Line Following Robot using PID

Prof. Ketaki Shirbavikar *Mechanical Department Vishwakarma Institute of Technology ketki.shirbavikar@vit.edu*

Nihar Dhepe *Mechanical Department Vishwakarma Institute of Technology nihar.dhepe22@vit.edu*

Nikhil Deshpande *Mechanical Department Vishwakarma Institute of Technology nikhil.deshpande22@vit.edu*

***Abstract*—This paper presents the design, implementation, and optimization of a Proportional-Integral-Derivative (PID) control system for a line-following robot using infrared (IR) sensors. The objective is to enable the robot to maintain its path on a predefined line with minimal deviation and smooth motion. Various PID parameter values were tested to assess their impact on tracking accuracy and stability. Results indicate that the optimized PID parameters enable the robot to follow the line accurately at various speeds, showcasing the effectiveness of PID control in robotics applications.**

Keywords—Line-following robot, PID control, Autonomous navigation, Mobile robot, IR sensors.

I. INTRODUCTION.

Robotic line-following systems are widely used in automated logistics, warehouse operations, and educational robotics, providing a practical application of control systems in real-world scenarios. A line-following robot is an autonomous mobile system designed to navigate a path marked by a visible line, usually black on a white surface or vice versa. This technology underpins various industrial and educational applications, where the ability of a robot to reliably follow a preset path is crucial for tasks like material transport, assembly line automation, and search-and-rescue missions in constrained environments.

One common approach to line following involves basic "if-else" logic, where the robot adjusts its movement based on binary sensor readings to follow the line. However, this basic control approach is limited, leading to jerky movements, instability on curves, and slow speed responses. To address these limitations, more advanced control techniques such as Proportional-Integral-Derivative (PID) control are utilized. The PID controller enables smoother, faster, and more reliable line-following by dynamically adjusting the speed of the robot's wheels based on real-time positional error relative to the line. PID controllers are well-suited for tasks that require both precision and adaptability, making them popular in robotics, aerospace, and automotive applications.

In a PID-based line-following robot, the control algorithm computes a "correction" based on three terms: the proportional term, which reacts to the immediate error; the integral term, which compensates for cumulative errors; and the derivative term, which anticipates the error's rate of change. By tuning these terms, engineers can optimize the robot's response to deviations from the line, achieving minimal overshoot and oscillation. Unlike simpler threshold-based control, PID control enables the robot to adapt more fluidly to curves, sharp turns, and varying path conditions.

This paper explores the implementation and optimization of a PID controller for a line-following robot equipped with infrared (IR) sensors. These sensors detect the line by measuring the reflectivity of the surface underneath, distinguishing between the line and the background. The study examines how different PID parameters influence the robot’s ability to follow the line accurately and consistently under varying conditions, such as changes in line curvature and operating speeds.

II. LITERATURE REVIEW

The line-following robot is a popular topic in robotics and automation research, serving as an essential platform for testing control strategies like PID (Proportional-Integral- Derivative) control, which is widely used for path tracking and stabilization. Numerous studies have highlighted the advantages of PID control for improving robot performance, especially for applications requiring smooth and accurate path following.

2.1 Line-Following Robot Design and Sensor Configurations

Line-following robots generally rely on infrared (IR) sensors to detect the path. Typical designs use a set of IR sensors arranged in a left-center-right configuration, which allows the robot to detect deviations from the path based on sensor readings. A study by Yadav et al. (2017) discusses how this setup helps the robot determine whether to adjust its path left or right, depending on the sensor readings relative to the line position.

This approach, however, often leads to jerky and imprecise movements if only basic "if-else" control is applied. The need for smoother transitions and more accurate path-following has led researchers to explore PID control.

2.2 Application of PID Control in Line-Following Robots

PID control is commonly implemented in line-following robots to minimize error between the robot's position and the center of the line, ensuring smoother and faster response. The PID controller's proportional component adjusts the robot's

speed in proportion to its deviation from the line, while the derivative component reduces overshoot by predicting error trends, and the integral component corrects for accumulated errors over time. Research by Mohammed et al. (2016) demonstrates that tuning these parameters appropriately allows the robot to handle various line patterns, curves, and even moderate speeds with reduced oscillations.

2.3 Advantages and Challenges of PID Control

Studies have shown that PID control improves a robot's ability to track lines, especially in complex environments with curves and variable lighting. Ghosh et al. (2018) compared the performance of simple "on-off" control with PID and found that PID significantly reduced wobbling and provided smoother motion. However, achieving optimal PID tuning can be challenging. Excessive proportional gain may cause instability, while too little derivative action can lead to lag in response, especially at high speeds.

