https://doi.org/10.31436/iiumej.v27i3.4508

Budget-Efficient PID Tuning of a Line-Following Differential-Drive AGV Using Extra Trees Surrogate-Assisted CMA-ES

MASTANG[1], BAKR AHMED TAHA[1], RETNA APSARI[2,3], MUHAMMAD MUKHLISIN[4], ENI DWI WARDIHANI[5], NORHANA ARSAD[1,2*]

[1]*Dept. of Electrical, Electronic & Systems Engineering, Universiti Kebangsaan Malaysia, 43600, Selangor, Malaysia*
[2]*Dept. of Engineering, Faculty of Advanced Technology and Multidiscipline, Airlangga University, 60115, Surabaya, Indonesia*
[3]*Dept. of Physics, Faculty of Science and Technology, Airlangga University, 60115, Surabaya, Indonesia*
[4]*Dept. of Civil Engineering, Faculty of Engineering, Semarang State Polytechnic, 50275, Semarang, Indonesia*
[5]*Dept. of Electrical Engineering, Faculty of Engineering, Semarang State Polytechnic, 50275, Semarang, Indonesia*

**Corresponding author: noa@ukm.edu.my*

(Received: 7 June 2026; Accepted: 22 August 2026; Published online: 11 September 2026)

ABSTRACT: Physical PID tuning for line-following automated guided vehicles (AGVs) is sample-inefficient because the gain-to-performance relationship is nonlinear and noisy, while every candidate must be executed on the real platform. Conventional population optimizers can therefore spend a limited experimental budget on unproductive trials, and evidence for surrogate-assisted CMA-ES under explicit true-evaluation accounting on embedded AGVs remains limited. This study evaluates an Extra Trees surrogate-assisted CMA-ES supervisory tuner on a single line-following differential-drive AGV. CMA-ES generates a virtual gain population in normalized space, and Extra Trees ranks it using the predicted objective and inter-tree dispersion; only a small, ranked subset undergoes physical evaluation. The objective combines integral absolute error, root mean square error, overshoot, actuation smoothness, and a lost-line penalty. We compared the method with pure CMA-ES, particle swarm optimization, and a genetic algorithm on Circle and 8Shape trajectories, using an identical budget of 57 true evaluations for every method, seed, and trajectory. The proposed method achieved the lowest mean final-best objective on both trajectories and improved on the strongest baseline by 30.8% on Circle and 20.8% on 8Shape. Cross-run distributions, empirical cumulative distribution functions, and time-domain responses support the same ranking. These results show that surrogate screening can improve sample efficiency for PID tuning on the tested embedded platform.

ABSTRAK: Penalaan PID secara fizikal bagi kenderaan berpandu automatik (AGV) pengikut garisan tidak cekap dari segi sampel kerana hubungan antara gandaan dan prestasi adalah tak linear serta bising, sedangkan setiap calon perlu diuji pada platform sebenar. Pengoptimum berasaskan populasi konvensional boleh menggunakan bajet eksperimen yang terhad pada percubaan yang kurang produktif, dan bukti bagi CMA-ES berbantukan surrogat dengan perakaunan penilaian sebenar yang jelas pada AGV terbenam masih terhad. Kajian ini menilai penala penyeliaan CMA-ES berbantukan surrogat Extra Trees pada sebuah prototaip AGV pacuan kebezaan pengikut garisan. CMA-ES menjana populasi gandaan maya dalam ruang ternormal dan Extra Trees menyusunnya menggunakan objektif ramalan dan sebaran antara pokok, manakala hanya subset kecil dengan skor terbaik menjalani penilaian fizikal. Fungsi

https://doi.org/10.31436/iiumej.v27i3.4508

objektif menggabungkan ralat mutlak bersepadu, ralat punca min kuasa dua, limpahan, kelancaran aktuasi, dan penalti kehilangan garisan. Kaedah ini dibandingkan dengan CMA-ES tulen, pengoptimuman kawanan zarah, dan algoritma genetik pada trajektori Circle dan 8Shape menggunakan bajet yang sama, iaitu 57 penilaian sebenar bagi setiap kaedah, benih rawak, dan trajektori. Kaedah yang dicadangkan mencapai purata objektif terbaik akhir terendah pada kedua-dua trajektori serta mengatasi garis dasar terbaik sebanyak 30.8% pada Circle dan 20.8% pada 8Shape. Taburan antara larian, fungsi taburan kumulatif empirik, dan respons domain masa menyokong kedudukan yang sama. Dapatan ini menunjukkan bahawa penapisan surrogat dapat meningkatkan kecekapan sampel bagi penalaan PID pada platform terbenam yang diuji.

KEYWORDS: *Automated Guided Vehicle, PID tuning, CMA-ES, Extra Trees Surrogate, Fixed-Budget Optimization.*

1. INTRODUCTION

Automated guided vehicles (AGVs) are a core technology in modern manufacturing and warehousing intralogistics. On production lines, they transport raw materials, work-in-process, components, tools, and finished products among storage locations, workstations, assembly cells, inspection points, and shipping areas. Their role extends beyond replacing manually pushed trolleys, as AGVs can coordinate material flow, reduce dependence on repetitive handling tasks, and connect production planning with physical logistics. Recent reviews therefore identify AGVs as an important element of smart manufacturing systems that integrate equipment, storage, and human work areas [1, 2].

AGVs are most effective when they can follow a planned route repeatedly with minimal supervision. In manufacturing, many vehicles operate on fixed or semi-fixed paths defined by magnetic tape, painted lines, QR codes, or virtual routes. Even when the global route is predetermined, the vehicle must maintain accurate local tracking because small lateral deviations can develop into oscillation or line-loss events. In this study, a line-following differential-drive AGV is used as a controlled experimental platform for local tracking, not as a proxy for every AGV architecture. A 15-sensor array estimates the lateral position of the line, and the controller maps the resulting error to differential left-right motor commands. Accordingly, the conclusions are restricted to this platform and the tested operating conditions.

Despite significant advances in sensors, mapping, reinforcement learning, and predictive control models, PID controllers are still relevant in AGV applications. PIDs are simple, computationally lightweight, and can be implemented on low-cost embedded hardware. The proportional component reacts to current errors, the integral component compensates for persistent bias, and the derivative component reduces rapid changes. For small AGVs operating on Raspberry Pi or microcontroller-class devices, PIDs provide a practical control law with low memory requirements and latency [3]. In real-world manufacturing, this simplicity matters because maintenance technicians must understand controller behavior, diagnose instabilities, and retune parameters when loads, floor friction, motor response, or sensor heights change. Therefore, PIDs often serve as inner-loop controllers, while more advanced algorithms act as supervisory tuners or trajectory planners [4].

Fig. 1 illustrates the classical closed-loop PID structure. The reference signal $r(t)$ defines the desired centerline, while the measured output $y(t)$ is obtained from the AGV sensor system. The tracking error $e(t)$ is processed by the proportional, integral, and derivative terms, whose sum produces the motor-control signal $u(t)$. A high proportional gain can speed up correction but can also induce oscillation. A large integral gain can remove persistent bias but increase

https://doi.org/10.31436/iiumej.v27i3.4508

windup, whereas a large derivative gain can damp rapid error changes but amplify sensor noise. PID tuning therefore requires a balance among tracking accuracy, transient behavior, command smoothness, and recovery from line-loss events.

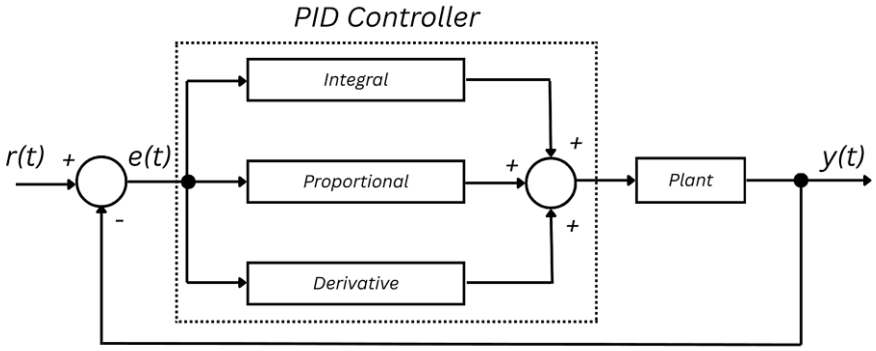


Figure 1. Classical closed-loop PID structure used as the conceptual basis for AGV line tracking.

The challenge of PID tuning for AGVs is the relationship between PID gain and tracking performance. It is nonlinear, noisy, and highly dependent on operating conditions. A gain set that works well on a smooth circle track may not remain optimal on an 8Shape track because the vehicle experiences alternating curvatures, sharper transitions, and different motor saturation patterns. Tracking sensors can also produce uncertainty. When a line is only partially detected, the estimated error can spike. When the line is lost, the controller must recover position without generating aggressive steering commands. Furthermore, AGVs are constrained by their onboard computing power and limited experiment budget. Each true evaluation requires operating the vehicle for a specified duration, recording sensor responses, calculating performance metrics, and resetting the experiment. Such evaluations are more expensive than surrogate predictions.

Classic empirical tuning methods such as manual tuning, Ziegler-Nichols, and relay feedback are attractive due to their simplicity. However, their performance is often limited when the objective combines multiple criteria such as error magnitude, overshoot, smoothness, and line-loss stability. Population-based optimizers, including genetic algorithm (GA), particle swarm optimization (PSO), differential evolution (DE), and beetle antennae search (BAS), can search nonlinear PID space without requiring gradients [5]. However, these methods generally require many true evaluations. In tight budget settings, candidate exploration can be inefficient because many gains are tested. CMA-ES is robust for continuous black-box optimization because it learns the covariance distribution over promising regions [6]. However, pure CMA-ES also consumes true evaluations for each candidate, making it expensive for AGV experiments.

