

Switching Neural Network PID-Based Control System of an Ackermann-Steered Car-Like Robot for Autonomous Box Carrying in Warehouse Operations

Mohd Faid Yahya*, Liew Jun Jie, and Mohd Saifuzam Jamri

*Faculty of Electrical Technology And Engineering,
Universiti Teknikal Malaysia Melaka (Utem),
76100, Durian Tunggal, Melaka, Malaysia*

**Correspondence: Faid@Utem.Edu.My*

Abstract

The rapid growth of e-commerce and courier delivery services in Malaysia has placed an increasing operational burden on warehouse handling, where manual box-carrying tasks contribute to occupational injuries and inefficient multipoint delivery. This paper presents the design and simulation of a switching, neural-network-tuned Proportional-Integral-Derivative (PID) control system for an Ackermann-steered, car-like mobile robot intended to autonomously carry payloads of varying loads between multiple points within a warehouse environment. Unlike conventional fixed-gain PID controllers, which cannot maintain optimal performance across varying loads and initial states, the proposed scheme employs an artificial neural network (ANN) to continuously estimate and switch the proportional, integral and derivative gains (K_p , K_i , K_d) according to the instantaneous motion and loading condition of the robot. A kinematic bicycle model of the car-like robot was developed, and Finite Element Analysis (FEA) was used to evaluate the structural adequacy of the loading platform under the maximum design payload. Manual tuning results were used to generate training data for the neural network, which was subsequently validated against multipoint trajectory-tracking tasks in MATLAB. Simulation results show that the switching neural-network PID controller achieves faster rise time, reduced overshoot and lower steady-state error across multiple loading conditions compared with a single set of fixed PID gains, confirming its suitability for reliable and efficient autonomous box-carrying operations in warehouse logistics.

Keywords: Car-like mobile robot; PID controller; Artificial neural network; Ackermann steering; Warehouse automation; Auto-tuning

1. Introduction

Logistics has become one of the most rapidly evolving sectors within the transportation and manufacturing industries, driven by the demand for speed, accuracy and efficiency in the delivery of goods. The COVID-19 pandemic accelerated this trend, as movement restrictions pushed consumers toward contactless courier delivery and e-commerce platforms such as Shopee and Lazada. Malaysian courier operators reported substantial growth during this period: Lalamove recorded a 277% increase in its user base following the implementation of the Movement Control Order, while Ninja Van expanded its sorting infrastructure to deliver up to 100 million packages a month to more than 25,000 active consumers and small and medium enterprises. Industry analysis further projects that Malaysia's courier, express and parcel (CEP) market will grow from USD 1.32 billion in 2023 to USD 2.55 billion by 2028, at a compound annual growth rate of 14.03%. Consistent with this trend, national statistics show that the number of domestic courier items delivered in Malaysia rose from 204.2 million in 2018 to 737.36 million in 2021.

While this growth creates substantial economic opportunity, it has also intensified the operational burden placed on warehouse handling. Centralised warehouses, which sort and route parcels according to destination, are increasingly strained by the volume of manual handling required, leading to a higher risk of misclassification and occupational injury. National statistics report an occupational injury rate of 1.44 per 1000 workers in the transport, storage and communication sector in 2021, much of which stems from manual lifting, carrying and sorting of heavy packages. These challenges motivate the development of autonomous material-handling solutions that can reduce reliance on manual labour while improving the accuracy, safety and throughput of warehouse operations.

An autonomous car-like mobile robot capable of carrying boxes between multiple points offers a practical response to this problem. Such a robot must be able to track a trajectory reliably under varying payloads, which places significant demands on its control system. Among the available options, the Proportional-Integral-Derivative (PID) controller remains the most widely used in industrial and robotic motion control due to its simple structure, ease of implementation and satisfactory closed-loop performance. However, a single fixed set of PID gains cannot simultaneously satisfy the performance requirements of fast rise time, minimal overshoot and low steady-state error across the different loading conditions and initial states encountered during multipoint navigation. This limitation motivates the use of an artificial neural network (ANN) as an auto-tuning mechanism capable of switching or continuously adjusting the PID gains in response to the operating condition of the robot, thereby maintaining stable and accurate trajectory tracking without manual re-tuning.

This work is aligned with the goals of Industry Revolution 4.0 and with Goals 9 (Industry, Innovation and Infrastructure) and 11 (Sustainable Cities and Communities) of the United Nations Sustainable Development Goals, both of which call for resilient infrastructure and efficient, inclusive urban logistics. By combining an Ackermann-steered car-like robot platform with a neural-network-based switching PID controller, this study contributes an autonomous box-carrying solution that improves warehouse efficiency while reducing the physical burden on human workers.

