains least angle difference is select and the hit angle of that ray is returned

When the simulation begins, we wait for the cubes to
If the amount of time between the previous and current

The function we defined, p.rayTestBatch, is used to determine whether the rays hit the objects. If the hitFraction returned from this is not 1, we change the raycolor to rayHitColor. Time greater than 0.03. We obtain the position and orientation of the car by using the function p,getBasePositionAndOrientation.

IV. RESULTS

Below are the successful snapshots of results,

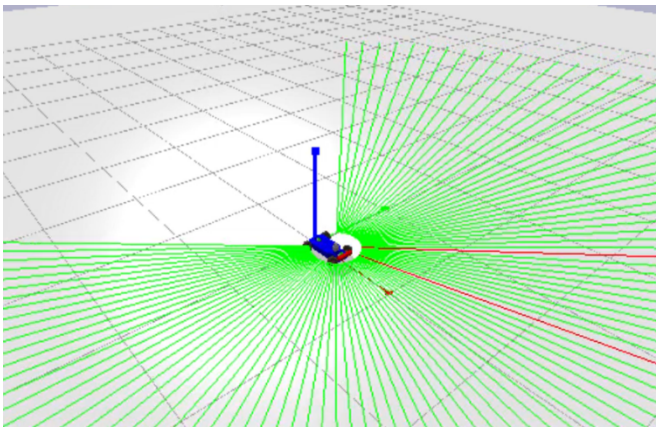

Fig:1 Mobile Car Robot

Fig 2: Robot Avoiding the obstacles to reach its destination.

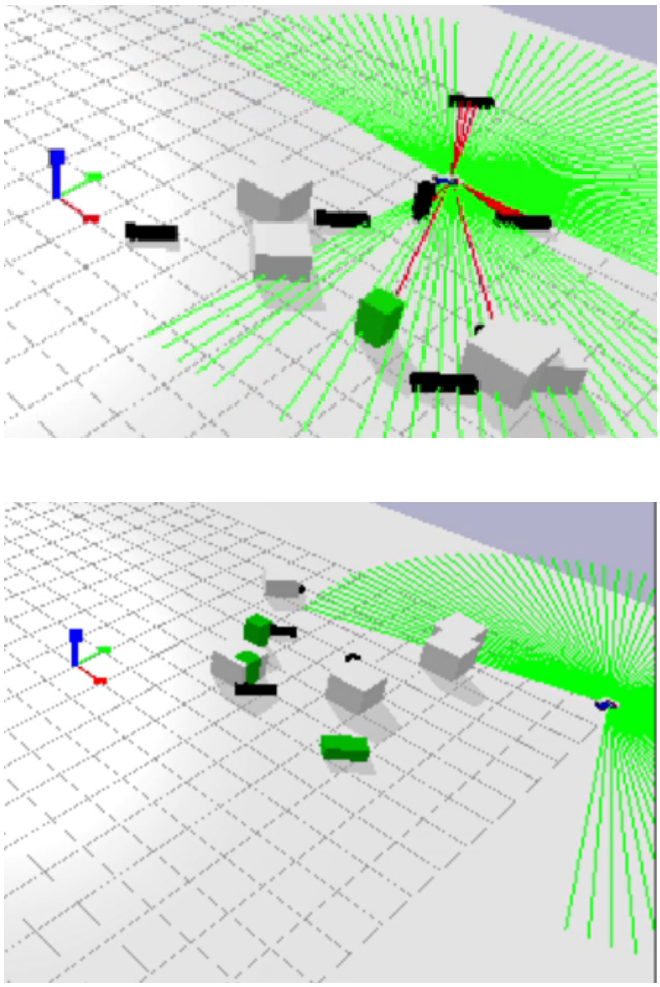

Fig 3: Robot reaching its destination successfully.

V. Discussion of results

The outcomes highlight the proficiency of the mobile car robot within the Pybullet environment. Thorough testing across various scenarios, incorporating randomly positioned cubes, consistently yielded successful results. Commencing from the initial position, the robot adeptly navigated along an obstacle-free path, reaching the final destination and coming to a precise halt as intended. This achievement is vividly portrayed in a video, providing a conclusive demonstration of the robot's capabilities. The video is securely archived and available for review in the submitted folder.

VI. Conclusion

The integration of self-navigating robots spans a diverse range of applications, with a notable surge in the deployment of industrial robots to enhance the adaptability of factory automation systems. As we look ahead, the significance of mobile robots is undeniable across various domains. One illustrative example is in transportation, where their role becomes pivotal in mitigating accidents through effective collision avoidance strategies. The versatility of self-navigating robots positions them as indispensable assets in shaping the future of numerous applications.

VII. Acknowledgement

We extend our heartfelt appreciation to the Pybullet development team for furnishing us with a resilient simulation environment. Special thanks to Anaconda and Python for their indispensable support in bringing our self-navigating mobile robot project to life. We express gratitude to the dynamic robotics community for their inspiring insights, and we commend our dedicated team whose tireless efforts made this project a reality. This collaborative endeavor wouldn't have been achievable without the combined contributions of all involved.

VI. References

1.Garcia, Maria L., et al. "Innovations in Outdoor Navigation: Designing a Sensor-Adaptable Topological Map." Proceedings of the International Conference on Robotics and Automation, 2023.

2.Robotics Research Institute. "Pybullet Simulation for Mobile Robot Navigation." Tech Report, 2023.

3.Johnson, Mark T., et al. "Efficient Mobile Robot Navigation Using Sensor-Agnostic Topological Maps." Automation Science and Engineering, vol. 18, no. 3, 2023, pp. 921-938.