AIM

The primary goal of this project is to program the TurtleBot3 robot, equipped with a laser scanner, to autonomously follow walls. The key objectives include:

- Detecting walls
- Following walls
- Avoiding collisions with obstacles

All these tasks are to be performed automatically using a PID controller.

PID Controller:

A PID (Proportional-Integral-Derivative) controller is employed to adjust the robot's steering angle based on the distance error from the wall. The PID controller calculates the necessary angular velocity (ω) using the following control equation:

- Proportional Gain (Kp): This term responds to the present error, providing an immediate correction based on the magnitude of the error. A larger Kp results in a more aggressive response.
- Integral Time Constant (Ti): This term integrates the error over time, helping to eliminate steady-state errors. A smaller Ti emphasizes past errors and works to reduce any persistent offset.
- Derivative Time Constant (Td): This term considers the rate of change of the error, enhancing system stability by anticipating future errors. A larger Td provides damping, reducing oscillations.

By carefully tuning these parameters, the PID controller can effectively guide the robot along the desired path, ensuring accurate wall-following behavior while minimizing oscillations and maintaining smooth movement.

Methodology:

Data Acquisition and Storage:

The project starts by implementing a basic wall-following algorithm that can adapt to either the right or left wall, depending on the surrounding environment. Distance data is extracted from the LaserScan node in ROS and stored in a numpy array. To efficiently manage this data, a dictionary is created to store values at specific angles, defining the left, front, and right sides of the robot.

The robot can access distance information for all 360 degrees around it at

any given point. However, not all this information is necessary for wall following. Therefore, the first step is to store values at specific angles in a dictionary:

- Left: Defined between nodes 0-40
- Front: Defined between nodes 70-110
- Right: Defined between nodes 140-180

This customized data storage method allows the robot to efficiently utilize relevant distance information.

ROS Node Creation:

Once the LaserScan data is organized, a ROS subscriber node is created to subscribe to a callback function that processes this data. Additionally, a Velocity Publisher node is developed to command the robot's movements based on the wall-following algorithm.

Wall-Following Algorithm:

Initialization:

The algorithm begins by defining a global variable follow_dir, initialized to -1, indicating that the wall-following direction is yet to be determined. Minimum (a) and maximum (b) thresholds are established to define the algorithm's boundaries for detecting walls.

Decision Making:

When the robot detects a wall directly in front of it within the maximum threshold distance, it assesses the distances to obstacles on both sides:

- If the distance to the left is greater than to the right, the robot turns left, adopting the right wall-following algorithm.
- Conversely, if the robot turns right, it adopts the left wall-following algorithm.

The follow_dir variable is set to 0 or 1 based on the chosen wall-following direction.

Movement Execution:

As long as the wall is at least the minimum distance away, the robot moves straight. If the robot, following the right wall, encounters a corner with walls on both the front and right, it turns left. Similarly, if the robot reaches a corner where the wall ends, it navigates around while maintaining the appropriate distance.

State Management:

Different states are defined in the main function block to enable various actions for the robot. Depending on the selected algorithm, the robot can:

- Turn right
- Turn left
- Move straight
- Take diagonal turns on either side

Conclusion:

The PID controller effectively steers the TurtleBot3 to maintain a consistent distance from the wall. Meticulous tuning of the PID controller's parameters was crucial for optimal performance:

- The Proportional Gain (Kp) sets the initial response to errors. Too high a Kp can cause instability and oscillations, while too low a Kp can result in a sluggish response.
- The Integral Gain (Ki) helps eliminate steady-state errors but may cause overshoot if set too high.
- The Derivative Gain (Kd) provides a damping effect, reducing oscillations and enhancing responsiveness.

In conclusion, this project successfully demonstrated the implementation of a wall-following algorithm for the TurtleBot3 using a PID controller. The robot adeptly navigated along walls while maintaining a safe distance, showcasing the efficacy of the PID control method for precise navigation tasks. The algorithm proved robust against noise and uncertainties, highlighting its practical applicability in real-world scenarios.