Autonomous Mobile Robots

CDA 4621

Lab Report 2

PID

Claude Watson

## Objective

The objective of this lab was to implement a Proportional-Integral-Derivative (PID) controller for autonomous navigation, focusing on two tasks: maintaining a specific distance from a front obstacle and following a wall using LIDAR sensors. The robot's goal was to navigate through a maze, avoid obstacles, maintain a constant distance from walls, and track its position using encoder and compass readings.

## Methodology:

### Task 1 – Distance Maintenance:

The robot's task in this section was to maintain a constant 1-meter distance from an obstacle in front. We used a simple PID controller based on LIDAR sensor data (index 400) to control the robot's speed based on the distance error.

#### Function: PID()

This function calculates the velocity needed to maintain the desired distance from the obstacle based on the proportional error.

Input: LIDAR range at index 400 (representing the front sensor).

Output: Motor velocity.

#### Formulas Used:

1. Error Calculation:  
Where:

* d\_actual is the current distance measured by LIDAR.
* d\_desired is the desired distance (1.0 meter in this case).

2. Proportional Control:  
Where:

* Kp is the proportional gain, set to 2.0.

3. Integral Control:  
The integral term is accumulated over time to correct small errors. In this case, the integral gain Ki is set to 0.

4. Derivative Control:  
The derivative term accounts for the rate of change of the error, helping to dampen oscillations. Here, Kd = 0.

5. Velocity Output:  
The calculated speed is returned and applied to both motors.

#### Saturation Function:

This function ensures that the robot’s motor velocities do not exceed the hardware's limit by capping the velocity to the maximum motor velocity.

### Task 2 – Wall Following and Navigation

In this task, the robot was required to follow a wall, maintaining a 0.4-meter distance from the side while avoiding collisions with obstacles and corners. This task was achieved through a combination of PID control for forward movement and angular velocity adjustments for wall-following.

#### Function: wallPID()

This function controls the robot’s angular velocity to maintain the desired lateral distance from the wall. The robot used LIDAR data from indices 200-390 (left) and 410-600 (right) to detect the closest wall and adjust its movement accordingly.

Input: LIDAR range data from both sides.

Output: Angular velocity based on the error in lateral distance from the wall.

#### Formulas Used:

1. Error Calculation:  
Where:

* d\_actual is the actual distance to the wall (either left or right) and d\_desired is the target distance (0.4 meters).

2. Proportional Control:

* Kp was set to 0.75 to provide a moderate response to the error.

3. Integral Control:

* Here, Ki = 0.00 since integral control was unnecessary for this task.

4. Derivative Control:

* The derivative control helps smoothen the motion when the robot encounters sharp corners. We set Kd = 9.0 to achieve rapid correction during high error rates (e.g., at corners).

5. Angular Velocity Output:

#### Function: wallFollow()

This function combines linear velocity from the PID() function and angular velocity from wallPID() to allow the robot to follow a wall. It adjusts the speeds of the left and right motors based on the distance from the wall, ensuring that the robot maintains the correct lateral distance while also moving forward.

* Input: Wall side (left or right), LIDAR readings.
* Output: Motor velocities for both wheels.

#### Position Tracking: updatePosition()

This function tracks the robot's position by calculating the distance it travels based on encoder readings and updates the robot’s coordinates and orientation using compass data.

#### Formulas Used:

1. Distance Traveled (from encoder):

Where:

* E\_current is the current encoder reading, r is the wheel radius (0.043m).

1. Position Update:  
   The robot's position (x, y) is updated based on the distance traveled and the compass reading (θ):  
   θ is converted from degrees to radians using θ\_rad = θ\_deg \* (π / 180).

#### Function: goal()

This function determines if the robot has reached its goal position. It checks if the robot’s coordinates are within a specified range around the target coordinates, (x = -1, y = 1), and stops the robot if it is close enough.

* Input: Current position.
* Output: Boolean indicating whether the goal is reached.

#### Function: rotate()

The rotate() function allows the robot to rotate by a specified angle in radians. It is used when the robot encounters a wall in front and needs to make a turn, typically a 90-degree turn, to continue navigating the maze.

* Input: Desired angle of rotation (in radians). Positive angles result in clockwise rotation, while negative angles produce counterclockwise rotation.
* Output: The robot rotates until the desired angle is reached, then stops.

#### Process:

1. The robot's compass reading is used to determine the robot's current orientation.
2. The function adjusts the motor velocities for the left and right wheels to create rotation (left wheel moves forward while the right wheel moves backward for clockwise rotation, and vice versa for counterclockwise).
3. The robot continuously checks the compass reading and compares it to the initial orientation to determine when the desired angle has been reached.
4. Once the robot has rotated by the specified angle, the motors stop, and the robot halts its rotation.

## Conclusion:

### Task 1 – Distance Maintenance

When implementing the PID controller for distance maintenance revealed several challenges in ensuring accurate distance control. Initially, I encountered a problem where the robot would move away from the wall instead of closer to it due to the error being calculated incorrectly. This was caused by the formula using , which produced a negative error. Adjusting the formula to  resolved the issue, allowing the robot to move toward the obstacle properly.

### Task 2 – Wall Following

In implementing the wall-following behavior, several issues arose related to handling distance to walls and navigating through sharp turns. Initially, the robot would move toward the wall instead of away when it got too close. This was due to the way velocities were calculated. Adding or subtracting the absolute value of the angular velocity corrected this, ensuring the robot steered in the correct direction.

Another issue was maintaining a consistent 0.4-meter distance from the wall. In maze 4, specifically when following the right wall, the deviations from the target distance were significantly noticeable, and the robot struggled to keep the distance stable. This instability made it difficult to maintain a smooth trajectory through the maze. Despite tuning the PID controller, the distance kept fluctuated.

Additionally, I found that the range of LIDAR data used for side detection was too narrow, causing the robot to struggle with turns. By increasing the range of LIDAR indices used for detecting walls, the robot’s ability to handle sharp turns improved significantly.

Lastly, achieving steady wall-following behavior during cornering required tuning the derivative term to smooth out the robot’s motion and prevent overshooting, but further adjustments are still needed to stabilize the wall-following overall.

Overall, the robot was generally able to follow the designated wall and reach the goal. However, when navigating maze 4 while following the right wall, the robot continuously looped around the right wall without progressing toward the goal.

## Videos

Task 1: <https://usfedu-my.sharepoint.com/:v:/g/personal/claudewatson_usf_edu/EYbK3QeDWNVEm8OeDeIO8jgBT0lK6DLcd0ue7YMYDxMiig?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=UZ8Rea>

Task 2: <https://usfedu-my.sharepoint.com/:v:/g/personal/claudewatson_usf_edu/ERpCG-9p12lHm-DqedDoTZ8ByVu_7tpYsy90lAFfJoi_TQ?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=LY3xmm>

## Author statement

I developed all the code and written report with the following exceptions:

Task 1:

[Lines 20:49 were developed with the help of generative AI]

Task 2:

[Lines 62:95, 129:149, 161:198, 200:211 were developed with the help of generative AI]

[Claude Watson: C.W., 10/21/2024]