Surrogate-assisted optimization offers a suitable compromise. A surrogate approximates the objective from evaluated samples, allowing an optimizer to screen many candidates before submitting a selected subset to high-fidelity evaluation [7, 8]. Extra Trees is attractive in this study because its randomized regression-tree ensemble can represent nonlinear interactions among K_p, K_i, and K_d, does not require feature scaling, trains quickly on small tabular datasets, and provides inter-tree prediction dispersion. The ensemble mean estimates candidate quality, while the dispersion is used as a heuristic exploration indicator. Combining Extra Trees with CMA-ES therefore couples covariance-based search with model-based filtering of expensive physical trials.

https://doi.org/10.31436/iiumej.v27i3.4508

This study's contribution lies in the system-level integration and experimental protocol rather than in any individual algorithmic component. CMA-ES, Extra Trees, PID control, and composite tracking metrics are established methods. Within the scope of a single line-following differential-drive AGV, this study contributes: (i) an explicit accounting scheme that separates virtual surrogate screening from costly true AGV episodes, (ii) uncertainty-informed Extra Trees ranking embedded in a CMA-ES supervisory loop so that only selected candidates consume the physical evaluation budget, (iii) integration of a 15-sensor error model, low-pass filtering, and a lost-line-aware composite objective, and (iv) a controlled fixed-budget physical comparison with pure CMA-ES, particle swarm optimization, and a genetic algorithm on Circle and 8Shape tracks. These contributions constitute a specific configuration and evaluation protocol for the tested platform. They are not presented as a new CMA-ES or Extra Trees algorithm, and broader transfer requires further validation.

Table 1. Representative PID tuning methods and their relevance to AGV line tracking

No	Tuning Method	Principle	Advantages	Disadvantages	Ref
1	Manual/empirical tuning	Adjusting K_p, K_i, K_d based on trial response.	Easy, transparent, no computation.	Subjective and often fails on multi-curvature trajectories.	[9]
2	Ziegler-Nichols / relay feedback	Using the oscillation gain and oscillation period to derive the gain.	Quick as a baseline for simple plants.	Aggressive tuning can cause oscillations and overshoot.	[9]
3	Cohen-Coon/model-based rules	Using the identified first-order-plus-delay response.	Good for known process models.	Requires a reliable model because AGV dynamics change by floor and load.	[9]
4	IMC-based PID	Lowering the gain of the internal-model-control filter.	Robust and easy to interpret.	Requires a plant model and can be suboptimal for nonlinear line tracking.	[9]
5	Fuzzy self-tuning PID	Fuzzy rules adapt the gain from error and the change in error.	Dealing with uncertainty and expert knowledge.	Rule design becomes complex and difficult to generalize.	[10]
6	Neural-network PID	Studying nonlinear mapping from state/error to gain.	Adaptive and expressive.	Requires training data and safety validation.	[11]
7	GA-PID	Evolutionary selection, crossover, and mutation optimize gain.	Global search and simple constraints.	Many evaluations and risk of premature convergence.	[12]
8	PSO-PID	Particles move based on personal best and global best.	Initial convergence is fast, and parameters are relatively few.	Can be stagnant, sensitive to inertia and swarm diversity.	[3, 13]
9	BAS-PID	Beetle antennae search using objective left-right probing.	Light and simple.	Sensitive to step size and noisy objectives.	[5]
10	DE-PID	The vector difference mutates the gain candidate.	Robust continuous optimization baseline.	Can consume many true trials in embedded experiments.	[9]
11	GWO-PID	The grey-wolf hierarchy directs exploration and exploitation.	Competitive metaheuristics for PID benchmarks.	Noisy AGV objectives can mislead the leadership hierarchy.	[9, 14]

No	Tuning Method	Principle	Advantages	Disadvantages	Ref
12	WOA-PID	The search is based on encircling and spiraling like a whale.	A simple population algorithm.	Limited adaptation of local covariance among interdependent PID gains	[15]
13	ABC-PID	Employed, onlooker, and scout bee colony exploration.	Good exploration.	It's inefficient when every trial is expensive.	[9]
14	Simulated annealing PID	Probabilistically accept worse solutions during the cooling process.	Can get out of local minima.	Sequential and sensitive to cooling schedule.	[9]
15	Bayesian optimization PID	Probabilistic surrogate and acquisition function select trial.	Very efficient sampling.	GP cost and kernel selection are difficult on noisy data.	[16]
16	Gaussian-process CMA-ES	CMA-ES uses a GP surrogate to reduce evaluations.	Uncertainty awareness and efficient sampling.	GP scalability and hyperparameter fitting can be expensive.	[17]
17	RBF surrogate CMA-ES	RBF approximates the objective in the CMA-ES loop.	Fast interpolation for smooth functions.	Less robust to discontinuities or line-loss penalties.	[18]
18	Reinforcement-learning PID	Agents learn the gain adjustment policy of rewards.	Potential for online adaptation.	Requires safe training and lots of interaction.	[19, 20]
19	Pure CMA-ES PID	Studying the covariance distribution directly from real objectives.	Powerful continuous black-box optimizer.	Each candidate consumes a real evaluation budget.	[6]
20	Proposed CMA-ES Tree	Extra Trees ranks CMA-ES candidates; only the top q = 2 candidates undergo true evaluation.	Budget-friendly, nonlinear, embedded-friendly, and uncertainty-aware.	Requires careful initial dataset and surrogate updates.	This paper

Table 1 summarizes the evolution of PID tuning methods from classical approaches to modern surrogate-assisted optimization approaches for AGV applications. Early methods such as manual tuning, Ziegler-Nichols, Cohen-Coon, and IMC have the main advantages of simplicity, transparency, and ease of implementation. However, these methods generally assume a relatively simple, stable, or well-modeled plant. In line-following AGVs, these assumptions often do not hold because vehicle response is affected by trajectory changes, floor friction, load, sensor noise, motor saturation, and line-loss conditions. Therefore, while classical methods are useful as an initial baseline, they cannot perform optimally when AGVs encounter multi-curvature trajectories such as Circle and 8Shape. Adaptive methods such as fuzzy self-tuning PID and neural-network PID can handle nonlinearity and uncertainty but require more complex rule design, training data, and safety validation before being applied to embedded systems. Several methods, such as GA, PSO, BAS, DE, GWO, WOA, ABC, simulated annealing, reinforcement learning, and CMA-ES, can perform a global search for combinations of K_p, K_i, and K_d without requiring analytical gradients. This suits AGV PID tuning because the objective function is black-box and obtained from tracking evaluation results. However, the main drawback of most of these methods is the need for many evaluations, whereas each PID trial on an AGV can be time-consuming, requires a trajectory reset, and risks producing unstable behavior. This is where surrogate-assisted methods become

important. Bayesian optimization, GP-CMA-ES, RBF surrogate CMA-ES, and CMA-ES Tree aim to reduce real-world evaluations by predicting candidate quality before testing. The proposed method, CMA-ES Tree, is more budget-efficient because Extra Trees filter CMA-ES candidates, so only the most promising candidates are sent for true evaluation.

2. METHODOLOGY

The methodology proposed in this paper treats PID tuning as a bounded black-box optimization problem. The objective is not assumed to be differentiable because it results from interactions among AGV dynamics, sensor readings, actuator saturation, and lost-line events. In this paper, the term true evaluation refers to a single high-fidelity episode execution using a prototype AGV. Surrogate prediction, internal candidate ranking, and CMA-ES population generation are not counted as true evaluations. This definition is used consistently to ensure that comparison with the baseline measures sampling efficiency, not the amount of unseen internal computation.

Fig. 2 summarizes the proposed Extra Trees surrogate-assisted CMA-ES supervisory architecture. On the optimizer side, CMA-ES generates λ=32 virtual PID candidates in normalized space. Extra Trees predicts the objective and inter-tree dispersion for each candidate, and the score in Eq. (18) ranks the population. Only the selected subset of q=2 candidates is executed on the physical AGV. On the control side, the PID controller receives tracking errors, generates the correction signal, and drives the differential motors while sensor feedback closes the loop. This architecture separates the fast, deterministic control loop from the slower supervisory optimization loop, which updates between episodes.

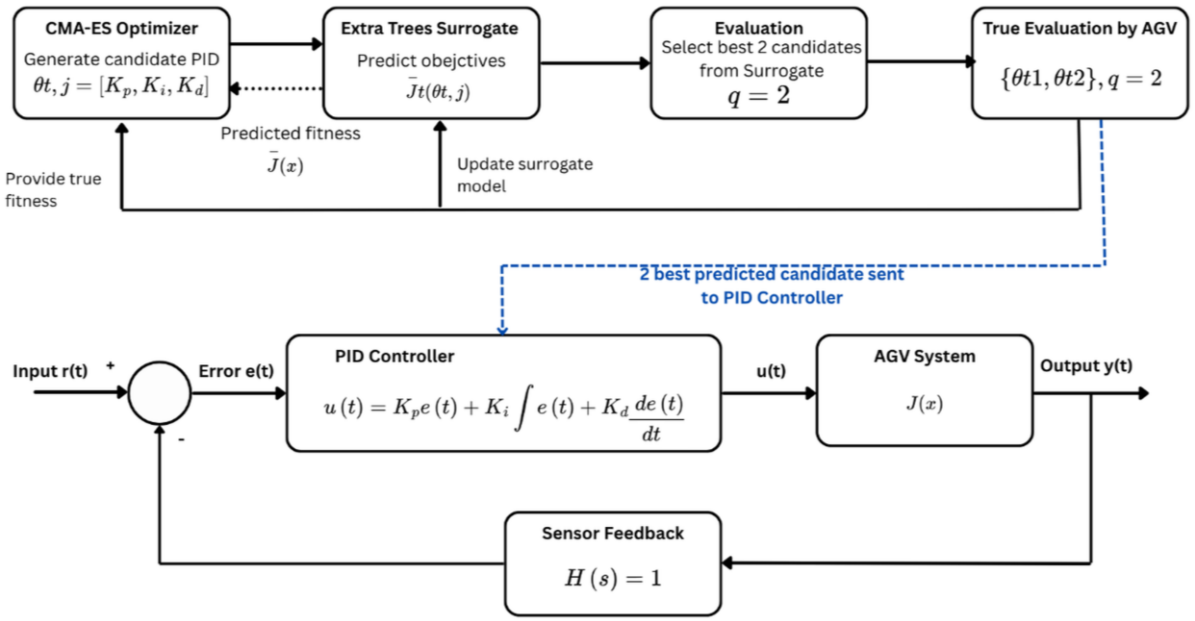


Figure 2. Proposed Extra Trees surrogate-assisted CMA-ES supervisory architecture and its relationship with the AGV PID feedback loop.