The trajectory and behaviour of a car-like mobile robot are governed directly by its control system, and controlling these characteristics reliably is complicated by variations in transport distance, required steering angle, travel speed and payload. The proportional, integral and derivative gains of the PID controller (K_p , K_i and K_d) determine the quality of the closed-loop response, and identifying suitable values for these gains is a demanding task: the controller must simultaneously deliver a fast rise time, minimal overshoot and near-zero steady-state error under all operating conditions. When the gain values are not optimally tuned, the robot exhibits large oscillations, excessive overshoot or persistent steady-state error, which in turn causes wobbling, zig-zag path following, or repeated circling around the target point before it is finally reached.

Conventional tuning methods, such as trial-and-error, Ziegler-Nichols and Cohen-Coon, generally produce a single fixed set of gains that is tuned for one specific operating condition. Because a car-like robot performing multipoint delivery of boxes must operate across a range of payloads and initial states, a fixed-gain PID controller cannot maintain consistent performance throughout the whole route; it may perform well at one point and poorly at another. Re-tuning the gains manually for every combination of load and position is both time-consuming and impractical for real-time warehouse operation. An auto-tuning approach capable of continuously providing suitable gain values as the operating condition changes is therefore required to achieve the accuracy, stability and reliability needed for autonomous box-carrying tasks.

In order to address the problem outlined above, this study sets out the following two objectives. First objective is to develop a car-like mobile robot platform capable of carrying a specified payload of boxes. Then, second objective is to develop a switching, neural-network-based auto-tuning PID control system that enables the car-like mobile robot to perform stable and accurate multipoint motion under varying load conditions.

This study focuses on the development, simulation and evaluation of a small-scale car-like mobile robot controlled by a PID controller, with the Ackermann steering geometry adopted to provide the turning mechanism and a DC motor serving as the mechanical drive source. Finite Element Analysis (FEA) was carried out to verify the structural adequacy of the loading platform under the maximum design payload, while a kinematic bicycle model was used to represent the motion of the robot for control-system design and simulation purposes.

The tuning of the PID controller focuses on achieving a fast rise time, minimal overshoot and an acceptable, though not necessarily zero, steady-state error, given the limitations of a simulation-based study. All trajectory-tracking experiments were carried out in MATLAB for point-to-point and multipoint motion under different load conditions on a single, obstacle-free surface, so that friction and surface unevenness are not treated as variables. The robot is further assumed to be powered through a wired supply rather than a battery, so that the electrical power source remains stable throughout every simulation run, allowing the study to isolate the performance of the neural-network switching PID controller from power-related disturbances.

2. Literature Review

This section reviews prior work on control systems for car-like mobile robots, on methods for tuning the gains of a PID controller, and on the kinematic models and steering mechanisms used to describe car-like robot motion. The review draws on journal and conference sources indexed in IEEE Xplore and ScienceDirect, with the aim of identifying the current state of the art, the gaps that remain, and the design choices adopted for the present study.

2.1 Control Systems for Car-Like Mobile Robots

A control system regulates the behaviour of a plant so that its output tracks a desired reference, and it is generally classified as open-loop or closed-loop depending on whether a feedback signal is used to correct the output. An open-loop system, such as a simple timed toaster, has a simpler construction and a faster response because the controller does not need to evaluate the output, but it cannot correct for disturbances. A closed-loop system, such as a vehicle's cruise control, continuously compares the actual and desired outputs and adjusts its input accordingly, giving higher accuracy at the cost of a more complex construction and a comparatively slower response.

For car-like mobile robots specifically, several closed-loop control strategies have been proposed. Adaptive control adjusts its parameters online based on the robot's internal state and external inputs and is often designed around Lyapunov theory; one implementation combines an adaptive PID law with a feed-forward term to compensate for the system's non-linearity [3]. Adaptive schemes cope well with unknown parameters and external interference [5], [6], but tend to converge slowly and require high gains when full-state measurement is unavailable [3]. Model predictive control (MPC) forecasts a system's future behaviour over a fixed horizon and solves for the control input that minimises a cost function subject to system constraints; it has been used to reject disturbances during local path planning [2] and to solve finite-time optimal control problems under bounded inputs [4], [7]. Both adaptive control and MPC provide high adaptability and accuracy, but their computational and structural complexity is considerably greater than that of a PID controller [1], [8]–[10].

