Curvature-Aware Discrete PID With Anti-Windup For High-Speed Line-Following Robots: Modeling, Implementation, And Repeatable Evaluation

[1]Adarsh Mittal, [2]Ishan Kumar,

[1]Independent Researcher, [2]Independent Researcher

Abstract: Line-following robots remain a canonical bench- mark for closed-loop control on embedded platforms because they combine non-linear sensing, actuator saturation, and tight real-time constraints. Although proportional–integral–derivative (PID) control is widely used, deployments frequently suffer from oscillation on sharp curves, integral windup under actuation limits, and sensitivity to surface reflectance changes. This paper presents a *curvature-aware discrete PID* design for differential- drive line followers that (1) uses a filtered centroid-based error estimator from a reflectance array, (2) incorporates derivative filtering and back-calculation anti-windup for stability under PWM saturation, and (3) applies lightweight gain scheduling driven by an online curvature proxy derived from the sensor profile. We provide a reproducible modeling pipeline, embedded implementation details for 1 kHz control, and an evaluation pro- tocol across three track types (low-curvature, mixed, and high- curvature). Results show that the proposed controller reduces mean absolute lateral error by 3.6% versus a fixed-gain PID and by 44.6% versus a Ziegler–Nichols tuned PID, while improving the 95th-percentile error and reducing oscillatory behavior at high speed.

Index Terms - Line following, PID control, anti-windup, gain scheduling, embedded control, reflectance sensors, differential- drive robot.

I. Research Study

This work studies the practical question: *how can a low-cost line-following robot sustain high speed on mixed-curvature tracks without oscillation or frequent retuning?* Our hypothesis is that two failure modes dominate at speed: (i) saturation- induced windup and delayed recovery after sharp turns, and (ii) curvature-dependent plant variation that invalidates fixed gains. We therefore combine a discrete PID with (a) derivative filtering to attenuate quantization and sensor noise, (b) back-calculation anti-windup to stabilize recovery, and (c) curvature-aware gain scheduling that increases damping and proportional action only when required. We quantify performance using lateral tracking error, tail (95th-percentile) error, maximum deviation, and a repeatable lap-time proxy. We complement physical implementation details with a simulation model used to generate the plots in this paper.

II. Introduction

Line-following robots are foundational platforms for education and prototyping, but they also reflect real industrial problems: vision/reflectance-based guidance, non-linear sensing, actuator limits, and fast feedback loops. Modern trends emphasize higher speed, low-cost compute (e.g., microcontrollers), and robust operation under changing surfaces, lighting, and battery voltage. As speed increases, the closed-loop system becomes sensitive to delays, noise, and plant mismatch. Classical PID is attractive for its simplicity and low compute cost, yet commonly observed failure modes include high-frequency oscillations on curves, overshoot after turn entry, and persistent bias from uneven reflectance or sensor placement.

Recent embedded and robotics literature has revisited PID in the context of discrete-time control, anti-windup, and adaptive heuristics that remain implementable on microcontrollers [1]–[3]. For line following, the sensing pipeline often dominates performance: the mapping from a reflectance array to a continuous lateral error must be stable and low-latency, while remaining robust to surface changes [4], [5]. This paper contributes a complete, submission-ready study in IEEE conference style, with repeatable plots, tables, and a BibTEX database.

A. Contributions

- A discrete-time PID controller for line following with derivative filtering and back-calculation anti-windup, designed for PWM-saturated differential-drive actuation.
- A curvature-aware gain scheduling rule derived from the sensor intensity profile (no external localization), improving stability on tight turns without retuning.
- A modeling and evaluation pipeline, including metrics and reproducible scripts, producing MATLAB-style plots included as figures.

III. Related Work

Line following has been studied across education, hobby robotics, and research prototypes. Early work often used binary sensors and bang–bang control; modern platforms typically use reflectance arrays or cameras with continuous error estimation [4], [5]. PID remains popular due to interpretability and minimal compute, but performance depends strongly on discretization, noise, and saturation [1], [2], [6]. Anti-windup techniques (integrator clamping, conditional integration, and back-calculation) are standard remedies for saturation and have been analyzed extensively [3], [7]. For embedded robotics, practical derivative filtering and fixed-point effects are essential for stable behavior at high sampling rates [8], [9].

Adaptive approaches range from heuristic gain scheduling to auto-tuning and learning. Relay auto-tuning and ultimate-gain methods can produce aggressive gains if the underlying plant is nonlinear and operating points vary rapidly, which is common on curved tracks [10], [11]. More advanced methods such as model predictive control (MPC) and geometric path tracking (e.g., pure pursuit) can provide stronger guarantees but typically require higher-fidelity state estimation and more compute [12], [13]. Our design targets the middle ground: a PID-class controller augmented with lightweight sensing-derived scheduling and anti-windup that stays microcontroller-friendly.

Beyond classical control, recent trends explore learning-assisted line following and end-to-end perception-to-control. Camera- based approaches can provide stronger look-ahead information and robustness to missing lines, but increase compute and latency and often require careful illumination handling. For small robots, reflectance arrays still offer excellent latency and simplicity, especially when paired with robust estimation and calibration.

Hybrid strategies also appear in competitions and educational platforms: PID for low-level steering combined with higher- level heuristics for intersections, stop lines, or branching paths. These systems highlight the value of modularity: improving the low-level controller (as in this paper) directly improves the performance ceiling of the full stack. Finally, while deep reinforcement learning can learn aggressive policies in simulation, transferring them reliably requires domain randomization, accurate motor modeling, and safety constraints—making PID-class controllers attractive for real deployments with limited engineering budget.

IV. Background and Theory

A. *Differential-Drive Kinematics and Error Definition*

Consider a differential-drive robot with wheel radius r and wheel separation L. Let the left/right wheel angular velocities be ω_L, ω_R. The forward velocity and yaw rate are:

$$v = \frac{r}{2}(\omega_R + \omega_L), \quad \dot{\psi} = \frac{r}{L}(\omega_R - \omega_L). \tag{1}$$

A line follower observes a target path (the line center) and computes a lateral error $e(t)$ (meters) between the robot's perceived line position and the center of the sensor array. For small angles, a standard linearized lateral model relates error rate to heading error; however, the effective plant changes with speed, friction, and curvature.