2.1. Problem Formulation

Eq. (1) defines the decision vector θ, where K_p, K_i, and K_d denote the proportional, integral, and derivative gains, respectively, and L denotes the feasible search domain. The gains are bounded by $0 \leq K_p \leq 30$, $0 \leq K_i \leq 3$, and $0 \leq K_d \leq 10$.

https://doi.org/10.31436/iiumej.v27i3.4508

$$\theta = \left[K_p\ K_i\ K_d\right]^T,\ \ L = [0{,}30] \times [0{,}3] \times [0{,}10] \tag{1}$$

These bounds are platform-specific safety and search limits established during preliminary bench testing with the same base speed, actuator limits, sensor arrangement, control direction, and track surface used in the experiments. The lower bounds are zero because negative gains reverse the intended corrective action. Above the selected upper limits, larger proportional gains produced sustained actuator saturation and lateral oscillation, larger integral gains increased windup and recovery time, and larger derivative gains amplified sensor noise and command chatter. The same domain was applied to every optimizer for fairness where these values are not proposed as universal AGV limits.

The optimization seeks a feasible PID gain vector that minimizes, as defined in Eq. (2) where θ^* is optimal PID gain. Because the J value is obtained from AGV experiments, the optimization process is considered black box. This means that the optimizer can only try a PID combination and see the resulting J value but does not know the derivative or gradient formula directly.

$$\theta^* = \arg min_{\theta \epsilon L}\, J(\theta) \tag{2}$$

CMA-ES is more stable when all variables operate on comparable numerical scales. Therefore, the physical PID gain is transformed into a normalized vector z as shown in Eq. (3). Each z component is on a unit interval, so that K_p, K_i, K_d contribute equally to the covariance adaptation.

$$z = \frac{\theta - \theta_{min}}{\theta_{max} - \theta_{min}}\ ,\ z\ \epsilon\ [0{,}1]^3 \tag{3}$$

Before the candidate is executed on the AGV, the normalized vector is converted back to the original PID domain as formulated in Eq. (4). This equation ensures that each selected candidate corresponds to a physically interpretable PID gain.

$$\theta = \theta_{min} + z \odot (\theta_{max} - \theta_{min}) \tag{4}$$

2.2. AGV Error Model

The AGV uses 15-line sensors. The relative positions of the sensors are modeled linearly from the left to the right side of the AGV body. With n_s=15 sensors, the first sensor has a position of -1, the middle sensor is around 0, and the last sensor has a position of +1. This normalized coordinate system is shown in Eq. (5). It can convert raw sensor readings into lateral errors that are independent of the physical distance between the sensors.

$$p_i = -1 + \frac{2(i-1)}{n_s - 1} \tag{5}$$

Eq. (6) converts the raw reading $s_i(k)$ into a nonnegative activation weight $\omega_i\ (k)$ by subtracting the detection threshold τ = 400. The value τ = 400 was obtained from preliminary static calibration of the 15-channel infrared array under the sensor height, track surface, and ambient lighting used in the experiments. It provided a common separation between line and background responses while retaining the response magnitude for raw weighting. The threshold was fixed for all methods and runs and is therefore a hardware and condition-specific setting, not a universal value.

$$\omega_i\ (k) = \max\ (0, s_i(k) - \tau) \tag{6}$$

Eq. (7) shows the logic of the sensor detection threshold. The robot is considered to have detected a line when at least one sensor has a positive activation. This logic is important because the objective function must penalize line-missing events. Without this component, the

optimizer might choose an aggressive gain that reduces short-term error but often causes the robot to deviate from the path.

$$\sum_{i=1}^{n_s} \omega_i(k) > 0 \tag{7}$$

Eq. (8) shows the measured lateral error calculated as the weighted centroid of the active sensors $e_m(k)$. If the left-hand sensor detects a stronger activation, the centroid shifts to the left. If the right-hand sensor is dominant, the centroid shifts to the right. This formulation yields a continuous error estimate from discrete sensor readings.

$$e_m(k) = \frac{\sum_{i=1}^{n_s} \omega_i(k) p_i}{\sum_{i=1}^{n_s} \omega_i(k)} \tag{8}$$

Sensor readings can fluctuate due to illumination, band reflectance, small vibrations, and sampling jitter. Therefore, a first-order low-pass filter as shown in Eq. (9) is used to smooth the measured error before it enters the PID controller. The parameter α_e determines the trade-off between responsiveness and noise suppression. $e(k)$ is filtered error.

$$e(k) = \alpha_e e_m(k) + (1 - \alpha_e) e(k-1) \tag{9}$$

2.3. Objective Function

The objective function is constructed from several complementary performance components. Eq. (10) represents the integral absolute error IAE, which measures the accumulated deviation from the center line. This component directly represents the tracking quality over the entire evaluation interval.

$$IAE = \sum_{k=1}^{N} |e(k)| \, \Delta t \tag{10}$$

Eq. (11) defines the RMSE, which gives greater weight to large and persistent errors. This metric is useful because two controllers can have similar IAE but different error variability.

$$RMSE = \sqrt{\frac{1}{N} \sum_{k=1}^{N} e^2(k)} \tag{11}$$

Eq. (12) shows the overshoot OS value calculated as the maximum absolute tracking error. This metric captures extreme deviations and is particularly relevant when the AGV is navigating curves or track transitions.

$$OS = max_{1 \le k \le N} |e(k)| \tag{12}$$

Eq. (13) shows the actuation smoothness SM calculated from the change in the PID correction signal $u(k)$. A low value means the controller avoids unnecessary command oscillations, thereby reducing motor stress and improving energy behavior.

$$SM = \sum_{k=2}^{N} [u(k) - u(k-1)]^2 \tag{13}$$

Eq. (14) defines the line-loss penalty PEN, which increases the objective value when the robot no longer detects the path. This component converts qualitative failures into quantitative optimization signals.

$$PEN = 100 N_{lost} \tag{14}$$

Eq. (15) defines the scalar objective used by every optimizer. The fixed weights in Table 2 place the greatest emphasis on accumulated tracking error while retaining smaller contributions from $RMSE$, overshoot, actuation smoothness, and line-loss events. Applying the same weights to every method ensures the comparison reflects how each optimizer allocates the evaluation budget, rather than differences in the performance criterion.

https://doi.org/10.31436/iiumej.v27i3.4508

$$J(\theta) = 1.0 \times IAE + 0.10 \times RMSE + 0.20 \times OS + 0.0005 \times SM + 0.01 \times PEN \quad (15)$$

2.4. Surrogate Extra Trees Model

After *t* true evaluations, the algorithm stores a dataset $\mathcal{D}_t$ as shown in Eq. (16) that contains the normalized PID vector and the observed objective values. This dataset is small but informative because it records regions where the real AGV performed well or poorly.

$$\mathcal{D}_t = \{(z_i, J_i)\}_{i=1}^{t} \quad (16)$$

Eq. (17) shows an Extra Trees Regressor F consisting of M random regression trees. Each tree f_m learns a different partition of the PID search space. Randomness is beneficial in this application because the dataset is small, nonlinear, and potentially noisy.

$$F = \{f_1, f_2 \dots f_M\} \quad (17)$$

Eq. (18) explains the candidate score function in surrogate model-based optimization. $S(z)$ is the candidate's score. The smaller the value of $S(z)$, the more attractive the candidate is to be selected. $\hat{\mu}(z)$ is the average predicted value of the surrogate model for candidate z. $\hat{\sigma}(z)$ is the level of uncertainty of the model's prediction on candidate z. If $\hat{\sigma}(z)$ is large, the model is not very confident in its prediction in that area. κ is an exploration control parameter. A large κ value makes the optimizer more daring to try uncertain areas. A small κ value makes the optimizer more focused on candidates that have been predicted well.

$$S(z) = \hat{\mu}(z) - \kappa\hat{\sigma}(z) \quad (18)$$

Eq. (19) shows the average prediction $\hat{\mu}(z)$ obtained by averaging the predictions across all trees in the ensemble. This value estimates the actual expected objective of the candidate without running an expensive AGV experiment.

$$\hat{\mu}(z) = \frac{1}{M}\sum_{m=1}^{M} f_m(z) \quad (19)$$

Eq. (20) defines $\hat{\sigma}(z)$, which represents the level of uncertainty in the model prediction for candidate z.

$$\hat{\sigma}(z) = \sqrt{\frac{1}{M-1}\sum_{m=1}^{M}[f_m(z) - \hat{\mu}(z)\,]^2} \quad (20)$$

2.5. CMA-ES Candidate Generation

Eq. (21) shows the CMA-ES mechanism in sampling candidate vectors from a multivariate Gaussian distribution in normalized space. The mean m_g represents the current search center, σ_g represents the global step size, and C_g represents the covariance matrix that captures the correlation between PID gains.

$$z_{g,j} \sim N\,(m_g, \sigma_g^2 C_g) \quad (21)$$

Eq. (22) shows a framework for preventing generated candidates from leaving the feasible interval. It does this by clipping each component to [0,1]. This repair rule is simple, deterministic, and consistent across all comparison algorithms.

$$z_{g,j} = \begin{cases} 0, & if\ z_{g,j} < 0 \\ z_{g,j}, & if\ 0 \le z_{g,j} \le 1 \\ 1, & if\ z_{g,j} > 1 \end{cases} \quad (22)$$

Eq. (23) defines Q_g as the subset of virtual candidates with the smallest surrogate scores. In each generation, $q = 2$ of the $\lambda = 32$ virtual candidates (6.25%) are sent to true evaluation. After the nine initial evaluations, this choice preserves frequent surrogate retraining within the 57-evaluation budget while providing two observed candidates for each distribution update. A larger q would reduce the screening benefit and the number of update cycles, whereas $q = 1$ would make an update more sensitive to a single noisy episode. Thus, $q = 2$ is an experiment-specific budget-noise trade-off rather than a universally optimal value.

$$Q_g = Top_q^{min}\left(\{z_{g,j}\}_{j=1}^{\lambda}; S\right), \quad q = 2 \tag{23}$$

2.6. CMA-ES Distribution Update

After true evaluation is performed, candidates are ranked based on the observed objective J. If the implementation uses the standard CMA-ES library, the mean, step-size, and covariance updates follow the CMA-ES procedure. Eqs. (24)-(26) are presented as a conceptual summary of the distribution update to explain how elite candidates shift the search center towards more promising PID regions.

$$m_{g+1} = \sum_{i=1}^{\mu} \omega_i z_{g,i}^e \tag{24}$$

The covariance estimate measures the distribution of elite candidates around the new mean. A small regularization term εI as shown in Eq. (25), is added to maintain numerical stability and avoid singular covariances.

$$C_{new} = \sum_{i=1}^{\mu} \omega_i \frac{\left(z_{g,i}^e - m_{g+1}\right)\left(z_{g,i}^e - m_{g+1}\right)^T}{\sigma_g^2} + \varepsilon I \tag{25}$$

Eq. (26) shows how the covariance matrix is updated by smoothing the previous covariance and the new estimate. This step avoids abrupt distribution changes and preserves useful search-direction information.

$$C_{g+1} = (1 - c_{cov})C_g + c_{cov}C_{new} \tag{26}$$

2.7. Algorithm Design

The CMA-ES Tree algorithm is designed for embedded deployment as a supervisory tuning module on the Raspberry Pi. This AGV prototype uses two controllers: an Arduino Uno microcontroller as the PID controller and a Raspberry Pi to coordinate the experiment, read performance logs, train the Extra Trees surrogate, generate gain candidates, and write the selected PID gain to the controller. This separation is crucial for real-time safety. The inner PID loop must remain deterministic, while the optimizer can operate at a slower timescale after each evaluation episode.

