Name: Rana Talal Ahmad Khan
Reg No: 450436

Project Report: Wall-Following TurtleBot 3 Using ROS and PID Control

Introduction

This project focused on developing and simulating a TurtleBot 3 robot to follow walls using the Robot Operating System (ROS) and a PID control algorithm. The primary goal was to enable the robot to autonomously maintain a specified distance from a wall.

Goals

- Create a TurtleBot 3 that can follow walls in a simulated environment.

- Implement PID control to keep the robot at a constant distance from the wall.

- Analyze the robot's performance within the simulation.

Methodology

- The TurtleBot 3 was set up and tested in a virtual environment using ROS and Gazebo.
- A lidar sensor was incorporated to measure distances from the wall, providing essential data for the control system.
- A PID controller was employed to adjust the robot's speed and direction, ensuring it remains at a set distance from the wall. The chosen PID gain values after tuning were:

  (Kp = 0.6)

  (Ki = 0)

  (Kd = 0.01)

- The robot was tested in a Gazebo simulation, configured with a specific wall layout to test the wall-following behavior.

System Setup

Install Ubuntu 20.04 (Focal Fossa) and ROS Noetic to provide the development environment for this project.

Catkin Workspace

Set up a catkin workspace to organize and manage the project files.

Script Development

Develop an executable ROS node script for the wall-following functionality and ensure it has the necessary permissions.

Installing Dependencies

Update and upgrade the system before installing TurtleBot3 packages:

```sh
sudo apt-get update

sudo apt-get upgrade
```

Install the necessary TurtleBot3 packages:

```sh
cd ~/catkin_ws/src/

git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git -b noetic-devel

git clone https://github.com/ROBOTIS-GIT/turtlebot3.git -b noetic-devel

cd ~/catkin_ws && catkin_make
```

Install the TurtleBot3 simulation packages:

```sh
cd ~/catkin_ws/src/
```

```
git clone https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
cd ~/catkin_ws && catkin_make
```

Edit the `.bashrc` file to include useful aliases:

```sh
gedit ~/.bashrc
```

Add these lines:

```sh
alias burger='export TURTLEBOT3_MODEL=burger'
alias waffle='export TURTLEBOT3_MODEL=waffle'
alias tb3fake='roslaunch turtlebot3_fake turtlebot3_fake.launch'
alias tb3teleop='roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch'
alias tb3='roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch'
alias tb3maze='roslaunch turtlebot3_gazebo turtlebot3_world.launch'
alias tb3house='roslaunch turtlebot3_gazebo turtlebot3_house.launch'
source /opt/ros/noetic/setup.bash
source ~/catkin_ws/devel/setup.bash
export TURTLEBOT3_MODEL=waffle
export SVGA_VGPU10=0
```

Project Directory Setup

Create the project package:

```sh
cd ~/catkin_ws/src
catkin_create_pkg my_turtlebot_pkg rospy geometry_msgs sensor_msgs
```

```` ``` ````

Transfer the wall-following script to this directory and make it executable:

```sh
mv /path/to/wall_follower.py ~/catkin_ws/src/my_turtlebot_pkg/src
chmod +x ~/catkin_ws/src/my_turtlebot_pkg/src/wall_follower.py
```

Rebuild the catkin workspace:

```sh
cd ~/catkin_ws && catkin_make
```

Running the Simulation

To start the simulation, execute:

```sh
export TURTLEBOT3_MODEL=waffle
roslaunch turtlebot3_gazebo turtlebot3_stage_1.launch
```

Run the wall-following script to activate the node.

Resetting the Simulation

If necessary, reset the Gazebo simulation:

```sh
rosservice call /gazebo/reset_simulation
```

Restart the script and resume the simulation.

Results

- The TurtleBot 3 effectively maintained the desired distance from the wall using the PID controller.
````

- The robot adapted to various wall layouts, demonstrating stable and consistent behavior.

- The simulation results confirmed the robustness and reliability of the PID control implementation for wall-following tasks.