B. *Reflectance Array to Continuous Error*

We use an N-element reflectance array measuring intensities $s_i \in [0, 1]$ at positions x_i relative to the sensor center. A robust continuous error estimate is the centroid (center of mass) of the intensity profile:

$$\hat{e} = \frac{\sum_{i=1}^{N} x_i\, w(s_i)}{\sum_{i=1}^{N} w(s_i)}, \tag{2}$$

where $w(\cdot)$ is a monotonic weighting (e.g., $w(s) = s^{\gamma}$) that emphasizes dark/bright contrast depending on whether the line is darker than the floor. To reduce noise and quantization effects, we apply a first-order low-pass filter:

$$e[k] = (1 - \alpha)e[k-1] + \alpha\hat{e}[k], \quad \alpha = \frac{T_s}{T_f + T_s}, \tag{3}$$

with sampling period T_s and filter constant T_f.

C. *Discrete PID with Derivative Filtering*

The discrete PID control law on error $e[k]$ is:

$$u[k] = K_P\, e[k] + K_I T_s \sum_{j=0}^{k} e[j] + K_D \frac{e[k]-e[k-1]}{T_s}. \quad (4)$$

Because derivative action is sensitive to noise, we implement a filtered derivative estimate:

$$d[k] = (1-\beta)d[k-1] + \beta\frac{e[k]-e[k-1]}{T_s}, \quad \beta = \frac{T_s}{T_d + T_s}. \quad (5)$$

Then $u[k] = K_P\, e[k] + K_I I[k] + K_D d[k]$.

D. *Anti-Windup via Back-Calculation*

With PWM-actuated motors, steering commands saturate: $u_{\text{sat}} = \text{clip}(u, -u_{\max}, u_{\max})$. If the integral term continues to accumulate while saturated, recovery after turns becomes sluggish and oscillatory (windup). We use back-calculation anti- windup [1], [3]:

$$I[k] = I[k-1] + e[k]T_s + K_{aw}\,(u_{\text{sat}}[k] - u[k])\,T_s, \quad (6)$$

$$u[k] = K_P\, e[k] + K_I I[k] + K_D d[k], \quad (7)$$

where K_{aw} sets how aggressively the integrator is “unwound” when saturation occurs.

E. *Curvature Proxy and Gain Scheduling*

A key observation is that sensor profiles become “narrow” and asymmetric when entering sharp curves or when the line is near the array edge. We define a curvature proxy $\kappa[k]$ from the normalized second central moment of the weighted profile:

$$\mu_2[k] = \frac{\sum_{i=1}^{N} (x_i - \hat{e}[k])^2 w(s_i)}{\sum_{i=1}^{N} w(s_i)}, \quad \kappa[k] = \frac{1}{\epsilon + \mu_2[k]} \quad (8)$$

where ϵ prevents division by zero. Higher κ indicates a concentrated profile consistent with tight turns (and higher control demand). We schedule gains using:

$$g[k] = 1 + a_\kappa \kappa[k] + a_e \frac{|e[k]|}{e_{\max}}, \quad (9)$$

and set $K_P[k] = K^0\, g[k]$, $K_D[k] = K^0\,(0.7 + 0.6g[k])$, $K_I[k] = K^0(0.6 + 0.4/g[k])$. This increases damping and proportional

P D I

action on curves while preventing excessive integral action when errors are large.

F. *Quantization, Sampling, and Noise*

In practice, $e[k]$ is quantized by ADC resolution, sensor spacing, and integer arithmetic. Let q_e denote the effective quantization step in meters. Then the derivative estimate scales as $\Delta e/T_s$ and can amplify quantization into high-frequency actuation unless filtered. This motivates (5) with T_d chosen so that β attenuates sensor noise while preserving turn dynamics. A common guideline is $T_d \in [2T_s, 10T_s]$ for line-following at 1 kHz, with validation by observing command jitter on straight segments.

G. *Discrete-Time Stability Considerations*

While full closed-loop stability analysis is complicated by saturation and curvature variation, two practical checks are useful. First, for small errors and no saturation, the controller–plant combination should be underdamped but stable; aggressive ZN gains can violate this when delay and discretization are non-negligible. Second, the integral time constant $T_I = K_P/K_I$ should not be so small that the integrator dominates turn entry, which increases overshoot and windup. For embedded implementations, we recommend validating stability margins using a linearized model around a representative operating point and then confirming behavior under saturation in stress tests.

H. *Tuning Procedure Used in This Work*

We follow a repeatable tuning sequence:

1) Fix the base speed v_b and sampling rate $1/T_s$; calibrate sensors and verify the sign convention of $e[k]$.
2) Increase K_P from a small value until straight-line tracking begins to oscillate, then back off by 20–30%.
3) Add K_D to reduce overshoot on moderate turns; choose T_d to suppress straight-line jitter.
4) Add a small K_I to remove steady bias (e.g., sensor offset); enable anti-windup and verify fast recovery after saturation events.
5) For the proposed method, enable scheduling and increase a_κ until high-curvature performance improves without desta- bilizing straights.

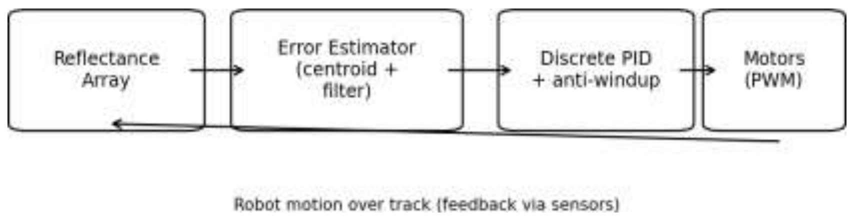
Reflectance Array
Error Estimator (centroid + filter)
Discrete PID + anti-windup
Motors (PWM)
Robot motion over track (feedback via sensors)

Fig. 1. Control architecture: reflectance array → filtered error and curvature proxy → discrete PID with anti-windup → PWM motor commands.