In a practical implementation, each evaluation episode begins by loading a vector of candidate PIDs. The AGV follows a defined trajectory for the duration of the evaluation, while the Raspberry Pi records timestamps, raw and filtered errors, line detection status, correction commands, and wheel commands. At the end of the episode, the objective component is calculated and stored. Extra Trees retrains on the updated dataset, and CMA-ES proposes the next candidate population. The surrogate filters the candidates so that only a small number of promising candidates are physically evaluated. This architecture reduces risky or unproductive real-world trials and suits low-cost AGV platforms.

Algorithm 1 shows how Fixed-Budget CMA-ES Tree PID Tuning optimizes PID parameters (K_p, K_i, K_d) on an AGV with a limited evaluation budget $B = 57$. The process

begins with normalization of the search space, initial evaluation of $N_0 = 9$ deterministic candidates, and then the best candidate is used to initialize CMA-ES. At each generation, CMA-ES generates virtual candidates; the Extra Trees surrogate then predicts their performance and uncertainty, so only the most promising candidates are evaluated directly on the AGV. The evaluation results are added to the history H, while CMA-ES adapts its mean, covariance, and sampling scale to improve the search. A restart mechanism is used when stagnation occurs, while micro-polishing will utilize the remaining budget to refine the best solution. With this strategy, the algorithm can obtain the best-observed PID parameters and the minimum objective value efficiently in a limited number of real evaluations.

Algorithm 1. Embedded supervisory tuning process for CMA-ES Tree

Input	Random seed s, PID bounds $K_p \in [0,30], K_i \in [0,3], K_d \in [0,10]$, evaluation budget $B = 57$, deterministic initial evaluations $N_0 = 9$, CMA-ES virtual population size $\lambda = 32$, number of true evaluations per generation $q = 2$, elite size μ=6, minimum surrogate training size $N_{\min} = 8$, surrogate exploration factor $\kappa = 0.35$, micro-polish threshold $B_m = 12$, episode duration $T = 20s$, and objective weights $\omega_{IAE}, \omega_{RMS}, \omega_{OS}, \omega_{SM}, \omega_{PEN}$.
Step 1	Define the PID search space using the global bounds of K_p, K_i, and K_d
Step 2	Normalize the PID search space into $z \in [0,1]^3$. Initialize the evaluation history $H \leftarrow \emptyset$, evaluation counter $n \leftarrow 0$, and generation counter g ← 0
Step 3	Generate $N_0 = 9$ deterministic initial PID candidates.
Step 4	For each initial candidate K_j, while $n < B$, run one AGV episode, calculate J_j, append (K_j, J_j) to H, and update $n \leftarrow n + 1$.
Step 5	Select $(K^*, J^*) = \arg min_{(K,J)\in H} J$. Initialize the CMA-ES mean using the normalized K^*, with $\sigma = 0.20$ and $C = I$
Step 6	While $n < B, set\ g \leftarrow g + 1$, calculate the remaining budget $r \leftarrow B - n$, and set the number of true evaluations to $q_g \leftarrow \min(2, r)$.
Step 7	If the number of evaluated candidates in H is at least $N_{min} = 8$, fit the Extra Trees surrogate using all evaluated pairs in H . Otherwise, skip surrogate fitting and use the trust-region-based candidate-selection fallback.
Step 8	If $r \leq B_{micro} = 12$, activate micro-polish: center the CMA-ES mean on the current K^*, set $\lambda_g = 2\lambda = 64$, and use $\sigma_{sample} = 0.40\sigma$. Otherwise, set $\lambda_g = 32$ and $\sigma_{sample} = \sigma$.
Step 9	Generate λ_g virtual candidates from the CMA-ES distribution in the normalized search space. Virtual candidates do not increase n.
Step 10	Remove candidates that duplicate or nearly duplicate previously evaluated candidates in H
Step 11	Use the Extra Trees ensemble to predict the mean objective $\hat{\mu}(z)$ and inter-tree standard deviation $\hat{\sigma}(z)$ of each remaining virtual candidate.
Step 12	Rank the virtual candidates using $S(z) = \hat{\mu}(z) - \kappa\hat{\sigma}(z) + 10[\max(0, d_{TR}(z)/R - 1]^2$.
Step 13	Select the q_g candidates with the lowest scores. For each selected candidate, if $n < B$, run one AGV episode, calculate its objective J, append $(K, J)\ to\ H$, and update $n \leftarrow n + 1$.
Step 14	Update (K^*, J^*) using the lowest observed objective value in H.
Step 15	Select the $\mu = 6$ best unique elites from H, and update the CMA-ES mean and covariance using the selected elites.
Step 16	Update the CMA-ES sampling scale and trust-region radius according to whether J^* improved.
Step 17	If the failure streak reaches the predefined stagnation threshold, restart the CMA-ES around K^*, reset $C \leftarrow I$, and set $\sigma \leftarrow 0.35$
Step 18	End while. If $n < 57$, return to Step 6. When n=57, terminate the optimization without evaluating additional candidates.
Output	Best observed PID vector $K^* = (K_p^*, K_i^*, K_d^*)$ and its objective value J^*.

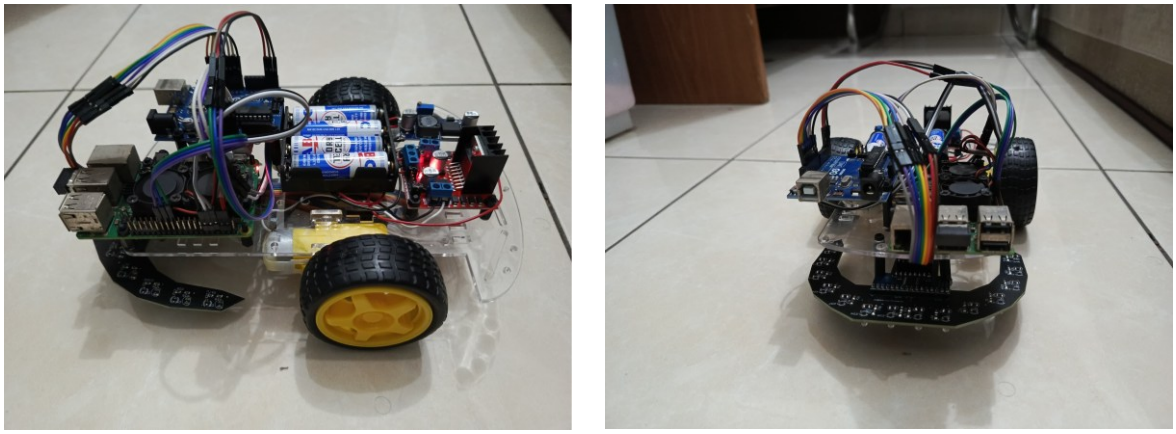
2.8. Experimental Design, Fairness, and Validation

Fig. 3 shows an experimental AGV prototype used as a hardware validation platform to test the control system and optimize PID parameters. Fig. 3(a) shows a side view of the prototype, where several key components are integrated on a single acrylic frame, including a

https://doi.org/10.31436/iiumej.v27i3.4508

Raspberry Pi, Arduino Uno, motor driver module, battery power supply, DC motor, drive wheels, and a sensor array on the front. This arrangement indicates that the prototype is designed as a compact mobile mechatronics system, with electronic components and actuators placed centrally to maintain mechanical balance during movement. The drive wheels and motor control module allow the AGV to perform differential maneuvers, while the front sensor module serves as the main input for detecting trajectory or environmental changes during navigation.

Fig. 3(b) shows the front view of the prototype, emphasizing the sensor position at the front of the AGV. Front-mounted sensor placement is important because it allows the system to obtain trajectory information early, before the vehicle body passes through the path, enabling real-time control corrections. This hardware architecture supports a closed-loop experimental scenario in which the control system reads sensor data, processes it into correction signals based on PID parameters, and then sends them to the left and right motors to adjust the AGV's direction. With this configuration, this prototype provides an objective evaluation platform to compare the performance of PID tuning methods, particularly based on indicators such as trajectory tracking accuracy, motion stability, overshoot, actuation smoothness, and trajectory maintenance ability under real-world experimental conditions.



(a) (b)

Figure 3. (a) Side view of AGV prototype, (b) Front view of AGV prototype

Fig. 4 shows the experimental track layout used to evaluate the AGV's path-tracking capabilities at two levels of geometric complexity. Fig. 4(a) shows an 8Shape track, which is more challenging because it combines straight segments, sharp turns, repeated direction changes, and a central intersection. This configuration tests the control system's dynamic response to rapid curvature changes, particularly when the AGV must make consecutive course corrections in a short period. Thus, the 8Shape track provides a more complex test scenario for assessing control stability, line-following ability, overshoot tendencies on turns, and the system's resilience to tracking errors as the vehicle moves from one track segment to another.

Fig. 4(b) shows the Circle trajectory, a closed pattern dominated by continuous turns and smoother directional transitions than the 8Shape trajectory. This trajectory was used to evaluate the AGV's ability to maintain stable motion on a circular path, particularly in balancing speed, steering correction, and consistent line sensor readings. Compared to the 8Shape trajectory, the Circle trajectory provides a more regular control load, making it suitable for observing the PID system's basic performance under relatively consistent curvature conditions. Using these two trajectories allows for a more comprehensive evaluation because the PID tuning method is tested not only under simple, stable conditions but also under track conditions that require more

https://doi.org/10.31436/iiumej.v27i3.4508

aggressive control adaptation. Therefore, the experimental setup in Fig. 4 plays a crucial role in assessing the AGV's objective performance based on metrics such as tracking error, actuation smoothness, overshoot, lost-line count, and aggregate objective value.

