Implementation and Tuning of a PID Controller for Wall-Following Robot using Pattern-Adaptive PID Tuning

Jonathan Atiene
Department of Artificial Intelligence
De Montfort University
Email: p2839161@dmu.ac.uk

***Abstract*—This report presents the implementation and tuning of a Proportional-Integral-Derivative (PID) controller for a wall-following pioneer robot. The work focuses on practical implementation in CoppeliaSim using Python, including output limitation, and parameter tuning. Experimental results demonstrate the effectiveness of a pattern adaptive tuning of the PID values through quantitative performance metrics and visual comparisons in the stipulated environment.**

***Index Terms*—PID control, robotics navigation, parameter tuning, wall-following, CoppeliaSim**

I. Introduction

Modern autonomous robots require precise control systems for effective navigation. Finite State Machines offer basic programmatic control through pre-defined states and reactive behaviours but lack the sophistication of more advanced control methods [2], PID controllers offer improved accuracy for continuous tasks like wall-following. In this report, I will explore the following:

- Implementation
- Practical tuning
- Quantitative Analysis

II. Methodology

A. PID Implementation

The controller architecture follows:

1) Proportional Control: Initial implementation used distance error $e(t) = d_{actual} - d_{desired}$. Gain K_p was set to 1, K_i integral gain was set to 0.5 and K_d was set to 0.5.

2) Error Management: The PID controller was used at first as an array with an unlimited size as long as the controller continued, then was set to 30 errors, but this did not reflect the true state of the controller as oscillations were observed at a high rate and the distance between each loop affected the rate of oscillations.

B. Tuning Strategy

This exercise used several tuning methods: Manual Random perspective, Ziegler-Nichols, and adaptive tuning.

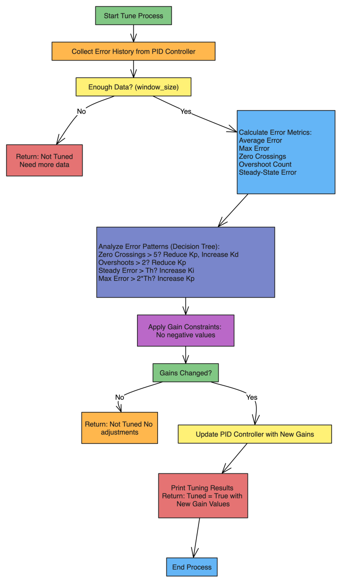
Start Tune Process
Collect Error History from PID Controller
Enough Data? (window_size)
No
Yes
Return: Not Tuned Need more data
Calculate Error Metrics: Average Error Max Error Zero Crossings Overshoot Count Steady-State Error
Analyze Error Patterns (Decision Tree): Zero Crossings > 5? Reduce Kp, Increase Kd Overshoots > 2? Reduce Kp Steady Error > Th? Increase Ki Max Error > 2*Th? Increase Kp
Apply Gain Constraints: No negative values
Gains Changed?
No
Yes
Return: Not Tuned No adjustments
Update PID Controller with New Gains
Print Tuning Results Return: Tuned = True with New Gain Values
End Process

Fig. 1. Adaptive tuning algorithm

1) Manual tuning: :

a) Increase K_p for faster response and to reduce steady-state error, but decrease it to prevent overshoot and oscillations.
b) Increase K_i to eliminate persistent steady-state error, but decrease it to prevent oscillations.
c) Increase K_d to dampen oscillations and prevent overshoot but decrease it to reduce noise and jerky control output.

2) Ziegler-Nichols:: The robot did not seem to attain optimal behaviour when used with the Ziegler-Nichols method this tends to confirm the work of Visioli who stated that While Ziegler-Nichols provided initial gain estimates [1], final tuning required iterative adjustments to address corner-handling overshoot and sensor noise [4].