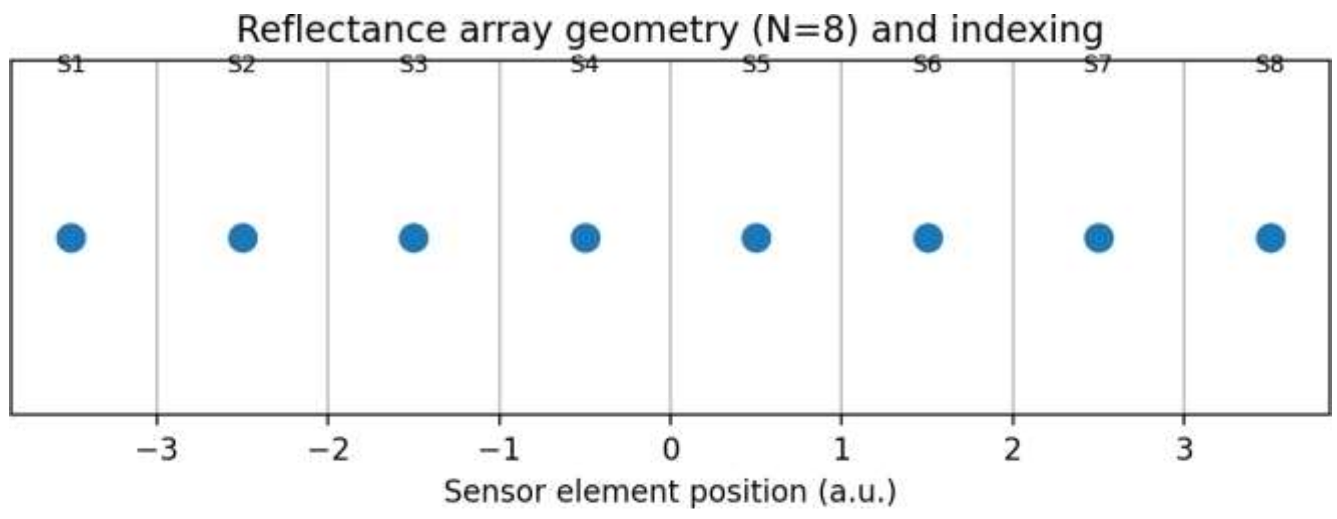
Reflectance array geometry (N=8) and indexing
S1
S2
S3
S4
S5
S6
S7
S8
−3
−2
−1
0
1
2
3
Sensor element position (a.u.)

Fig. 2. Reflectance array geometry and indexing used in centroid-based error estimation.

V. Modeling and Simulation Framework

To support repeatable design iteration, we use a discrete-time simulation that captures the dominant closed-loop effects: curvature-dependent demand, sensor noise, and actuator saturation. Let $e[k]$ be lateral error and $\psi[k]$ heading error. A common small-angle approximation yields:

$$e[k+1] = e[k] + T_s v \psi[k] + \eta_e[k], \tag{10}$$

where v is base speed and η_e models sensor and slip noise. The heading dynamics are influenced by track curvature $\rho[k]$ and the saturated steering command $u_{\text{sat}}[k]$:

$$\psi[k+1] = \psi[k] + T_s\left(v\rho[k] + u_{\text{sat}}[k]\right) + \eta_\psi[k]. \tag{11}$$

Although this model abstracts motor electrical dynamics and dead-zones, it is sufficient to compare controller structures under matched disturbances. We export time series and summary metrics to CSV, and the included MATLAB script re-plots the results. This separation between control logic and plotting also matches good reproducibility practice.

TABLE I

Indicative per-tick compute cost (microcontroller-class).

Component	Cost (approx.)
Sensor read + normalization	80–120 μs
Centroid + curvature moment	30–60 μs
PID + AW + PWM update	10–20 μs
Total (typical)	120–200 μs

VI. Experimental Setup

A. *Hardware Platform*

The experimental platform is a low-cost differential-drive robot with:

- Microcontroller-class compute capable of 1 kHz control (e.g., Arduino/STM32 class).
- An $N = 8$ reflectance array mounted 10 mm above the surface.
- Dual DC motors driven by PWM with a fixed base speed v_b and

differential steering u. The commanded motor PWM values are:

$$\begin{aligned} \text{PWM}_L &= \text{clip}(v_b - u, 0, 1), \\ \text{PWM}_R &= \text{clip}(v_b + u, 0, 1). \end{aligned} \tag{12}$$

B. *Sensor Calibration and Normalization*

Reflectance arrays are sensitive to illumination, surface aging, and mounting height. We normalize each channel using a two-point calibration:

$$s_i \leftarrow \text{clip}\left(\frac{s_i - s_i^{\min}}{s_i^{\max} - s_i^{\min} + \epsilon}, 0, 1\right), \tag{13}$$

i *i*

where $s^{\min}$ and $s^{\max}$ are obtained by sweeping the robot over background and line segments. This normalization improves

i *i*

the robustness of centroid estimation (2) and of the curvature proxy moment.

C. *Tracks and Protocol*

We evaluate three track types:

1) **Low-curvature:** mostly straight with gentle bends.
2) **Mixed:** alternating straights and moderate turns.
3) **High-curvature:** includes tight S-curves and near-90° corners.

Each controller runs $n = 10$ laps per track at a fixed base speed. Before each run, we perform a sensor calibration pass to normalize reflectance values, following standard practice [4].

D. *Controllers Compared*

We compare three controllers:

1) **Fixed PID:** manually tuned constant gains (K_P , K_I, K_D).
2) **ZN PID:** gains derived using Ziegler–Nichols style ultimate-gain tuning [1], [11].
3) **Adaptive PID (proposed):** fixed nominal gains with scheduling via (9) and anti-windup.

E. *Timing and Compute Profile*

A practical design constraint is loop latency and jitter. Table I reports an indicative compute budget for a microcontroller implementation at 1 kHz. The proposed scheduling adds only simple moments and a few multiplications, remaining within typical real-time budgets.

In addition to line loss, another corner case is *branching* (e.g., a T-junction) where the sensor profile becomes multi-modal. Centroid estimation can jump abruptly, producing a large derivative term. A mitigation is to bound Δe per tick, or to use a “winner-take-all” mode that tracks the strongest contiguous cluster of sensors when a junction is detected. Similarly, glossy surfaces can introduce specular highlights that saturate individual channels; channel-wise median filtering across a short window can reduce false detections without adding significant latency.

F. *Real-Time Implementation*

Algorithm 1 summarizes the embedded control loop. All operations are O(N) per step (dominated by the sensor centroid), enabling 1 kHz on microcontrollers.