(a) (b)

Figure 4. (a) 8Shape track, (b) Circle track

Table 2 lists the parameter settings used for the comparison. We repeated each method-track combination with 20 random seeds, and limited each run to 57 true evaluations. We held the PID domain, objective function, line threshold, evaluation duration, base speed, anti-windup policy, error filter, and initial conditions constant. We did not count surrogate predictions as true evaluations because they were not executed on the AGV. The proposed method could generate and score many candidates internally, but only physically executed candidates consumed the budget. Pure CMA-ES, PSO, and GA were subject to the same 57-call limit, so the experiment compares sample efficiency under an identical physical budget.

Table 2. Reproducibility and fairness parameters for all methods

Group	Parameter	Value	Fairness Objective
PID Domain	PID Search Limit: Kp_Bounds, Ki_Bounds, Kd_Bounds	Kp ∈ [0, 30] Ki ∈ [0, 3] Kd ∈ [0, 10]	The platform-specific bounds were set by preliminary safety tests and applied identically to all methods where they are not universal AGV limits.
Line sensor	Num_Sensors, Line_Threshold, Use_Raw_Weight, sensor_positions	15 IR sensors, threshold 400, raw weighting True, sensor position [-1, 1]	The threshold was calibrated once for the tested sensor height, surface, and lighting and then fixed for all methods and runs.

https://doi.org/10.31436/iiumej.v27i3.4508

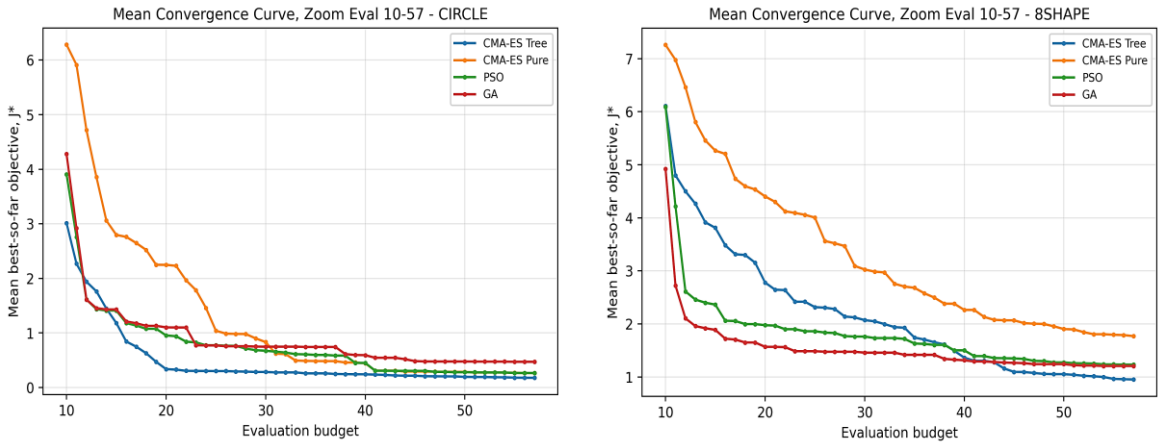
Group	Parameter	Value	Fairness Objective
Control and Actuator	Sampling interval/control frequency, base speed, actuator limits, integral clamp, derivative filter, initial/reset condition, and lost-line recovery.	Dt=0.032s, fc=31.25Hz, vbase=3 rad/s, clipped to ±6.28 rad/s, $\alpha_e = 0.30$, clip I(k)= -2 to 2, $\alpha_D = 0.20$, $e_f = 0, e_{last} = 0, e_m(k) = 0$	All control and actuator parameters are applied identically to all methods.
Budget	Eval_Budget, episode duration, objective weight	57 real evaluations, Evaluation_Duration_Ms = 20000, W_Iae=1.0, W_Rms=0.10, W_Os=0.20, W_Sm=0.0005, W_Pen=0.01	All optimizers receive the same number of evaluations and use the same objective function, so the method's advantage comes from search efficiency, not a larger budget.
CMA-ES Tree	Sa_Init_Evals, Sa_Cma_Lambda, Sa_True_Evals_Per_Gen, Sa_Cma_Mu, Sa_Sigma_Init, Sa_Sigma_Min, Sa_Sigma_Max, Sa_Cov_Learning_Rate, Sa_Stagnation_Restart, Sa_Micro_Polish_Last number of trees M, max_features, min_samples_leaf, maximum depth, Random state	Initial true eval 9, virtual candidates 32, true eval/generation 2, elite 6, initial sigma 0.20, sigma min–max 0.015–0.55, covariance learning rate 0.28, stagnation restart default 8, but sweep seeds 2–10, micro-polish last 12 eval, M=1000, max_features=1.0, min_samples_leaf=1, max-depth=none, rand_state=s+1000+g	CMA-ES Tree uses virtual candidates for exploration, but only candidates actually tested in the field are counted toward the budget. This maintains fairness compared to other methods, which are also limited to 57 true evaluations.
Pure CMA-ES	Cma_Lambda, Cma_Mu, Cma_Sigma_Init, Cma_Sigma_Min, Cma_Sigma_Max, Cma_Cov_Learning_Rate, Cma_Stagnation_Restart, Cma_Micro_Polish_Last	Population λ=6, elite μ=3, initial sigma 0.20, sigma min–max 0.015–0.55, covariance learning rate 0.28, stagnation restart default 8, sweep seeds 2–10, micro-polish last 12 eval	All Pure CMA-ES candidates are fully evaluated directly in AGV, with no Extra Trees surrogates. Therefore, each candidate consumes a real budget directly.
PSO	Swarm_Size, Pso_Iters, W_Max, W_Min, C1, C2, Vmax_Frac	Initial Default Swarm_Size=12, Pso_Iters=8, But Automatically Selected to Have Swarm_Size × Pso_Iters = 57, Effective= 19 × 3 = 57, Inertia 0.90 → 0.40, C1=1.80, C2=1.80, Vmax_Frac=0.20	PSO is automatically adjusted so that the number of particle evaluations exactly matches the budget of 57, so it does not get more evaluations than other methods.
GA	Fair_Auto_Params, Pop_Size, Generations, Elite_N, Tournament_K, Crossover_Rate, Mutation_Rate_Gene, Mut_Sigma_Frac	Fair_Auto_Params=True, Initial Default Is Pop_Size=14, Generations=7, But Is Automatically Selected to Have Pop_Size × Generations = 57, Effective: 19 × 3 = 57, Elite 2, Tournament 3, Crossover 0.90, Mutation Gene 0.25, Mutation Sigma Fraction 0.12	The GA is made fair by adjusting the population size and number of generations so that total individual evaluations equal the budget of 57, rather than following the default configuration, which can exceed the budget.

3. RESULTS

This section interprets the convergence, distribution, and time-response results from CMA-ES Tree and the three baselines. It focuses on the final objective J, convergence efficiency, cross-seed distribution, and time-domain trade-offs. Fig. 5(a) shows the

https://doi.org/10.31436/iiumej.v27i3.4508

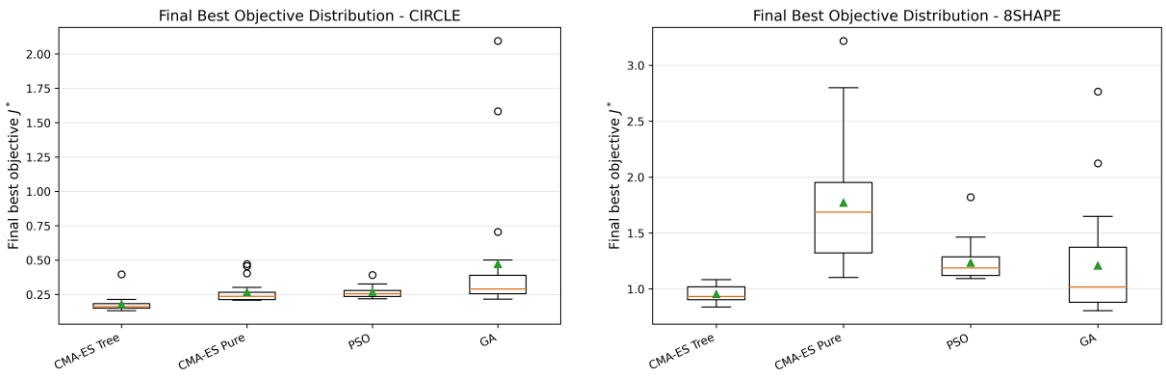
convergence curves of the mean best-so-far objective J on the Circle path for four methods: CMA-ES Tree, CMA-ES Pure, PSO, and GA. In general, all methods decrease J as the evaluation budget increases. This indicates that the optimization process has successfully found a better combination of PID parameters. However, CMA-ES Tree shows the lowest observed mean best-so-far objective across all methods at the evaluated budgets. From the start of the evaluation, this method reduces the objective value sharply and achieves the lowest final value compared to the other methods. CMA-ES Pure requires a larger budget to decrease due to its very high initial value, while PSO and GA decrease more steadily but stop at a higher final value than CMA-ES Tree. This shows that on the Circle trajectory, the Tree-based surrogate-assisted approach uses the evaluation budget more efficiently to find the best PID parameters.



(a) (b)

Figure 5. (a) Average convergence in circle track (b) Average convergence in 8Shape track

Fig. 5(b) shows a more challenging pattern on the 8Shape trajectory, as the initial objective value is higher and the decrease is slighter than on the Circle trajectory. This is understandable because the 8Shape trajectory has more complex directional changes and curvature, so PID tuning requires stronger adaptation. On this trajectory, CMA-ES Tree still shows the best performance at the end of the evaluation, with the lowest final J value and a curve that continues to decrease until it approaches the maximum budget. GA is also competitive because it decreases rapidly at the beginning of the evaluation but then tends to stagnate. PSO shows a gradual decrease, while CMA-ES Pure is the slowest method because its objective value remains the highest until the end of the evaluation. Thus, these two graphs confirm that CMA-ES Tree has the best convergence efficiency and solution quality on both simple and complex trajectories, especially in fixed-budget scenarios with a limited number of evaluations.



