Contents

Name: Akansha Bhatia - **Student ID:** M00975443 – **Subject Code**: PDE4430

GitHub Link: https://github.com/Akansha-Robotics/MR_CW2.git

Demo Video: https://www.youtube.com/watch?v=TamdVQ6wsvE

Guide to open Gazebo World & Robot Model

To open the Assessment world, you would need to go to the launch file inside robot_model_pkg with full path of ~/robot_ws/src/robot_model_pkg/launch and type below roslaunch

roslaunch robot_model_pkg robot_gazebo.launch

Next, you would need to load the robot model with the below roslaunch entered in the same terminal

roslaunch robot_model_pkg robot_spawn.launch

In addition, you can also the teleoperation file to control the robot within the same terminal using below roslaunch

roslaunch robot_model_pkg teleop.launch

These 3 terminal windows will be kept running for having the robot model within the gazebo assessment world. The reason for placing these elements in three different launch files was due to the fact that gazebo kept crashing when it was placed in one launch. Therefore, having different launch files will allow each element to properly load and not cause any error.

Finally, to launch the random objects in the gazebo world, you would need to go to launch file inside the assessment world with full path of /robot_ws/src/assessment_world-main/launch and type below roslaunch

roslaunch assessment_world objects.launch

Autonomous Navigation & Object Detection Strategy

In the GitHub submitted, I have included 2 codes to attempt to do autonomous navigation which are located in the controller folder with path /robot_ws/src/controller/scripts.

LiDar Sensor

: Conducting autonomous navigation includes various aspects with the first one being the sensors attached to the robot model allowing for the environment to be perceived. Therefore, I have added a LiDar sensor in which you can view the rays once the robot model is open in gazebo along with details of file found in /robot_ws/src/robot_model_pkg/urdf/lidar.xacro path. Furthermore, if you open the topic list by using ctrl + t, you would be able to view the topic called robot/base/laser/scan, you will notice that the sensor is able to make out the shape of the pillars.

SLAM Mapping: The next step would be to create a map of the area in which the robot would be moving. This map will allow the robot to identify what are the permanent obstacles placed, such as the pillars and goal post, along with the objects that have appeared, such as the spheres added to the area. In addition, with this map created, the robot would also be able to conduct localization in which it is able to find its position in the area based on the start point or frame. Through localization, the robot would be able to find its current position and orientation but also identify the target location and path to travel towards it.

Virtual Wall and Path: Another aspect to consider in the mapping process would be adding virtual walls if the robot is in quite an open area and the requirement calls for ensuring that the robot is moving in a particular area. By adding a virtual wall in the map, the robot understands that area can be crossed through and would avoid going beyond that point. In addition, certain obstacles, glass walls or tall tables in which the stand is quite thin, might not show up as obstacles due to lidar lasers passing through the glass wall or thin stands not being thick enough to be detected. Therefore, by enabling the user to add these virtual walls will ensure these obstacles to be properly marked and avoid the robot from colliding with the obstacle. Furthermore, by enabling the user to add a virtual path would communicate to the robot that these are the best paths to take in the area. Even though certain paths might have a faster time of arrival according to calculations done using the map, the reality might not be the same when operating in the real world as the environment might provide different challenges not seen on the map. For example, a path that crosses against a big entrance door where it is expected a huge flow of people to pass by might not be the best path as the robot would constantly be stopped due to detected obstacles coming toward it. Therefore, the user can create a virtual path that walks on side and further away from the entrance door allowing for the robot to move faster due to the lack of crowd coming toward the robot.

Attempted Navigation Code: In my GitHub submission, I have attempted to tackle this challenge starting with creating a teleoperation option (teleop.launch) that will enable the user to remotely move the robot allowing it to scan the area and build the map. In addition, the robot_spawn.launch file includes nodes for Gmapping which will enable the mapping to occur.

In addition, I have also created a python script called current_loc.py, located in the path of /robot_ws/src/controller/scripts, in which would display one set of X Y Z coordinates to provide the current location of the robot. The purpose of this script was to conduct localization by providing values of where the robot is placed in the gazebo world. Ideally, with an autonomous mobile robot, this process would be automated as the robot would be constantly calculating its place in the map along with making decisions based on the current location.

Furthermore, another python script created for navigation is called nav.py located in path /robot_ws/src/controller/scripts, which would move the robot according to the points entered by the user. This script was an attempt to move the robot from the values given to test the navigation. However, it requires more time to be given in development as a Map would need to be created using Rviz. Ideally, these X Y Z coordinates would not be imputed by the user but rather moved to location through teleoperation and indicated to the robot to automatically collect the coordinate values to ensure high level of precision.

Object Recognition: The final aspect of the strategy would be including an object recognition code that is connected with an HD camera on the robot allowing the robot to identify whether this object is a target to bring to the goal point or whether it is an obstacle. Using libraries such as YOLOv8 to recognize the object will allow for the robot to effectively identify and make decisions accordingly. Furthermore, the robot model can be further adjusted so that the left and right guide positions the object while the robot body could have a sweeper or suction technology to bring in the object to be stored. Once the storage is filled in the robot, the robot could then change path towards the goal point to place the object