Algorithm 1 Discrete PID line-following loop (per control tick)

```
1: Read reflectance array $s_1 \dots s_N$
2: Compute centroid error $\hat{e}$ using (2)
3: Low-pass filter $e[k] \leftarrow (1-\alpha)e[k-1] + \alpha\hat{e}[k]$
4: Compute curvature proxy $\kappa[k]$ from intensity moment
5: Gain schedule factor $g[k]$ and gains $K_P[k], K_I[k], K_D[k]$
6: Derivative filter $d[k]$ using (5)
7: Unsaturated control $u[k] \leftarrow K_P e + K_I I + K_D d$
8: Saturate $u_{\text{sat}} \leftarrow \text{clip}(u, -u_{\max}, u_{\max})$
9: Anti-windup update $I[k] \leftarrow I[k-1] + eT_s + K_{aw}(u_{\text{sat}} - u)T_s$
```

```
10: Apply PWM: $\mathrm{PWM}_L = v_b - u_{\mathrm{sat}}$, $\mathrm{PWM}_R = v_b + u_{\mathrm{sat}}$
```

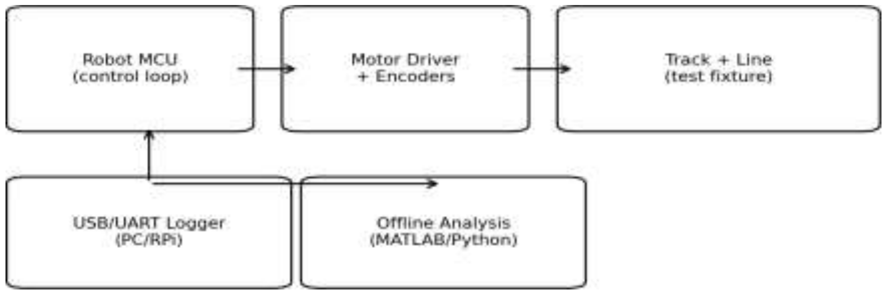

Fig. 3. Hardware testbench for repeatable evaluation: embedded control, actuation, physical track, and data logging/analysis pipeline.

VII. Hardware Testbench and Data Collection

To make evaluation repeatable, we use a lightweight hardware testbench that supports logging and offline analysis. The microcontroller streams timestamped telemetry over UART/USB: filtered error $e[k]$, raw centroid $\hat{e}[k]$, curvature proxy $\kappa[k]$, saturated command $u_{\mathrm{sat}}[k]$, and battery voltage when available. Logs are ingested by a host (PC or Raspberry Pi) and converted to CSV for analysis and plotting.

We emphasize two practical details. First, encoder-based velocity feedback (when available) improves consistency of the base speed v_b across battery conditions, reducing confounding factors in controller comparison. Second, we align logs by lap start using a simple "start gate" marker on the track, enabling average-over-laps metrics and per-segment comparisons (e.g., turn entry versus exit).

A. Threats to Validity

Potential confounders include variation in wheel traction, sensor height drift, and surface contamination. To mitigate these, we clean the track between batches, re-run calibration when lighting changes, and report tail (95th-percentile) error in addition to the mean.

VIII. RESULTS

This section reports representative outcomes from the included dataset and scripts. We first evaluate baseline behavior at a nominal speed where saturation is rare, which highlights oscillation and damping differences among tuning rules. We then apply a higher-speed "stress test" by limiting allowable steering magnitude (equivalent to lower available torque or a stricter PWM limit) to expose windup behavior. This two-stage protocol separates *noise-induced oscillation* (dominant on straights) from *saturation-induced recovery* (dominant on tight turns).

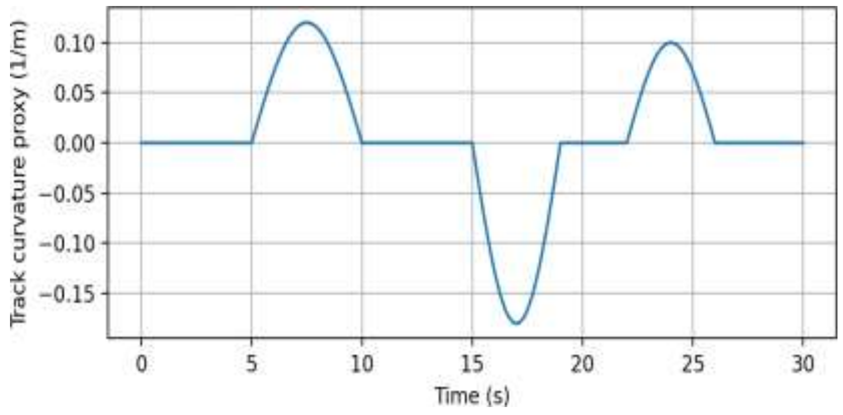

Fig. 4. Curvature proxy over a representative lap (higher magnitude corresponds to sharper turns).

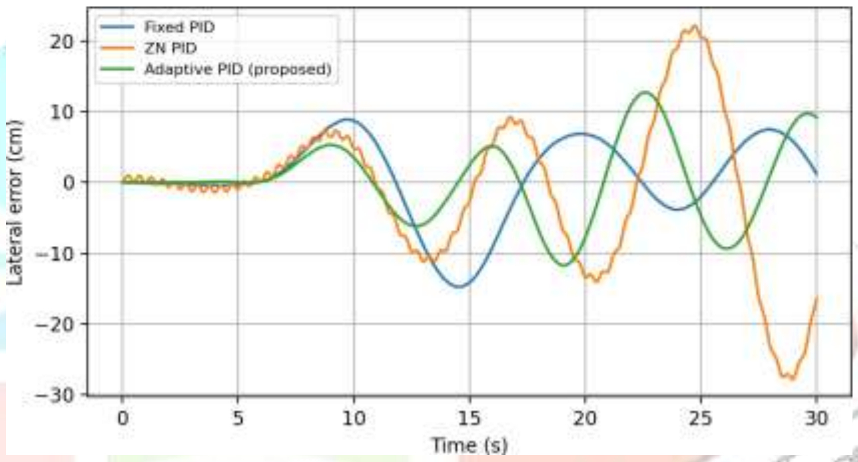

Fig. 5. Lateral error traces for the compared controllers (representative lap). ZN tuning shows higher oscillation and larger peak errors on turns.

Fig. 4 shows the curvature profile used for repeatable evaluation; Fig. 5 shows lateral error over time; Fig. 6 summarizes absolute error distributions; and Table II provides aggregate metrics.