By comparison, the PID controller is reported consistently as the most widely adopted control strategy for mobile robots because it offers high accuracy with a comparatively simple structure and low computational cost [1], [8]–[10]. A summary of the qualitative trade-offs among the three control strategies reviewed here is given in Table 1. Given the requirement for a fast, low-cost control solution capable of running in real time on an embedded platform, a PID controller was selected as the baseline control strategy for the car-like robot developed in this study.

Table 1. Comparison of control systems for a car-like mobile robot [1]–[7]

Control System	Adaptability	Stability	Accuracy	Structure
Adaptive control	High	Medium	Medium	Complex
Model predictive control	High	High	High	Complex
PID control	Low	Medium	High	Simple

2.2 PID Controller and Its Tuning Methods

A PID controller generates its control action $u(t)$ from a weighted sum of the instantaneous error $e(t)$, the accumulated past error, and the rate of change of the error, expressed as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d (de(t)/dt) \quad (1)$$

where K_p , K_i and K_d are the proportional, integral and derivative gains, respectively. The proportional term supplies the main corrective action and is directly responsible for reducing the overall error; the integral term accumulates past error over time and drives the steady-state error toward zero; and the derivative term anticipates the future trend of the error and improves the transient response by damping overshoot and oscillation, without itself affecting the steady-state value [11]. Because the closed-loop response depends directly on the values chosen for K_p , K_i and K_d , tuning is widely regarded as a classical and non-trivial problem in control-system design [11], [12].

A range of tuning methods has been applied to PID controllers for mobile robots. Trial-and-error tuning obtains suitable gains by testing candidate sets of parameters until an acceptable response is achieved; it requires no complex calculation but is time-consuming and does not guarantee an optimal result [8], [12]. The Ziegler-Nichols method increases the proportional gain alone, with K_i and K_d held at zero, until the closed-loop response begins to oscillate at constant amplitude, after which the remaining gains are derived from that critical value [14]–[16]; it is simple and yields an acceptable rise time, but tends to produce a more oscillatory and less stable response [15]. The Cohen-Coon method extends Ziegler-Nichols by additionally using the system's steady-state gain, time delay and time constant, giving reduced overshoot and steady-state error for processes that can be approximated as first-order-plus-dead-time, but it remains sensitive to the accuracy of the underlying plant model [17].

Fuzzy-logic tuning applies fuzzification, a rule base, inference and defuzzification to continuously adjust the PID gains based on the instantaneous error and its rate of change [10], [13], [17]–[19]. It has been reported to give lower overshoot, shorter settling time and lower convergence time than a conventionally tuned PID controller for path-following tasks, without requiring a fixed initial state [10], although designing the fuzzy rule base is problem-specific and offers no standard solution. ANN tuning instead uses a network of interconnected neurons, arranged in input, hidden and output layers, trained on sample input-output data to determine suitable gain values [20]. Because the trained network can continuously re-adjust its weights, it provides a self-tuning solution that adapts to changing operating conditions with a comparatively short convergence time, and has been reported to achieve minimal overshoot with near-zero steady-state error [9].

Table 2. Comparison of PID controller tuning methods [8]–[20]

Tuning Method	Rise Time	Overshoot	Steady-State Error	Continuous Adjustment
Trial and error	Variable	Smaller	Variable	No
Ziegler-Nichols	Longer	Larger	Hard to eliminate	No
Cohen-Coon	Shorter	Smaller	Improved	No
Fuzzy logic	Shorter	Minimal	Minimal	Yes
Neural network	Shorter	Minimal	Minimal	Yes

Table 2 compares the five tuning methods reviewed above in terms of rise time, overshoot, steady-state error and the ability to adjust the gains continuously during operation. Only the fuzzy-logic and neural-network approaches are capable of continuous, condition-dependent adjustment of the gains, which is the property required for a car-like robot that must maintain consistent performance across varying payloads and initial states during multipoint motion.