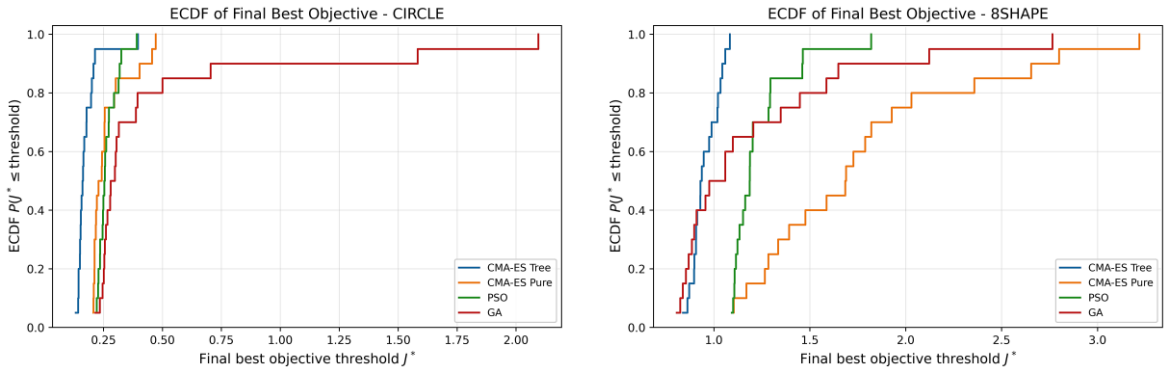
(a) (b)

Figure 6. (a) Final best objective in circle track (b) Final best objective in 8Shape track

https://doi.org/10.31436/iiumej.v27i3.4508

Fig. 6(a) shows the final distribution of the best objective J on the Circle path after the entire evaluation budget has been used. A lower J value indicates better PID tuning quality. On this path, CMA-ES Tree performs best because it has the lowest median, the narrowest data distribution, and few outliers. This indicates that CMA-ES Tree not only finds solutions with small objective values but is also more consistent across runs. CMA-ES Pure and PSO have slightly higher medians, with relatively controlled variation, while GA shows the widest distribution and several large outliers. The outliers in GA indicate that in some runs, this method failed to find the best-observed combination of PID parameters within the same evaluation budget.

Fig. 6(b) shows that the 8Shape trajectory is more challenging than the Circle trajectory, as evidenced by the generally higher final J values across all methods. On this trajectory, CMA-ES Tree again performs best, with the lowest median and the most stable distribution compared to other methods. GA has a fairly competitive median, but its spread is larger and includes high outliers, resulting in less consistent performance. PSO shows intermediate results with relatively small variations, but its objective median is still higher than CMA-ES Tree. Meanwhile, CMA-ES Pure has the highest final objective value and the largest distribution, indicating that without the assistance of a surrogate Tree, the search process requires more real-world evaluations to reach a competitive solution. Thus, this boxplot confirms that CMA-ES Tree is more consistent across the tested runs and more budget-efficient at producing the best PID parameters, on both simple and complex trajectories.



(a) (b)

Figure 7. (a) ECDF in circle track (b) ECDF in 8Shape track

Fig. 7(a) shows the ECDF of the final best objective J on the Circle path. The ECDF measures the probability that a method produces a final objective value less than or equal to a given threshold. Therefore, a curve farther to the left and rising more quickly to 1 indicates a better and more consistent method. On the Circle path, CMA-ES Tree is the best method because its curve is farthest to the left and rises very sharply. This means that most CMA-ES Tree runs have already achieved low J values at a small threshold. PSO and CMA-ES Pure are in the middle, while GA has a very long distribution tail up to large J values. The long tail for GA indicates that while some runs can produce reasonably good solutions, others fail to achieve optimal performance within the same evaluation budget.

Fig. 7(b) shows that the 8Shape path is more difficult than the Circle path because the entire curve shifts to the right, meaning the final objective values are generally higher. On this path, CMA-ES Tree remains superior because its curve is farthest to the left and reaches a cumulative probability of 1 at the lowest threshold. This means all CMA-ES Tree runs produced relatively good and stable final solutions. PSO came in second because its curve still rose quite quickly but required a larger threshold J than CMA-ES Tree. GA showed quite

https://doi.org/10.31436/iiumej.v27i3.4508

competitive initial performance in some runs, but had a long right tail, indicating performance variation and the risk of bad runs. Meanwhile, CMA-ES Pure was the weakest method on 8Shape because its curve was farthest to the right and required the highest objective threshold to cover all runs. Overall, this ECDF graph confirms that CMA-ES Tree has the best reliability in fixed-budget scenarios, especially because it produced consistently low objective values on both simple and complex paths.

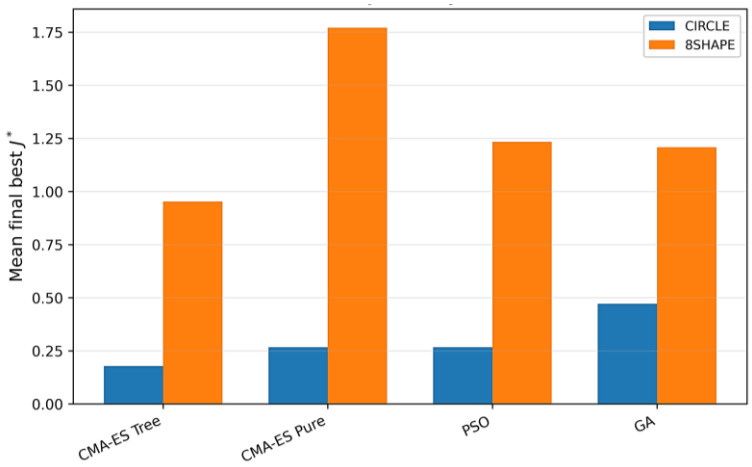


Figure 8. Average final best objective value by method and trajectory.

Fig. 8 compares the mean final best objective J by optimization method and trajectory type. Because J represents the objective function to be minimized, a smaller J value indicates better-quality PID parameters. On the Circle trajectory, all methods produced relatively low final objective values compared to the 8Shape trajectory. This indicates that the Circle trajectory is easier to optimize due to its simpler motion pattern and more stable curvature changes. Among all methods on the Circle trajectory, CMA-ES Tree obtained the lowest average J value, followed by CMA-ES Pure and PSO, which were nearly equal. Meanwhile, GA produced the highest average J value on the Circle trajectory, making its performance less efficient than the other methods under simple trajectory conditions.

On the 8Shape trajectory, all methods experienced an increase in J values. Nevertheless, CMA-ES Tree still shows the best performance with the lowest average J value compared to CMA-ES Pure, PSO, and GA. PSO and GA are at an intermediate level with relatively close values, while CMA-ES Pure produces the highest objective value on 8Shape, indicating that pure CMA-ES search without surrogate assistance requires more evaluations to reach a competitive solution. Overall, this graph confirms that CMA-ES Tree is more effective and budget-efficient at producing the best PID parameters on both simple and complex trajectories.

Fig. 9 shows the RMSE distribution of the best candidates for each run for the four PID optimization methods on two trajectory types, Circle and 8Shape. In general, the Circle trajectory exhibits lower RMSE than the 8Shape trajectory. On the Circle trajectory, all methods fall within a narrow error range, but CMA-ES Tree Circle appears to perform best due to its lowest median RMSE and very narrow distribution. PSO and CMA-ES Pure are also quite stable, while GA Circle has a slightly higher median and several large outliers, making its performance less consistent than the other methods.

On the 8Shape trajectory, the differences between the methods become more pronounced. CMA-ES Tree 8Shape performs best, with the lowest median RMSE among the methods and a relatively narrow distribution indicating more stable tuning results across runs. GA 8Shape has a fairly competitive median, but its spread is wider, and outliers reduce consistency.

https://doi.org/10.31436/iiumej.v27i3.4508

Meanwhile, PSO 8Shape and especially CMA-ES Pure 8Shape show higher RMSE, with large variations and outliers. Thus, this graph supports that the CMA-ES Tree approach is more effective and more consistent across the tested runs, especially on complex trajectories, as it produces PID candidates with smaller and more stable tracking errors.

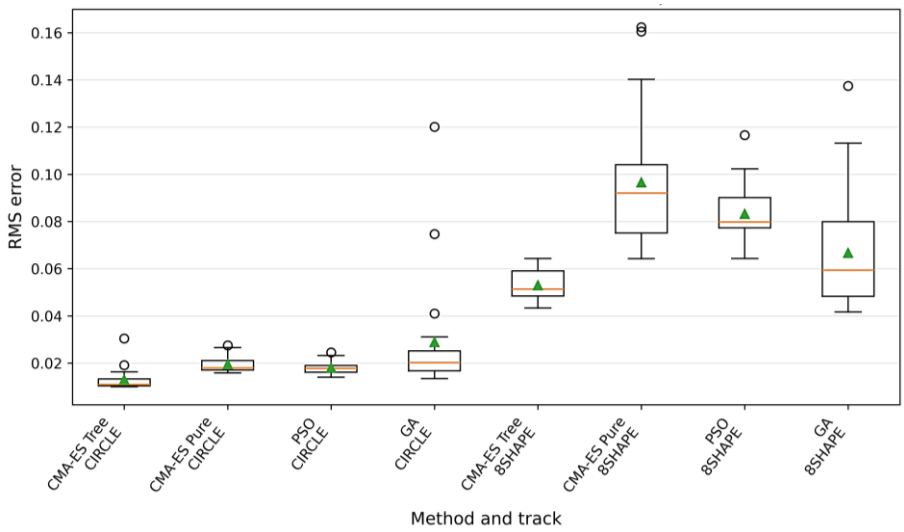


Figure 9. Distribution of RMSE of the best candidate.