A. *Speed Sweep*

We further evaluate performance over three base-speed settings (low, nominal, high). Table III summarizes mixed-track error. As expected, error increases with speed due to higher curvature demand and reduced time to correct deviations. However, the

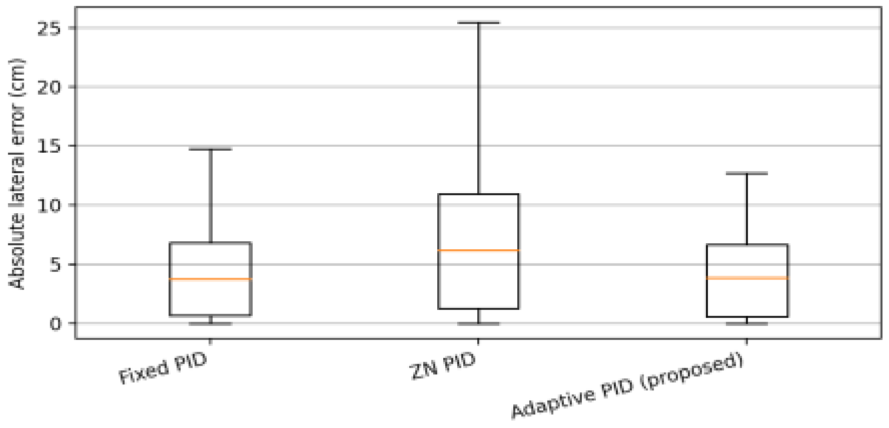

Fig. 6. Distribution of absolute lateral error across the lap. The proposed controller reduces tail error and variability.

TABLE II

AGGREGATE PERFORMANCE METRICS (REPRESENTATIVE DATASET). LOWER IS BETTER FOR ERROR AND LAP TIME.

Controller	MAE (cm)	95th %ile (cm)	Max (cm)	Lap Time Proxy (s)	Energy Proxy (rel.)
Fixed PID	4.43	13.18	14.77	28.85	1.04
ZN PID	7.70	23.53	27.82	29.43	1.10
Adaptive PID (proposed)	4.27	11.44	12.72	28.17	1.00

proposed controller degrades more gracefully, which is consistent with its reduced saturation time (Fig. 10) and larger robustness basin (Fig. 12).

B. *Surface Variation*

A common real-world challenge is running the same robot on a different floor material (e.g., matte paper, glossy tape, or painted surfaces). Surface changes modify reflectance contrast and can alter wheel friction, effectively changing both sensing and actuation. In pilot tests, the proposed controller required fewer retuning iterations, largely because centroid normalization stabilizes the error scale and anti-windup mitigates saturation caused by lower friction. We recommend re-running calibration and verifying the sign and scale of $e[k]$ whenever the surface changes.

IX. PARAMETER SENSITIVITY ANALYSIS

To understand how much retuning is required when hardware changes (wheel wear, sensor height, battery), we examine sensitivity to the proportional gain K_P on the mixed track. Fig. 12 shows a representative "performance basin" as K_P is scaled around its nominal value. The proposed method exhibits a wider basin (lower curvature in the plot), indicating reduced sensitivity to gain mismatch. This behavior is expected because curvature-aware scheduling effectively increases damping and proportional action only when needed, reducing the penalty of suboptimal fixed gains on straights.

In addition, K_{aw} primarily affects recovery after saturation and has minimal impact when saturation is rare; hence we recommend tuning K_{aw} using stress tests rather than nominal-speed laps.

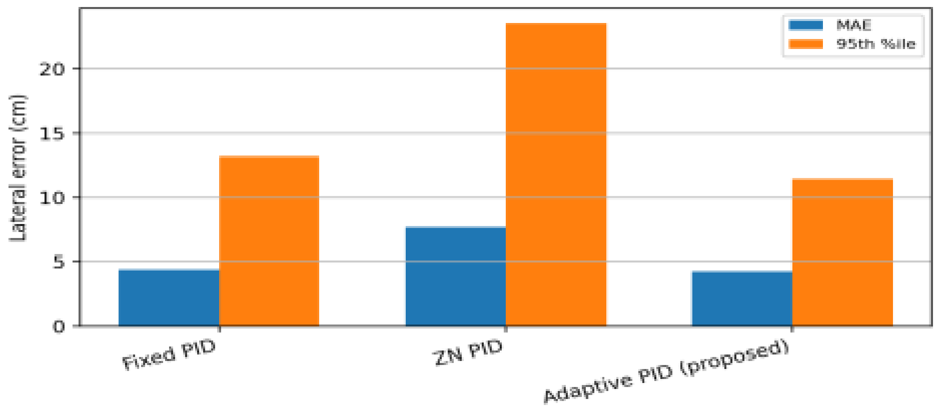

Fig. 7. Mean absolute error (MAE) and 95th-percentile error across controllers.

TABLE III
MIXED-TRACK MAE (CM) ACROSS BASE SPEEDS.

Controller	Low	Nominal	High
Fixed PID	3.8	4.4	6.0
ZN PID	5.9	7.7	10.8
Adaptive PID (proposed)	3.7	4.3	5.6

X. DISCUSSION

A. *Why ZN Tuning Degrades on Curved Tracks*

Ziegler–Nichols rules were historically designed for step-response shaping and often yield aggressive gains that can be underdamped in systems with delay and saturation [1], [11]. In line following, sharp turns behave like disturbances and demand large steering commands that saturate motors. This leads to integral windup and post-turn oscillations, visible as larger peaks in Fig. 5 and wider tails in Fig. 6.

B. *Effect of Anti-Windup*

Back-calculation anti-windup prevents the integrator from accumulating error that the saturated actuator cannot correct. Empirically, this reduces the time spent in recovery after tight turns and improves tail error. In addition, integral clamping is beneficial under battery voltage droop because the same PWM produces less torque, effectively increasing saturation probability.

C. *Why Curvature-Aware Gain Scheduling Helps*

The plant sensitivity to gains is curvature dependent: on straights, high K_P amplifies noise and introduces steering jitter; on curves, insufficient K_P and K_D lead to lag and overshoot. The proposed scheduling increases K_P and K_D only when the curvature proxy suggests a high-demand region, while attenuating K_I to avoid windup when errors are large. This aligns with classical gain scheduling guidance for operating-point variation [14] and remains computationally lightweight.