3) Adaptive Tuning:: The Figure 1 shows the algorithm that was used in tuning the PID values the pid values started with K_p = 1.0 K_i = 0.5 and K_i = 0.5 the stepwise adjustments were manually tuned with values between 0.5 - 0.05 in the figure 2 we notice

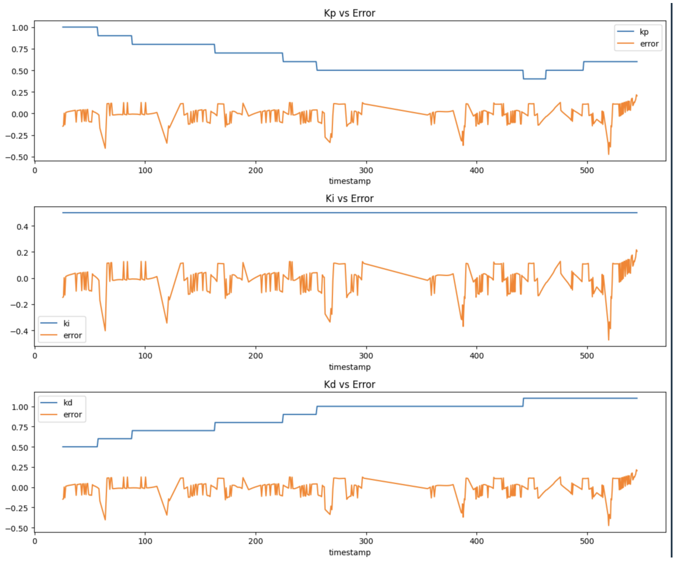

Fig. 2. Tuning process showing error reduction across time

C. *Effect of adaptive tuning*

1) Kp vs Error: As the proportional gain (Kp) decreases stepwise, the error showed fluctuations, indicating the effect of Kp on stabilizing or destabilizing the system.
2) Ki vs Error: The integral gain (Ki) remained constant, suggesting no significant tuning influence on error correction over time in this scenario.
3) Kd vs Error: The derivative gain (Kd) increases incrementally, showing the potential impact on error smoothing, though some oscillations persisted.

III. Results and Analysis

The performance of the wall-following robot was evaluated using a Proportional-Integral-Derivative (PID) control system with parameters configured as follows: $K_p = 9$, $K_i = 0.0005$, and $K_d = 0.1$. The robot was tested in corners and straight lines to check its effectiveness:

- **Base Speed:** 0.75 m/s
- **Wander Speed:** 1.0 m/s
- **Wall Follow Distance:** 0.3 m
- **Obstacle Threshold:** 0.5 m
- **Detection Threshold:** 1.0 m

A. *Error Convergence*

The graph in Figure 3 illustrates the error convergence of the robot as it adjusts its path to maintain the desired wall-following threshold. The error is the difference between the desired wall-following distance (0.3 m) and the distance detected by the sonar sensors.

From the graph, it is evident that the error showed oscillations before converging over time. The initial spikes in the error are attributed to sudden changes like curves and sharp corners. The PID controller shows the capability to compensate effectively for these disturbances, demonstrating stability in maintaining the desired distance.

B. *Performance Metrics*

To further evaluate the performance of the robot, several key metrics were calculated based on the error data collected during operation. These metrics provide deeper insights into the effectiveness and stability of the control system. The results are summarized in Table I.

Metric	**Value**
Maximum Overshoot	0.59 m (0.6%)
Integral Absolute Error (IAE)	38.8914
Noise Level (Standard Deviation)	0.0000
Noise Level (Root Mean Square)	2.2236

TABLE I
Quantitative Performance Metrics of the Wall-Following Robot.

a) Maximum Overshoot: The maximum overshoot, calculated as the largest deviation of the error from the desired value during operation, was observed to be 0.59 m, corresponding to 0.6% of the wall-following distance. This overshoot is within acceptable bounds for the given environment.

b) Integral Absolute Error (IAE): The integral absolute error (IAE), a measure of cumulative error magnitude, was calculated to be 38.8914 during the entire duration.

c) Robots Position: The tuned PID (Kp=9, Ki=0.0005, Kd=0.1) reduced path deviation significantly compared to the manual programmatic control in the simulated environment.