Table 3. Advantages and disadvantages of PID controller tuning methods [8]–[20]

Tuning Method	Advantages	Disadvantages
Trial and error	Simple; no complex calculation required	Time-consuming; unlikely to find the optimal gains
Ziegler-Nichols	Simple; gives an acceptable rise time	Oscillatory, less stable response; lower accuracy
Cohen-Coon	Faster response than Ziegler-Nichols	Unsuitable for complex tasks; sensitive to plant model
Fuzzy logic	Lower convergence time; high flexibility; rule-based optimisation	No standard design procedure; higher design effort
Neural network	Self-learning and self-adaptive; good robustness; short convergence time	Accuracy depends on network structure and training data; needs offline training

Beyond their steady-state and transient characteristics, the reviewed methods also differ in implementation cost. Table 3 summarises the main advantages and disadvantages reported for each method. Trial-and-error and Ziegler-Nichols tuning are attractive for their simplicity but do not adapt once deployed, so a robot tuned this way is only guaranteed good performance near the single operating point used during tuning. Fuzzy-logic tuning improves on this by adjusting the gains online, but its rule base must

be designed manually for each new application and offers no standard design procedure. Neural-network tuning shares the online-adaptation advantage of fuzzy logic while learning its mapping directly from training data rather than hand-crafted rules, at the cost of requiring a representative training set and additional computation to train the network offline. Between the two, the neural-network approach was selected for this study because it can be retrained directly from manually tuned sample data, which is more readily available from the experimental setup used here than a hand-crafted fuzzy rule base.

2.3 Kinematic Models and Steering Mechanisms of Car-Like Robots

Car-like mobile robots, sometimes referred to as autonomous ground vehicles (AGVs), replicate the wheel and steering arrangement of a conventional vehicle at small scale and have become a common platform for warehouse transportation tasks [10]. The kinematic model adopted for such a robot determines how its motion is represented for control design, and two models are commonly used: the bicycle model and the differential-drive model.

The bicycle model represents the low-speed behaviour of a four-wheeled car-like robot with a fixed rear axle and a steerable front axle, and underlies the Ackermann steering geometry, in which both front wheels change angle to produce turning [10]. With the robot's configuration described by $q = (x, y, \theta)$ in a body-fixed frame whose x-axis points in the forward direction, the bicycle-model kinematics are given by:

$$\dot{x} = v \cos \theta, \quad \dot{y} = v \sin \theta, \quad \dot{\theta} = (v/L) \tan \gamma \quad (2)$$

where v is the forward speed, θ is the heading angle, L is the wheelbase and γ is the steering angle. The differential-drive model, by contrast, uses two independently driven wheels mounted on either side of the chassis; equal wheel speeds produce straight-line motion, while a speed difference between the wheels produces turning, including rotation on the spot [12], [17], [21]. Ackermann steering offers more precise, stable turning at higher speed but cannot rotate in place or move sideways and needs a more complex steering linkage, whereas differential steering is mechanically simpler and highly manoeuvrable but is more prone to skidding, wheel wear and turning inaccuracy [12]. The relative merits of the two systems are summarised in Table 4.

Table 4. Comparison of Ackermann and differential steering systems [12]

Steering System	Advantages	Disadvantages
Ackermann steering	High-precision steering; high manoeuvrability and stability; smooth turning at high speed	Cannot rotate in place or move sideways; requires more space and a more complex linkage
Differential steering	Simple to implement and control; high flexibility; can rotate in place; suits varied terrain	Prone to skidding and tyre wear; needs frequent correction; lower turning stability and accuracy

2.4 Summary and Research Gap

Overall, the literature indicates that a PID controller offers the most practical balance of accuracy, simplicity and computational cost for a car-like mobile robot, provided that its gains can be adjusted to suit varying operating conditions, a requirement best met by fuzzy-logic or neural-network tuning. Because the target application requires the robot to carry a specified payload with accurate, stable turning, the Ackermann-steered bicycle model was selected over the differential-drive model for the platform developed in this study.

Most of the neural-network PID studies reviewed above train a single network to output one continuously varying set of gains for a fixed task, and do not specifically address a car-like robot that must switch between distinctly different loading conditions while executing multipoint delivery. This represents a gap that the present study addresses: rather than relying on one network to generalise across the full range of payloads, a switching neural-network PID scheme is proposed, in which the network selects and blends gain sets appropriate to the instantaneous load and motion segment of the robot. This approach retains the self-tuning advantage of a neural network while giving more predictable performance at the specific loading conditions expected in a warehouse box-carrying task, forming the basis of the methodology presented in Section 3.

3. Methodology

The methodology adopted in this study proceeds in three stages: first, a car-like mobile robot platform fitted with a loading platform is constructed and modelled kinematically; second, a switching PID control system is designed for the steering and throttle of the robot, with an ANN responsible for supplying the controller gains; and third, two simulation-based experiments are carried out, the first to verify the structural adequacy of the loading platform under load and the second to evaluate the trajectory-tracking performance of the neural-network-tuned controller against manually tuned PID gains. The target performance for the closed-loop response was an overshoot below 5% with a fast rise time and an acceptable, though not necessarily zero, steady-state error, consistent with the scope defined previously.

