Detailed Report: Wall and Line Follower Robot

1. Introduction

This report describes the implementation of a ROS (Robot Operating System) node for a TurtleBot3 Burger robot to perform wall following and line following tasks. The node utilizes sensor data (LiDAR and camera) for real-time navigation and integrates a PID controller for precise adjustments during line following.

2. Code Structure

The code consists of the following main components:

1. **Wall Following (LiDAR):**
   - Processes data from the '/scan' topic to detect walls and obstacles.
   - Implements logic to adjust linear and angular velocities to navigate corridors safely.

2. **Line Following (Camera):**
   - Subscribes to the '/camera/image' topic to receive image data.
   - Uses OpenCV to detect a line in the image and computes its centroid.

3. **PID Controller:**
   - Calculates proportional, integral, and derivative corrections to ensure smooth line tracking.

4. **ROS Integration:**
   - Publishes movement commands to the '/cmd_vel' topic to control the robot's motion.

3. Algorithms and Logic

3.1 Wall Following Logic:

The wall following logic uses LiDAR data to measure distances in specific directions:

- **Front (355° to 360°):** Detects obstacles directly ahead.

- **Front-Left (0° to 70°):** Measures proximity to the left wall.

- **Front-Right (270° to 340°):** Measures proximity to the right wall.

- **Side Distances:** Helps detect the end of walls.

Decisions are made based on these distances:

1. Stop if an obstacle is detected in the front.

2. Turn away from walls that are too close.

3. Move forward when the path is clear.

3.2 Line Following Logic:

The line following logic uses camera data processed with OpenCV:

- Converts the RGB image to HSV and masks a specific color (e.g., green).

- Detects contours in the masked image and identifies the largest one.

- Computes the centroid of the contour and calculates the error relative to the camera center.

- Uses a PID controller to adjust angular velocity based on the error.

4. PID Controller

The PID controller ensures smooth and precise line tracking. Its components are:

- **Proportional (P):** Corrects based on the current error.

- **Integral (I):** Compensates for accumulated past errors.

- **Derivative (D):** Predicts future errors based on the rate of change.

The control signal is computed as:

PID = Kp * error + Ki * integral + Kd * derivative

Where Kp, Ki, and Kd are the controller gains.

5. Visualizations

During execution, the node provides visual outputs for debugging:

- Masked image showing the detected line.

- Contours drawn around the detected line.

- Centroid of the line highlighted for error calculation.

6. Execution Steps

1. Run the ROS node:

   rosrun <package_name> wall_following.py

2. Launch the robot simulation or hardware setup.

3. Use 'rqt_image_view' to verify the camera topic and adjust if necessary.

4. Observe the robot navigating walls and lines based on sensor inputs.

7. Advantages and Limitations

Advantages:

- Real-time navigation using sensor data.

- Combines wall following and line following capabilities.

- PID controller ensures smooth and precise motion.

Limitations:

- Limited to environments with clear walls and lines.

- Performance depends on sensor calibration and lighting conditions.

- May require parameter tuning for optimal PID performance.