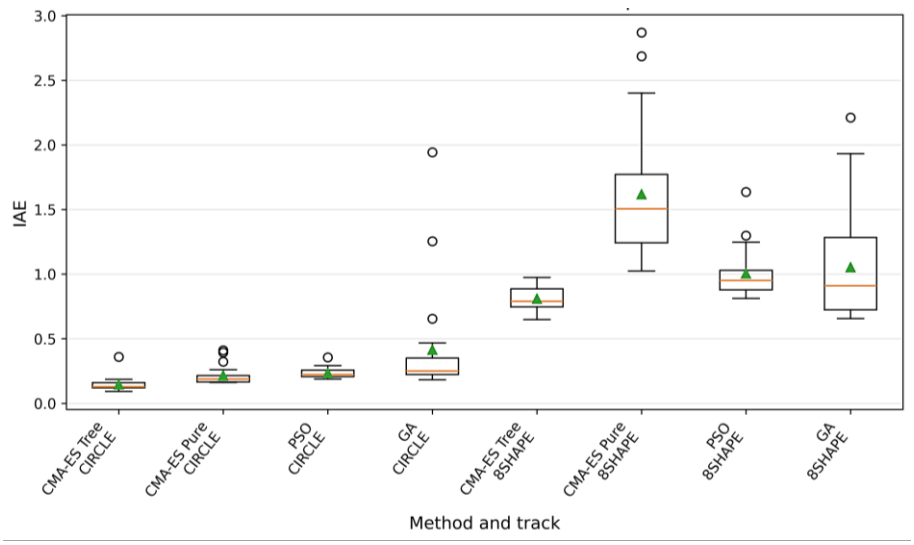


Figure 10. Distribution of the best candidate IAEs.

Fig. 10 shows the distribution of the Integral Absolute Error (*IAE*) of the best PID candidate for each run for the four optimization methods on the Circle and 8Shape trajectories. *IAE* represents the accumulated total tracking error over the course of an episode. The lower the *IAE*, the better the *AGV* maintains its position relative to the track line. On the Circle trajectories, all methods produced relatively low *IAE*. Among the methods on the Circle trajectories, CMA-ES Tree performed best, with the lowest median *IAE* and the narrowest data distribution. CMA-ES Pure and PSO remained relatively stable but had slightly higher medians. Meanwhile, the Circle GA showed greater variation and several high outliers, indicating that the GA method produced suboptimal PID parameters in some runs.

On the 8Shape trajectories, IAE values increased for all methods. CMA-ES Tree 8Shape still produces the lowest IAE and a relatively narrow distribution compared to other methods,

https://doi.org/10.31436/iiumej.v27i3.4508

so it can be said to be more consistent in minimizing error accumulation. PSO 8Shape is in the middle, with a higher median IAE than CMA-ES Tree but more stable than GA. GA 8Shape has quite wide variations and several high outliers, while CMA-ES Pure 8Shape shows the highest IAE and the largest distribution, indicating less consistent performance within a limited evaluation budget. Overall, this graph confirms that CMA-ES Tree is more effective and more consistent across the tested runs in producing PID parameters with low accumulated tracking error on both simple and complex trajectories.

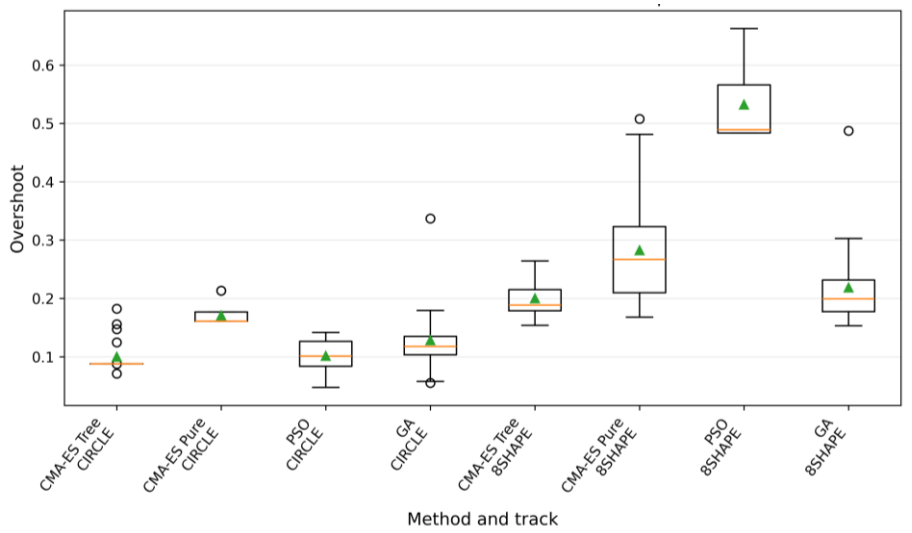


Figure 11. Distribution of overshoot of the best candidates.

Fig. 11 shows the overshoot distribution of the best PID candidates for each run for the CMA-ES Tree, CMA-ES Pure, PSO, and GA methods on the Circle and 8Shape trajectories. Overshoot represents the maximum deviation of the AGV response when it crosses or overshoots the reference position of the trajectories. The smaller the overshoot value, the better the stability of the PID control. On the Circle trajectories, the overshoot values for all methods were relatively low compared to the 8Shape trajectories. CMA-ES Tree Circle and PSO Circle performed best due to their low median overshoot. However, CMA-ES Tree had several outliers, indicating that higher overshoot occurred in a small portion of the runs. CMA-ES Pure Circle had a very narrow distribution, but its median overshoot was higher than those of CMA-ES Tree and PSO. Meanwhile, GA Circle was intermediate, with one relatively high outlier.

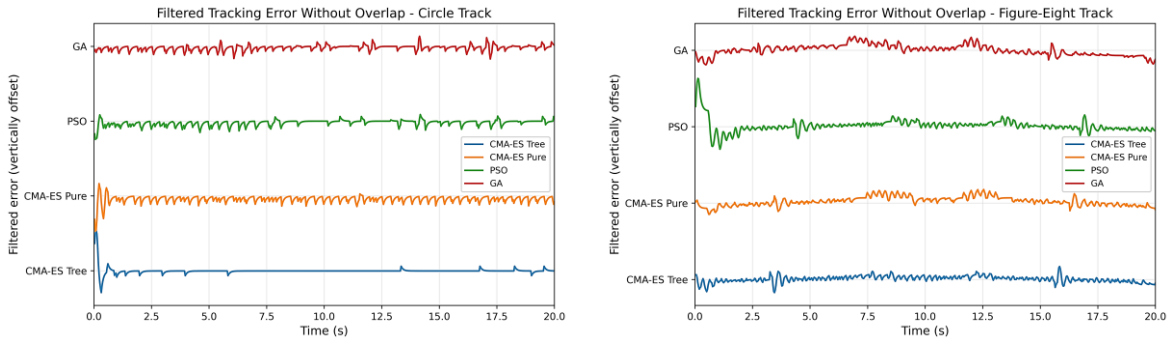
On the 8Shape trajectory, the overshoot value increases across all methods. CMA-ES Tree 8Shape produces relatively low and stable overshoot compared to CMA-ES Pure and PSO. This indicates a better ability to prevent overly aggressive AGV responses. GA 8Shape also has a relatively low median overshoot, but high outliers indicate potential instability in some runs. In contrast, PSO 8Shape has the highest overshoot, with a large median and spread, indicating that the PSO-derived PID parameters tend to produce more aggressive control responses on complex trajectories. Overall, this graph confirms that CMA-ES Tree is more consistent across the tested runs in suppressing overshoot, especially on complex trajectories, making it more suitable for AGV PID tuning in fixed-budget scenarios.

Fig. 12(a) shows the filtered tracking error on the Circle trajectory for the best candidate for each method. The main comparison is based on the oscillation amplitude, stability, and number of spikes on each curve. On the Circle trajectory, CMA-ES Tree showed the most stable response because, after the initial transient, the error rapidly decreased and the curve remained relatively flat. CMA-ES Pure still exhibited fairly consistent periodic oscillations,

https://doi.org/10.31436/iiumej.v27i3.4508

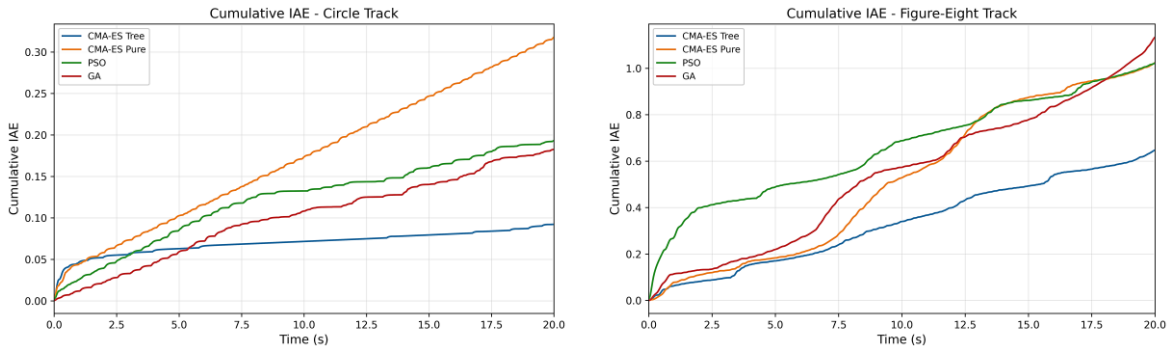
PSO had some fluctuations and spikes, while GA showed coarser error variations and was not as smooth as CMA-ES Tree. This indicates that the PID parameters from CMA-ES Tree kept the AGV closer to the trajectory line more smoothly on simple trajectories.

Fig. 12(b) shows conditions on the 8Shape trajectory. On this trajectory, all methods experienced more noticeable fluctuations in tracking error than on the circle track, particularly at the beginning and at several trajectory transition points. Nevertheless, CMA-ES Tree still produces the most controlled error pattern, with relatively small oscillation amplitudes and no prolonged large deviations. CMA-ES Pure shows larger ripples and error changes; PSO experiences a fairly strong initial transient and fluctuations at several time intervals, while GA shows higher error variations and tends to be less smooth. Overall, this graph confirms that CMA-ES Tree produces a more stable and smoother tracking response on both Circle and 8Shape paths, making it more effective for AGV PID tuning in maintaining path-tracking accuracy.