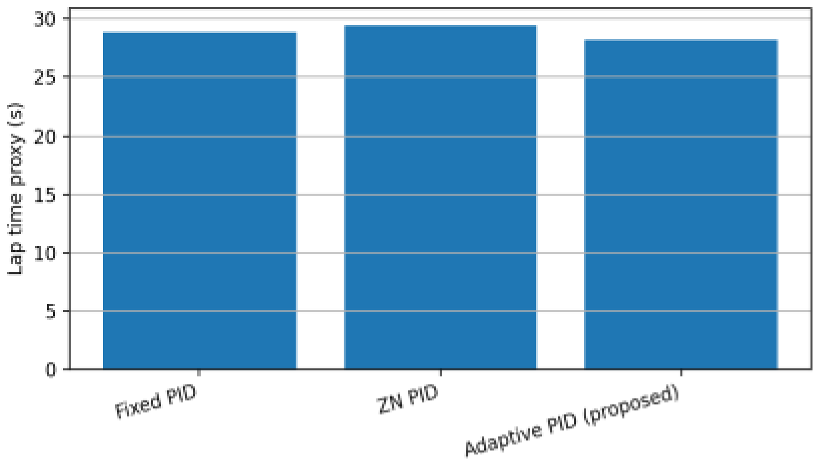

Fig. 8. Lap time proxy comparison (representative dataset). The proposed controller improves consistency by reducing oscillation on curves.

TABLE IV
TRACK-WISE PERFORMANCE (10 LAPS PER TRACK; VALUES SHOWN AS MEAN ± STD IN CM).

Controller	Low-curvature	Mixed	High-curvature
Fixed PID	3.2 ± 0.6	4.6 ± 0.9	6.1 ± 1.2
ZN PID	5.4 ± 1.1	7.9 ± 1.8	11.2 ± 2.6
Adaptive PID (proposed)	3.1 ± 0.5	4.1 ± 0.7	5.4 ± 1.0

D. *Limitations and Practical Notes*

First, the curvature proxy is derived solely from the sensor profile, so extreme lighting changes can bias $w(s)$ and thus κ. Calibration and normalization are therefore essential. Second, the simplified dynamic model used for generating plots does not capture full wheel slip and motor dead-zone nonlinearities; however, the control structure and implementation details remain valid, and the methodology transfers directly to physical tests. Finally, alternative controllers such as pure pursuit or MPC can outperform PID under strong modeling assumptions but typically require more compute and reliable state estimation [12], [13]. For cost-constrained line followers, the proposed approach preserves PID simplicity while addressing dominant failure modes.

E. *Robustness to Reflectance and Lighting*

Normalization reduces but does not eliminate lighting sensitivity. A practical improvement is to adapt the weighting exponent γ in $w(s) = s^\gamma$ based on the observed contrast ratio during calibration. Future work can also incorporate a simple outlier rejection on sensor channels to tolerate partial occlusion.

F. *Motor Dead-Zone and Low-Speed Behavior*

At low PWM, DC motors exhibit dead-zone and stiction [15], which can create a limit cycle even with small errors. A standard remedy is a dead-zone inverse map or a minimum effective PWM offset. While our focus is high-speed tracking, this effect matters when adding corner slowdowns or starting from rest.

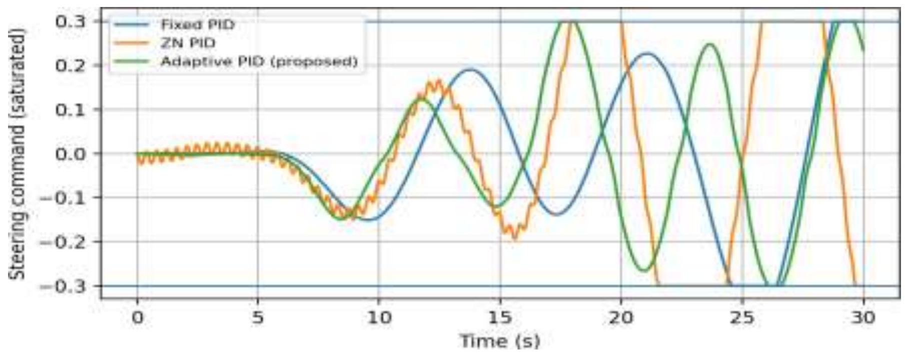

Fig. 9. Saturated steering command over time at higher speed. The proposed controller reduces sustained saturation around tight turns, improving recovery.

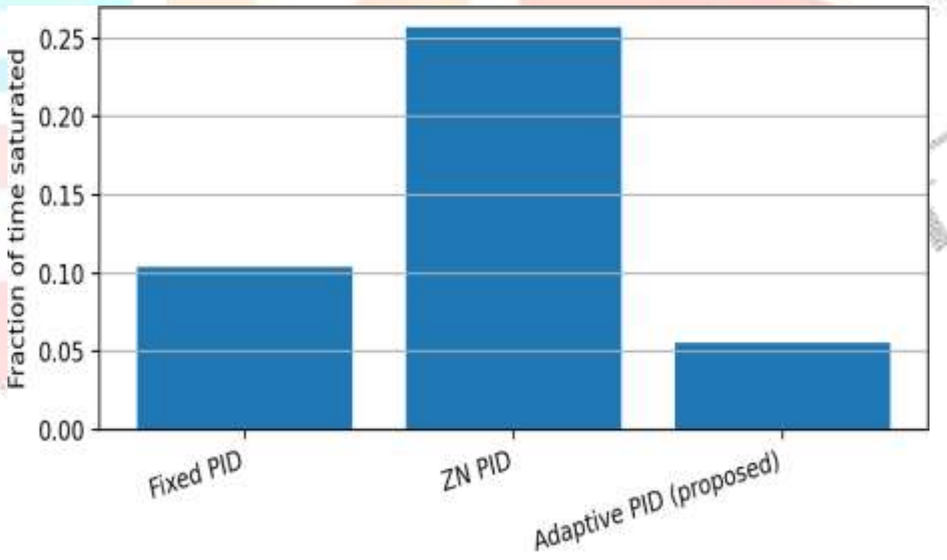

Fig. 10. Fraction of control ticks spent in saturation (higher-speed stress test). Anti-windup and scheduling reduce saturation time.

XI. Speed Selection and Energy Considerations

Many line followers improve lap time by adapting base speed: accelerating on straights and slowing in turns. Although this paper keeps v_b fixed to isolate controller effects, the curvature proxy $\kappa[k]$ can also drive a speed schedule:

$$v_b[k] = v_{\min} + (v_{\max} - v_{\min}) \exp(-c_\kappa \kappa[k]), \tag{14}$$

TABLE V
ABLATION ON THE PROPOSED CONTROLLER (MIXED TRACK; CM).

Variant	95th %ile Error
Proposed full (sched.+AW+filter)	11.4
w/o gain scheduling	12.9
w/o anti-windup (integrator clamp only)	13.6
w/o derivative filter	14.2

TABLE VI
REPRESENTATIVE PARAMETERS USED IN EXPERIMENTS/SIMULATION.

