Hafiz Muhammad Shahid Tariq

450950

Wall Following for TurtleBot3 using PID Control

Assignment 3

Table of Contents

[Introduction 1](#_Toc167219691)

[Setting Up the Robot 1](#_Toc167219692)

[Processing LiDAR Data 1](#_Toc167219693)

[Obstacle Detection and Avoidance 1](#_Toc167219694)

[Using PID Control to Avoid Obstacles 1](#_Toc167219695)

Introduction  
This assignment focus on developing a robust algorithm to enable the TurtleBot3 to autonomously navigate by maintaining a consistent distance from a wall. This task involves implementing a Proportional-Integral-Derivative (PID) controller, a widely used control loop mechanism that continuously calculates an error value as the difference between a desired setpoint and a measured process variable. By adjusting the robot's movements based on this error, the PID controller ensures precise and stable wall following. This project highlights key aspects of robotics, including sensor data acquisition, control system design, and real-time processing, providing practical insights into the application of PID control in autonomous mobile robots.

Setting Up the Robot  
First thing is importing necessary libraries for communication with the Robot Operating System (ROS), handling messages, and performing numerical operations. It then creates a ROS node named "explorer\_node," which acts as the brain of the robot controller. The node subscribes to a topic called "/scan" to receive data from the LiDAR sensor. This data contains distance measurements to surrounding objects. It defines various parameters for the robot's behavior:

* The minimum safe distance the robot should maintain from obstacles.
* The speed at which the robot moves forward in a clear path.
* The speed at which the robot rotates when avoiding obstacles.
* PID controller parameters for adjusting movement based on LiDAR data.

Processing LiDAR Data  
whenever a new LiDAR scan arrives, the laser\_callback function is called. This function first prepares a message to send velocity commands to the robot. It then extracts the distance readings from the LiDAR data and ensures they are within valid ranges. The function calculates the time difference between the current scan and the previous one to understand how much time has passed.

Obstacle Detection and Avoidance  
Checking the LiDAR data from the front and back of the robot to see if there are any obstacles within the minimum safe distance. If there are no obstacles:   
1) The robot sets its linear velocity to the move\_speed to move forward.   
2) The angular velocity (turning speed) is set to zero, indicating a straight path.

Using PID Control to Avoid Obstacles  
If obstacles are detected, the program calculates the error, which is the difference between the desired safe distance and the closest obstacle. It uses a Proportional-Integral-Derivative (PID)controller to determine how much the robot needs to turn to avoid the obstacle. Here's a breakdown of the PID controller  
**Proportional control**: The robot turns more significantly the closer it gets to an obstacle.  
**Derivative control:** The robot considers how quickly the obstacle distance is changing to anticipate future movements.

Based on the calculated error and its change over time, the robot adjusts its angular velocity (turning speed).The robot also reduces its linear velocity (forward speed) while turning to avoid collisions.