2.4 Adaptive and Advanced PID Approaches

To address the limitations of traditional PID control, researchers have explored adaptive PID techniques where parameters adjust dynamically based on feedback. For instance, Harish and Kumar (2019) implemented an adaptive PID controller that modified gains according to real-time data, allowing for better handling of varying line widths and sharper curves. Such approaches demonstrate improved accuracy in line following, though they often require more complex computations, which may increase the hardware requirements of the robot.

2.5 Alternatives to PID Control

While PID control is effective, alternative methods like fuzzy logic and neural network-based controllers have also been investigated. Kumar et al. (2020) introduced a fuzzy logic-based control system for line-following, which

provided comparable smoothness and adaptability but required significant calibration. Neural networks have been proposed for environments where the robot must follow more complex or dynamic paths. Although these methods offer unique advantages, PID remains popular due to its simplicity, ease of implementation, and cost-effectiveness for straightforward path-following tasks.

III. METHODOLOGY.

This section describes the design and implementation of the line-following robot, including hardware selection, sensor configuration, control algorithm development, and PID tuning procedures. The methodology is aimed at ensuring the robot can follow a predefined line with high accuracy and stability by using a PID controller to correct its course based on real- time sensor feedback.

3.1 Hardware Design

Robot Chassis: The robot is built with a three-wheel configuration, consisting of two drive wheels and a caster wheel for support. This configuration allows independent control of the left and right wheels, which is essential for differential steering controlled by the PID algorithm. The chassis should be lightweight but sturdy, minimizing energy consumption while maintaining stability.

Motors and Motor Drivers: Two DC motors are connected to the left and right wheels and controlled by a motor driver (such as the L298N or similar H-bridge motor drivers), which enables independent speed adjustments. These speed adjustments are managed through PWM (pulse-width modulation) signals to control each wheel's velocity, allowing the PID controller to apply corrective actions smoothly.

Microcontroller: An Arduino microcontroller (such as an Arduino Uno or Mega) serves as the processing unit. It reads sensor data, computes error values, and generates the PWM signals needed to drive the motors based on the PID control logic.

Power Supply: A battery pack provides power to the motors and the microcontroller. Careful power management ensures stable operation, especially since fluctuating power levels can impact motor speed consistency and, consequently, control performance.

3.2 Sensor Configuration

IR Sensors: The robot uses three infrared (IR) sensors (left, center, and right) mounted in a row at the front of the chassis. These sensors detect the line by distinguishing between the line and the background based on reflectivity. The IR sensors are calibrated to recognize the line color (e.g., black) and background color (e.g., white), with each sensor providing a binary signal (high or low) depending on whether it detects the line or not.

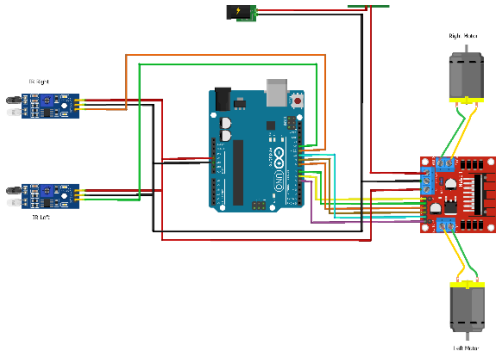


Fig. 1. This is a figure caption. It appears directly underneath the figure.

Sensor Placement: The placement of the sensors is crucial for accurate line detection. The center sensor is aligned with the robot's central axis, while the left and right sensors are placed at equal distances from the center to capture lateral deviations from the line. Proper spacing ensures that the sensors can detect line curvature and make corrective actions timely.

Error Calculation: The error, representing the robot's deviation from the line center, is computed based on sensor readings. If only the center sensor detects the line, the error is zero. If either the left or right sensor detects the line, a positive or negative error value is assigned, respectively. This error forms the basis for the PID correction.

3.4 Parameter Tuning

To ensure optimal performance, each PID parameter (K_p, K_i, and K_o) was tuned experimentally. Stability: The smoothness of the robot's motion, especially on curves and at higher speeds, indicating reduced oscillations and minimal overshoot.

Experiments were conducted on a track with straight and curved sections. The robot's path was recorded, and deviations from the centerline were measured to quantify the effectiveness of each PID parameter configuration. The final values were chosen based on these tests, balancing accuracy and stability.