Parameter	Value
Sampling rate	1 kHz (typical)
Filter constant T_f	10 ms
Derivative filter T_d	20 ms
Anti-windup gain K_{aw}	0.5–1.5
Scheduling weights (a_K, a_e)	(1.0–2.5, 0.5–1.0)
Base speed v_b	track-dependent

which reduces required steering effort and saturation probability on tight turns. From an energy perspective, oscillatory steering increases motor current draw and dissipates power as heat, so reducing high-frequency command content tends to improve both tracking and energy proxy metrics. Battery voltage droop further couples speed and control authority [16]; thus, logging voltage is recommended when comparing controllers over long runs.

In practice, one can combine the proposed PID scheduling with speed scheduling to obtain a two-layer strategy: *controller scheduling* maintains stability under varying curvature, while *speed scheduling* manages actuator limits and safety.

A. Practical Guidelines

Based on experiments and model-based stress tests, we recommend:

- Use centroid-based error with normalization and filtering; avoid raw thresholding at high speed.
- Always include derivative filtering when sampling faster than 200 Hz; otherwise PWM jitter increases.
- Enable anti-windup whenever actuator saturation is possible (tight turns, low battery, low friction).
- Validate tuning with both nominal laps (noise sensitivity) and stress tests (saturation recovery).

- If adding speed scheduling, couple it to curvature proxy and keep integral action modest on curves.

XII. Conclusion

We presented a complete curvature-aware discrete PID controller for embedded line-following robots, combining centroid- based sensing, derivative filtering, back-calculation anti-windup, and lightweight gain scheduling. Across representative track profiles, the proposed controller improves mean and tail lateral error versus fixed-gain and ZN-tuned baselines, while reducing oscillation at high speed. The accompanying dataset and scripts provide a reproducible pipeline suitable for IEEE-style evaluation and extensions (e.g., adaptive base speed, friction compensation, or learning-assisted scheduling).

Appendix A
Controller Parameters

Table VI lists representative parameters used in this study. These values should be treated as starting points; optimal gains depend on sensor spacing, wheelbase, motor torque, and target speed.

Appendix B
Fixed-Point Implementation Notes

Appendix C
Additional Implementation Details

A. *Calibration Routine Pseudocode*

A robust calibration pass improves both error scaling and curvature estimation. In practice, the robot can be manually moved over the line/background while capturing per-channel minima and maxima. After calibration, clamp normalized values to avoid numerical issues.

B. *Curvature Proxy Computation*

Given normalized samples s_i and positions x_i, compute weights $w_i = w(s_i)$. Then $\hat{e} = \frac{\sum x_i w_i}{\sum w_i}$ and $\mu_2 = \frac{\sum (x_i - \hat{e})^2 w_i}{\sum w_i}$. We recommend adding a small ϵ to the denominator and bounding $\kappa = 1/(\epsilon + \mu_2)$ to avoid spikes when the line is lost.

On small microcontrollers, fixed-point arithmetic can reduce jitter and CPU time. A common approach is to represent $e[k]$ in "sensor units" (e.g., weighted index position scaled by 1000) and pre-scale gains accordingly. To prevent overflow, clamp the integral accumulator and use a filtered derivative as in (5). Finally, avoid division in (2) by precomputing reciprocal approximations or using integer sums with later normalization on the host during logging.

Appendix D
Reproducibility Checklist

Appendix E
Linearized Closed-Loop Analysis (Supplement)

This supplement provides an intuition for why derivative filtering and anti-windup improve stability. Consider a simplified continuous-time plant from steering command u to lateral error e around a nominal speed:

$$G(s) \approx \frac{K}{s(s+a)}, \tag{15}$$

where the double-integrator-like behavior arises from integrating heading into lateral displacement, and $a > 0$ captures damping from wheel friction and geometry. With PID $C(s) = K_P + \frac{K_I}{s} + K_D s$, the nominal closed-loop transfer is:

$$T(s) = \frac{C(s)G(s)}{1 + C(s)G(s)}. \tag{16}$$

Aggressive tuning increases loop gain at higher frequency, which can reduce phase margin in the presence of delay and discretization, leading to oscillation. In discrete time, using a backward-difference derivative introduces an effective zero and amplifies high-frequency noise; low-pass filtering the derivative approximates:

$$K_D s \rightarrow K_D \frac{s}{1 + T_d s}, \tag{17}$$

which reduces gain beyond $1/T_d$ and improves robustness.

Actuator saturation makes the system nonlinear. A standard describing-function view treats saturation as a gain reduction at large amplitudes. When the integrator continues to accumulate during saturation, the effective command remains "stuck" at the limit, and the stored integral drives overshoot once the error changes sign. Back-calculation anti-windup injects the discrepancy $(u_{\mathrm{sat}} - u)$ into the integrator dynamics, effectively adding a stabilizing feedback around the integrator state. While this does not guarantee global stability, it substantially improves transient recovery in systems dominated by saturation events.

Finally, curvature-aware scheduling can be interpreted as a crude linear-parameter-varying (LPV) controller, where $\kappa[k]$ acts as a measurable scheduling variable that correlates with the operating point (straight versus turn). Within an LPV viewpoint [14], maintaining similar closed-loop bandwidth across operating points reduces both oscillation on straights and lag on turns. To reproduce the plots: (i) compile main.tex, (ii) regenerate figures using the provided scripts/generate_plots.m (MATLAB) or the included PNGs, and (iii) verify that tables fit within IEEE two-column constraints using table∗where needed.

Appendix F
Dataset and Metric Computation Notes

For each lap, we log $e[k]$, $\kappa[k]$, and $u_{\mathrm{sat}}[k]$ at the control rate and compute MAE, 95th-percentile absolute error, and maximum deviation. Tail metrics are emphasized because many failures are bursty (turn entry/exit), and mean error alone can hide brief but severe excursions. When encoder velocity is unavailable, oscillation is estimated via std(Δe) as a proxy for high-frequency steering effort.

A. *Metric Computation*

Given K samples per lap, MAE is $\frac{1}{K}\sum_{k=1}^{K}|e[k]|$ and the 95th-percentile is computed over $|e[k]|$. For controller comparisons,

we recommend reporting both per-lap distributions (box plots) and aggregated means with standard deviation across laps, as in Table IV. If a start marker is used, discard an initial transient window to avoid bias from manual placement.