(a) (b)

Figure 12. (a) Filtered Tracking Error in Circle track (b) Filtered Tracking Error in 8Shape track



(a) (b)

Figure 13. (a) Cumulative IAE in circle track (b) Cumulative IAE in 8Shape track

Fig. 13(a) shows the cumulative IAE for the Circle trajectory, which is the accumulation of absolute error during one AGV operating cycle. A lower curve indicates a smaller total tracking error accumulated over the entire episode. On the Circle trajectory, CMA-ES Tree produced the lowest cumulative IAE until the end, around 0.09, and the curve flattened relatively gradually after the initial phase. This indicates that the CMA-ES Tree PID candidate maintained a consistently small error throughout the trajectory. GA and PSO ranked in the middle with larger final error accumulations, while CMA-ES Pure had the highest cumulative IAE, around 0.32, indicating that errors were small but repeated over time, resulting in the largest accumulation.

https://doi.org/10.31436/iiumej.v27i3.4508

Fig. 13(b) shows the cumulative IAE on the 8Shape trajectory. On this trajectory, CMA-ES Tree remains the best method because its curve is the lowest until the end, around 0.65, resulting in the smallest total tracking error. PSO experiences a very rapid increase in error at the beginning, while CMA-ES Pure and GA show a sharp increase in the middle to the end of the trajectory. GA even reaches the highest cumulative IAE at the end. Overall, this graph confirms that CMA-ES Tree is more effective at suppressing the accumulation of tracking error on both simple and complex trajectories.

4. DISCUSSION

Figs. 5-8 indicate that the principal advantage of CMA-ES Tree was how it allocated the fixed true-evaluation budget. CMA-ES retained a broad virtual search over the normalized PID domain, whereas Extra Trees screened the virtual population before physical execution. The real platform was therefore used mainly to evaluate candidates predicted to be promising or informative under the inter-tree dispersion term. Pure CMA-ES, PSO, and GA had no comparable screening stage, so every candidate immediately consumed a true evaluation. The observed convergence advantage should consequently be interpreted as improved sample allocation under an equal physical budget, not as an advantage obtained from additional AGV trials.

Trajectory geometry changed the difficulty of the search and exposed different optimizer behavior. The Circle track imposed relatively regular curvature, allowing all methods to enter a lower-objective region. The alternating curvature, direction changes, and central transition of the 8Shape track generated a more heterogeneous mapping between PID gains and tracking performance. Under that condition, the surrogate-assisted method continued to improve late in the budget, whereas the baselines more often plateaued or retained broad final distributions. This pattern supports the proposed screening mechanism for the two tested trajectories, but it does not establish that the same ranking will hold for every path geometry.

The boxplots and empirical cumulative distribution functions provide information beyond the mean final objective. Their left-shifted and comparatively concentrated patterns show that CMA-ES Tree was more likely to return a low-objective solution across repeated seeds. This cross-run reliability matters in experiment-based tuning because a single favorable run does not show that an optimizer can repeatedly find a usable controller within a restricted budget. The longer tails observed for some baselines indicate a greater risk of ending the experiment with a poor gain set, even when their best runs were competitive.

Figs. 9-13 summarize the distributions of time-domain-derived performance metrics and reveal trade-offs rather than uniform dominance across all metrics. CMA-ES Tree produced the most consistent reduction in RMSE, accumulated error, and cumulative IAE, particularly on the 8Shape track. Its overshoot, however, was not the minimum in every condition. PSO was competitive on Circle, and GA showed a competitive median on 8Shape but with less reliable tails. This distinction matters because the optimizer minimizes a composite objective that also includes actuation smoothness and a lost-line penalty. The selected controller should therefore be viewed as a budget-constrained compromise among tracking accuracy, extreme deviation, command variation, and recovery reliability rather than a solution optimized for one metric alone.

From an implementation perspective, the two-level architecture keeps the time-critical PID loop on the Arduino, while training and candidate selection occur between episodes on the Raspberry Pi. This separation preserves deterministic inner-loop execution and permits the supervisory tuner to use more virtual candidates than could be tested physically. Extra Trees is

suitable for the small nonlinear tabular dataset because it retrains quickly and does not assume a smooth objective. Nevertheless, inter-tree dispersion is a heuristic indicator rather than a calibrated error probability. Its value depends on the diversity and coverage of the samples accumulated.

The methodological contribution is therefore the integration of explicit true-evaluation accounting, uncertainty-informed surrogate ranking, and embedded physical validation under a common budget. We do not claim that CMA-ES, Extra Trees, PID control, or the individual performance metrics are new. The evidence instead shows that their specific integration can improve the probability of finding a useful PID setting when physical evaluations are selective and expensive. This positioning aligns the contribution with the scope of the experiment and avoids treating two-track validation as a universal AGV framework.

Several limitations constrain the interpretation of these findings. The experiments used one line-following differential-drive prototype, sensor arrangement, two trajectories, and fixed hardware and operating settings. The PID bounds and the threshold $\tau = 400$ were calibrated for the tested actuator limits, sensor height, surface, and lighting, so they should be recalibrated for another platform. The study did not vary steering architecture, payload, base speed, floor friction, battery state, sensor layout, or environmental illumination. In addition, surrogate quality depends on the coverage and noise level of the initial and sequential data, while inter-tree dispersion is not a formally calibrated uncertainty measure. Future work should validate the method on longer industrial routes and multiple AGV platforms, include systematic load and speed changes, and conduct ablation and sensitivity studies for the surrogate score, q, initialization, and other supervisory design choices.

5. CONCLUSION

This study evaluated an Extra Trees surrogate-assisted CMA-ES supervisory tuner for PID gain selection on a line-following differential-drive AGV under an identical budget of 57 true evaluations. By ranking a larger virtual CMA-ES population and physically executing only selected candidates, the method achieved the best end-of-budget objective and the most favorable cross-run reliability on both tested trajectories. The time-domain evidence further showed that the selected gains reduced tracking-error accumulation while maintaining a practical compromise with overshoot, command smoothness, and line-loss protection. The results support surrogate screening as a sample-efficient supervisory strategy for the tested embedded platform when physical trials are costly and limited. They are not a claim of universal performance across AGV architectures. The study therefore provides a focused experimental basis for broader multi-platform and variable-operating-condition validation.

ACKNOWLEDGEMENT

This research was funded by the Ministry of Higher Education (MoHE) Malaysia through the Fundamental Research Grant Scheme (FRGS), grant number FRGS/1/2021/TK0/UKM/02/17. We also acknowledge support from the National Research and Innovation Agency (BRIN) Indonesia and the Indonesia Endowment Fund for Education Agency (LPDP) under the Perjanjian Pendanaan Program Riset dan Inovasi untuk Indonesia Maju Gelombang 4 tahun 2023.

REFERENCES

[1] Oyekanlu EA, et al. (2020) A review of recent advances in automated guided vehicle technologies: Integration challenges and research areas for 5G-based smart manufacturing applications. IEEE Access, 8:202312-202353.

[2] Dos Reis WPN, Couto GE, Junior OM. (2023) Automated guided vehicles position control: A systematic literature review. Journal of Intelligent Manufacturing, 34(4):1483-1545.

[3] Alwan MMA, Green AA, Noori AS, Aldair AA. (2021) Design and implementation of line follower Arduino mobile robot using Matlab Simulink toolbox. Iraqi Journal for Electrical and Electronic Engineering, 17(2):11-16.

[4] Moshayedi AJ, Abbasi A, Liao L, Li S. (2019) Path planning and trajectory tracking of a mobile robot using bio-inspired optimization algorithms and PID control. 2019 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), 1-6.

[5] Moshayedi AJ, et al. (2022) Simulation and validation of optimized PID controller in AGV (Automated Guided Vehicles) model using PSO and BAS algorithms. Computational Intelligence and Neuroscience, 2022(1):7799654.

[6] Henclová K. (2019) Using CMA-ES for tuning coupled PID controllers within models of combustion engines. Neural Network World, 29(5):325-344.

[7] Chen JC, Chen TL, Teng YC. (2021) Meta-model based simulation optimization for automated guided vehicle system under different charging mechanisms. Simulation Modelling Practice and Theory, 106:102208.

[8] Krawczyk K, Arabas J. (2025) Single-objective surrogate models for continuous metaheuristics: An overview. Applied Sciences, 15(16):9068.

[9] Patil RS, Jadhav SP, Patil MD. (2024) Review of intelligent and nature-inspired algorithms-based methods for tuning PID controllers in industrial applications. Journal of Robotics and Control (JRC), 5(2):336-358.

[10] Aliskan I. (2025) The optimization-based fuzzy logic controllers for autonomous ground vehicle path tracking. Engineering Applications of Artificial Intelligence, 151:110642.

[11] Yu J, Liang M, Peng W, Wu T, Rong C, Zhang D. (2023) Speed control based on an improved PID controller with BP neural network for two wheel differential AGV system. 2023 IEEE 6th Student Conference on Electric Machines and Systems (SCEMS), 1-5.

[12] Ha VT, Thuong TT, Truc LN. (2024) Trajectory tracking control based on genetic algorithm and proportional integral derivative controller for two-wheel mobile robot. Bulletin of Electrical Engineering and Informatics, 13(4):2348-2357.

[13] Demir MH, Demirok M. (2023) Designs of particle-swarm-optimization-based intelligent PID controllers and DC/DC buck converters for PEM fuel-cell-powered four-wheeled automated guided vehicle. Applied Sciences, 13(5):2919.

[14] Jangid P, Nagar V, Sharma A, Nagar S, Palwalia DK. (2025) Design a PID controller based on grey wolf optimization algorithm for single-area load frequency control. Journal of Emerging Science and Engineering, 3(1):e36-e36.

[15] Mosaad AM, Attia MA, Abdelaziz AY. (2019) Whale optimization algorithm to tune PID and PIDA controllers on AVR system. Ain Shams Engineering Journal, 10(4):755-767.

[16] Van Niekerk JA, le Roux JD, Craig IK. (2023) On-line automatic controller tuning of a multivariable grinding mill circuit using Bayesian optimisation. Journal of Process Control, 128:103008.

[17] Tumpach J, Koza J, Holeňa M. (2023) Improving optimization with Gaussian processes in the covariance matrix adaptation evolution strategy. ITAT'23: Information Technologies – Applications and Theory, 3498(10).

[18] Khouzani FF, Mirzaei A, Plante PL, Gewali L. (2025) CMA-ES with radial basis function surrogate for black-box optimization. ITNG 2025. Advances in Intelligent Systems and Computing. 1463:367-379.

[19] Bujgoi G, Sendrescu D. (2025) Tuning of PID controllers using reinforcement learning for nonlinear system control. Processes, 13(3):735.

[20] Van Niekerk JA, le Roux JD, Craig IK. (2025) Reinforcement learning based automatic tuning of PID controllers in multivariable grinding mill circuits. Control Engineering Practice, 165:106522.