AuE 893: Sp’20: Autonomy Science and Systems

Department of Automotive Engineering, Clemson University

Adhiti Raman, Mugdha Basu Thakur, Ardashir Bulsara and Venkat Krovi

Assignment 4 - Obstacle avoidance

Learning outcomes:

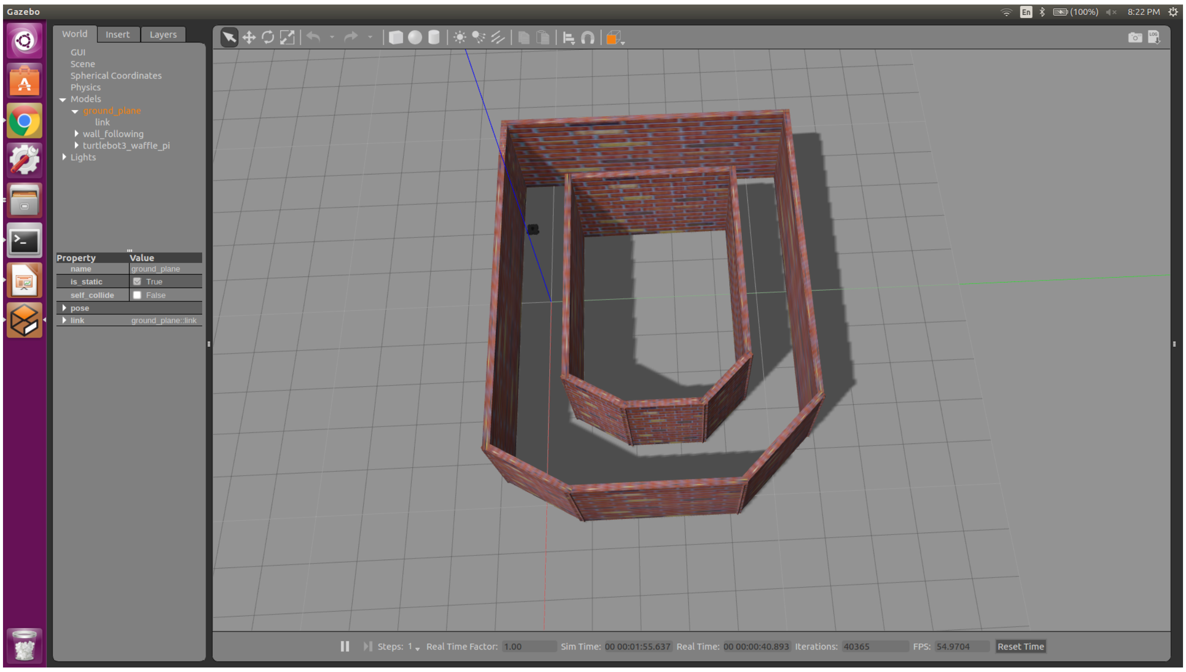
1. Manipulating scan data for navigation
2. Implementing P or PD controllers in Python

Instructions:

This assignment builds on the last assignment by upping the level of complexity. There are three parts. To get started create a new ROS package called "assignment4_obstacleavoidance" in your catkin workspace.

Part1: Wall following

Download the world files uploaded on canvas. The world for wall following:



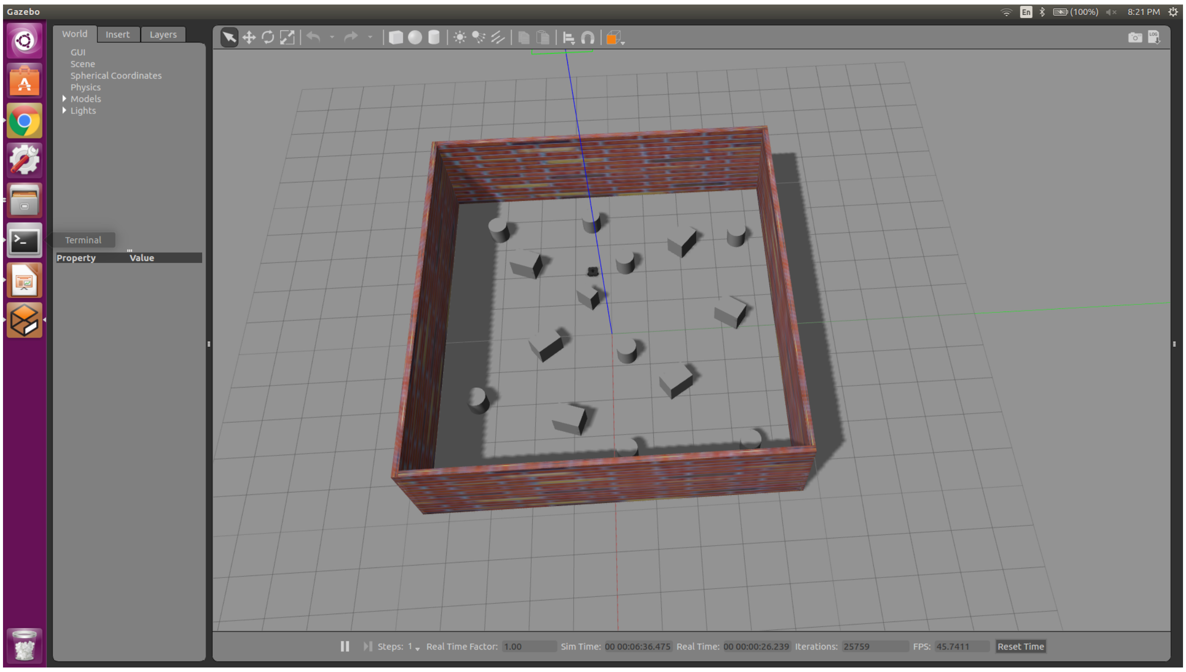
Property	Value
name	ground_plane
is_static	☑ True
self_collide	□ False
pose	
link	ground_plane::link

Steps: 1 Real Time Factor: 1.00 Sim Time: 00 00:01:55.637 Real Time: 00 00:00:40.893 Iterations: 40365 FPS: 54.9704

In the first part of this assignment you will create a python code `wall_follower.py` that extracts the `/scan` data and chooses data points that reflect the sides of the robot. Feed the robot a constant forward velocity and attempt to avoid the wall. You can write a simple P controller to maintain the equal distances from each face of the wall, or follow a single face while maintain a fixed (threshold) distance from it. Or you can come up with your own novel method.

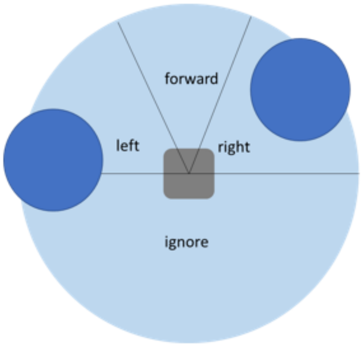
Part 2: Obstacle avoidance

The world for obstacle avoidance:



Steps: 1 Real Time Factor: 1.00 Sim Time: 00 00:06:36.475 Real Time: 00 00:00:26.239 Iterations: 25759 FPS: 45.7411

In this part you will create a python code `wander.py` where you implement obstacle avoidance using scan data. You can separate the scan into different sectors. For example:



In the right sector, the sum of all data points, divided by the number of data points (mean data) is less than that for the left sector or the forward sector -- this implied that should move forward-left. The steering angle of the robot can be a function of these three means. You may choose other approaches -- you may divide the data into several sectors or choose to not use sectors at all.

The robot can be given a constant forward velocity. There is not `goal` location, so the robot will be expected to simple wander around the room avoiding obstacles.

Part 3: Obstacle avoidance in the Turtlebot

Implement the obstacle avoidance code on the actual Turtlebot. There are cardboard boxes in our lab that can be used as obstacles and set up in the 4th floor mess area whenever you are testing out your code.

Your submission should include:

1) Videos of all three parts

2) Launch files to run each part

3) A README that shows the instructions the TA should input into the terminal to run your code, explanation of your implementation, and clearly indicate the contributions of each member.

Submit the link to the repo on Canvas.