IV. Conclusion

The implemented PID controller demonstrated significant success over control especially in the scenario of wall following. It was able to recover from noise Manual tuning and adaptive tuning were used. Different methods were used including using PID to correct turn angles this did not produce good results as oscillations and awaiting time for turning robots yielded errors.

Comparing this to manual control it is clear that errors converged, while in the manual error, we had fluctuating values around ± 0.2m.

Future work should investigate adaptive PID gains for dynamic environments and how to use manual controls to overshadow noise.

References

[1] Åström, K.J. and Hägglund, T. (1995). *PID Controllers: Theory, Design, and Tuning*. Research Triangle Park, NC: Instrument Society of America.
[2] Brooks, R. (1986). A robust layered control system. IEEE Journal of Robotics.
[3] M. Brett (2005). PID Without a PhD. Embedded Systems Programming.
[4] Visioli, A. (2001). Tuning of PID controllers with fuzzy logic. *Control Engineering Practice*, 9(5), pp. 519–532.
[5] Ziegler, J.G., Nichols, N.B. (1942). Optimum settings for automatic controllers. Transactions of the ASME.

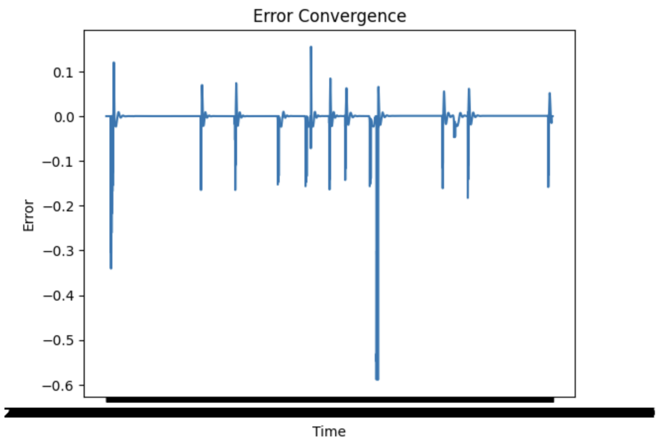

Fig. 3. Error Convergence during wall-following operation.

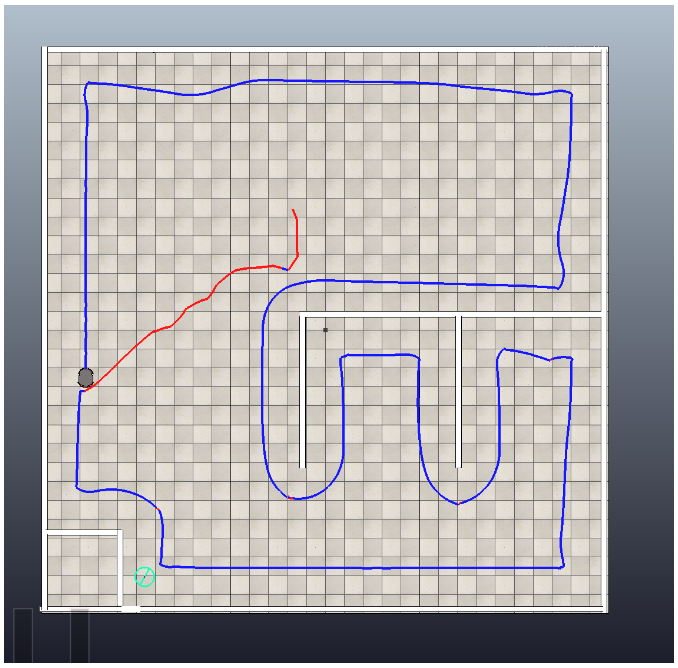

Fig. 4. robots path during adaptive tuning.