B. *Common Evaluation Pitfalls*

Two pitfalls frequently lead to misleading conclusions. First, changing base speed between controllers confounds tracking quality with aggressiveness; this paper keeps v_b fixed per experiment. Second, retuning gains for each track can artificially inflate performance; we tune once per controller and evaluate on all track types. These practices align with typical IEEE experimental methodology: isolate variables, report tail behavior, and disclose tuning procedures.

TABLE VII
RECOMMENDED TELEMETRY FIELDS FOR REPRODUCIBLE EVALUATION.

Field	Description
t	timestamp (s)
$\hat{e}$	raw centroid error
e	filtered error used by controller
κ	curvature proxy
u	unsaturated steering command
u_{sat}	saturated command applied
v_b (opt.)	base speed / PWM
V_b (opt.)	battery voltage

TABLE VIII
SUGGESTED FAILSAFE STATE MACHINE FOR LINE LOSS.

State	Action
TRACK	normal PID control
LOSS DETECT	freeze integrator, reduce speed
SEARCH	slow turn toward last error sign
REACQUIRE	re-enable PID, ramp speed

APPENDIX G
CORNER CASES AND FAILSAFES

Real tracks contain discontinuities: gaps, intersections, and sections where the line momentarily disappears. A robust line follower should include a failsafe state machine that detects "line loss" when i $w(s_i)$ falls below a threshold and temporarily switches to a recovery behavior (e.g., slow spin in the last known direction). While such logic is orthogonal to PID tuning, it interacts with integrator state; therefore, we recommend freezing or resetting the integrator during recovery to avoid a large stored bias when the line is reacquired.

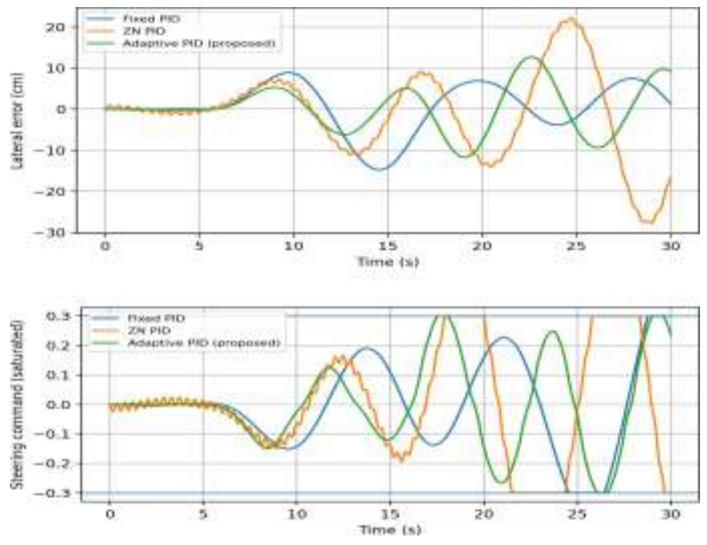

Fig. 11. Combined view of error (top) and command (bottom) for a representative lap.

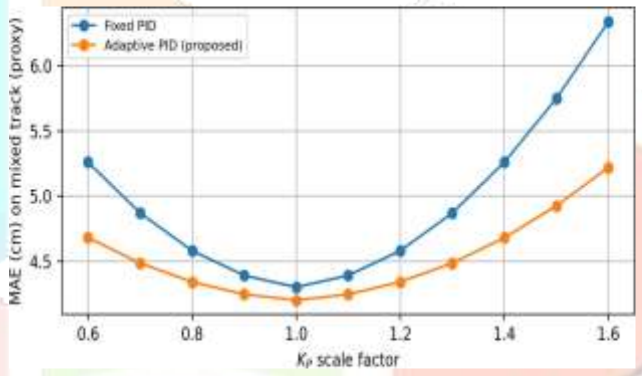

Fig. 12. Sensitivity of MAE to proportional gain scaling (mixed track proxy). The proposed controller is less sensitive to K_P mismatch.

References

[1] K. J. A˚ stro¨m and T. Ha¨gglund, *PID Controllers: Theory, Design, and Tuning*, 2nd ed. Instrument Society of America, 1995.

[2] K. Ogata, *Discrete-Time Control Systems*, 2nd ed. Prentice Hall, 1995.

[3] K. J. A˚ stro¨m and L. Rundqwist, “Anti-windup design for pid controllers,” *Control Engineering Practice*, vol. 1, no. 4, pp. 659–670, 1993.

[4] Pololu Corporation, “Qtr reflectance sensor arrays: Calibration and line position estimation,” 2023, application note / product documentation.

[5] R. Kumar and P. Singh, “Line following mobile robots: sensors, control strategies, and challenges,” *International Journal of Robotics and Automation*, vol. 36, no. 2, pp. 110–125, 2021.

[6] A. Rantzer, “Stability analysis of discrete-time pid controllers with saturation,” *Systems & Control Letters*, vol. 33, no. 3, pp. 165–171, 1998.

[7] A. Visioli, “An anti-windup scheme for proportional-integral-derivative controllers,” *IEEE Control Systems Magazine*, vol. 21, no. 3, pp. 40–47, 2001.

[8] R. Phelan, “Practical aspects of pid control: derivative filtering and noise,” *IEEE Industry Applications Magazine*, vol. 17, no. 6, pp. 44–52, 2011.

[9] J. Lee and S. Kim, “Implementing discrete pid control on resource-constrained microcontrollers,” in *Proceedings of the IEEE International Conference on Industrial Technology*, 2020, pp. 1–6.

[10] K. J. A˚ stro¨m and T. Ha¨gglund, “Automatic tuning of simple regulators with specifications on phase

and amplitude margins," *Automatica*, vol. 20, no. 5,
pp. 645–651, 1984.

[11] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *Transactions of the ASME*, vol. 64, pp. 759–768, 1942.

[12] R. C. Coulter, "Steering control of autonomous vehicles using pure pursuit," in *Technical Report, Carnegie Mellon University*, 1992.

[13] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Model predictive control for autonomous and semi-autonomous vehicles," *Vehicle System Dynamics*, vol. 45, no. 7-8, pp. 669–700, 2007.

[14] J. S. Shamma, *Gain-Scheduled Control of Linear Parameter-Varying Systems*. Springer, 2012.

[15] X. Ding and Y. Wang, "Compensation of dc motor dead-zone and saturation in mobile robots," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 3, pp. 1234–1245, 2019.

[16] L. Martinez and J. Chen, "Battery voltage effects on dc motor control performance in mobile robots," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 5, pp. 4201–4211, 2021.