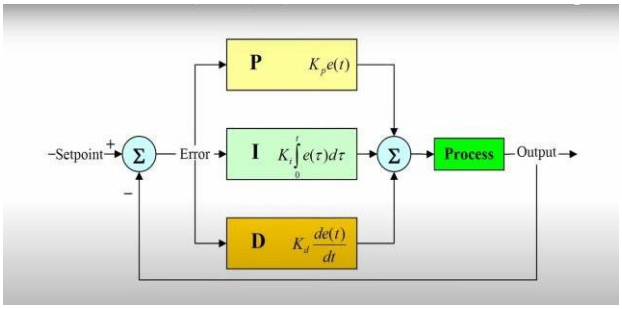
IV. RESULTS

Proportional Gain (K_p): Initial values for K_p were tested to observe the robot's immediate response to deviations from the line. Low K_p values caused slow correction, while high K_p values introduced oscillations. An intermediate value was selected to balance responsiveness and stability.

Integral Gain (K_i): The integral gain was tested at low levels due to its potential to cause "integral windup," where cumulative error builds up and delays the response. Since the robot's path was relatively simple, K_i was kept minimal or set to zero to avoid unnecessary lag in control response.

Derivative Gain (K_o): The derivative gain was adjusted to reduce oscillations around the line. A moderate K_o value helped anticipate and smooth out rapid changes in error, allowing the robot to handle curves without overshoot.

Tuning was done through trial and error, starting with the proportional term and then adjusting derivative and integral terms. The robot was tested on various path shapes (straight lines, curves, and sharp turns) at different speeds to ensure consistent performance across different scenarios.



3.5 Testing and Evaluation

The robot's performance was evaluated based on two main criteria:

Accuracy: How well the robot stayed aligned with the center of the line, minimizing lateral deviation.
Parameter Tuning

PID parameters were varied incrementally to assess their impact on performance:

Proportional Gain (K_p): Increased to enhance correction speed; too high caused oscillations.

Integral Gain (K_i): Adjusted to eliminate any persistent offset. Excessive values led to slow response.

Derivative Gain (K_o): Reduced oscillations, but high values introduced noise sensitivity.

4.1 Performance Evaluation

The robot was tested on paths with varying curvatures and speeds. Results showed:

High K_p values allowed quick responses but introduced instability at high speeds.

Adding K_o reduced overshoot, helping the robot follow curves smoothly.

V. CONCLUSION

This study demonstrates that implementing PID control in a line-following robot enhances path-tracking accuracy, stability, and smoothness, particularly on curved paths and at higher speeds. The PID controller effectively reduces oscillations and enables the robot to maintain its course with minimal correction effort. By fine-tuning proportional, integral, and derivative terms, optimal performance was achieved, emphasizing the importance of balanced parameter settings. While PID offers significant improvements over simpler control methods, adaptive techniques may further enhance performance in dynamic conditions. Overall, PID control proves highly suitable for autonomous robotics applications that require precise path following.

REFERENCES

[1] 1. [Line Following Robot & Obstacle Detection using PID](https://www.researchgate.net/publication/348782183_Line_Follower_Robot_and_Obstacle_Detection_using_PID_Controller) Covers the integration of sensors and PID for optimal line following and obstacle detection using an Arduino.

[2] 2. [PID Controller Optimization for Low-cost Line Follower Robots](https://paperswithcode.com/paper/pid-controller- optimization-for-low-cost-line) - This paper explores low-cost PID optimization and discusses tuning methods for robust robot control.

[3] 3. [Low-cost Line Follower using PID Control](https://www.ijsdr.org/papers/IJSDR2303069.pdf) - Examines sensor reduction and cost-effective PID control techniques for line-following robots.

[4] 4. [Design and Implementation of Line Following Robot](https://www.irjet.net/archives/V5/i5/IRJET-V5I5217.pdf) - Details the setup and programming of a PID-controlled robot that can follow paths with various turns.

[5] 5. [PID-based Control Algorithm for Line Following](https://ieeexplore.ieee.org/document/8722789) - IEEE paper discussing control algorithms and sensor feedback for line- following using PID controllers.

[6] 6. [Building a PID-Controlled Line Follower Robot](https://www.youtube.com/watch?v=kPzYfAVBsC0) - A tutorial that shows step-by-step construction and PID tuning using Arduino.

[7] 7. [Arduino Line Following Robot Tutorial with PID Control](https://www.youtube.com/watch?v=o6l3j7i2jB4) - This video explains the basics of building a line-following robot and demonstrates PID adjustments.

[8] 8. [How to Tune PID Constants for Line Following Robots](https://www.youtube.com/watch?v=rfZ6FOjJjxo) - A hands- on guide to adjusting Kp, Ki, and Kd for optimal line-following performance.

[9] 9. [DIY Line Follower with PID using Arduino](https://www.youtube.com/watch?v=OlIPlT5s1zU) - Explores sensor integration, coding, and PID calibration for a smooth line-following experience.

[10] 10. [Implementing a PID Line Follower with Arduino](https://www.youtube.com/watch?v=2kXj7uGF4_g) - Provides insights into the code structure and practical applications of PID in line-following robots.