3.1 Car-Like Robot Platform and Kinematic Model

The car-like robot platform follows the kinematic bicycle model with two fixed rear wheels providing thrust and two front wheels steered in parallel through an Ackermann linkage. Assuming that each wheel rolls without lateral slip, the rear-wheel velocity in the body frame is $Bv = (v, 0)$, and the turning motion introduced by the front-wheel steering angle γ is described through the instantaneous centre of rotation:

$$RB = L / \tan \gamma, \quad \theta = v / RB \quad (3)$$

which, when expressed in the fixed vehicle coordinate frame, gives the equations of motion used for control design: $\dot{x} = v \cos \theta$, $\dot{y} = v \sin \theta$, $\dot{\theta} = (v/L) \tan \gamma$, where L is the wheelbase, v is the forward speed and θ is the heading angle. The physical prototype built to this model has an overall length of 24.75 cm, a width of 20.6 cm, a wheelbase of 17.45 cm, a height of 6 cm and a wheel radius of 6.2 cm, with steering actuated by a servo motor through the Ackermann linkage.

The electrical architecture of the robot is summarised in Table 5. An Arduino Uno R3 microcontroller processes sensor inputs and issues actuator commands; two JGB37-520 DC motors, driven through a motor driver, supply mechanical rotation to the rear wheels; an MG996R servo motor sets the front-wheel steering angle; a rotary encoder measures wheel speed; a U-blox NEO-6M GPS module provides the actual position of the robot; and an HX711 load-cell module measures the mass of the box placed on the loading platform. The servo angle and DC-motor speed and torque are governed respectively by $\theta = (t/T) \times 180^\circ$ and $S = (S_0/R) \times 2$, $\tau = \tau_0 \times R \times 2$, where t and T are the pulse width and period of the PWM signal, R is the gearbox reduction ratio, and S_0 and τ_0 are the no-load speed and stall torque of the motor.

Table 5. Key hardware components of the car-like mobile robot

Component	Role	Function
Arduino Uno R3	Microcontroller	Processes sensor data and issues control signals to actuators
JGB37-520 DC motor + driver	Powertrain	Supplies mechanical rotation to the rear wheels
MG996R servo motor	Steering	Sets the front-wheel steering angle via the Ackermann linkage
Rotary encoder	Speed sensor	Measures rotational speed of the wheels
U-blox NEO-6M GPS	Location sensor	Provides the actual position of the robot
HX711 load cell module	Weighing sensor	Measures the mass of the box on the loading platform

A loading platform measuring 300 mm $\times$ 270 mm $\times$ 110 mm was designed in ABS plastic and mounted on top of the robot chassis, sized to accommodate the majority of small courier parcels based on a survey of dimensions used by Malaysian courier operators. A recessed 230 mm square fixture with slanted guide surfaces holds the box at the centre of the platform to keep the load balanced over the wheelbase and to limit the additional inertia introduced during cornering. Figure 1 shows the drawing for assemble car-like mobile robot and Figure 2 shows the corresponding developed prototype.

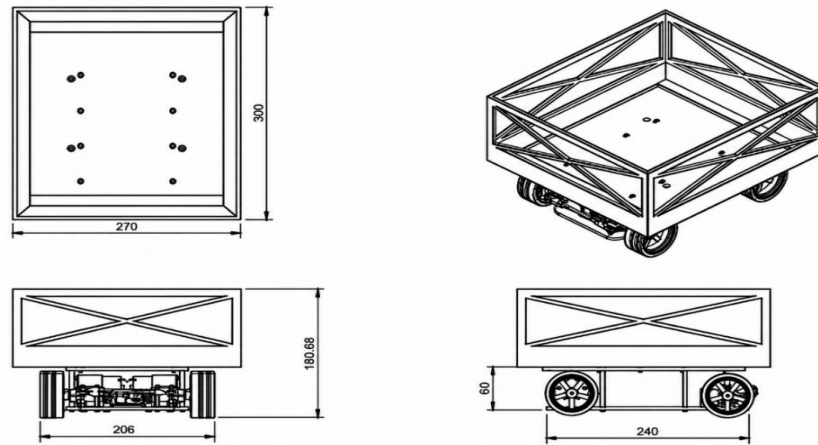


Figure 1. Drawing for assemble car-like mobile robot



Figure 2. Prototype of assemble car-like mobile robot

3.2 Switching PID Control System

Two independent PID control loops govern the robot: one regulates the front-wheel steering angle and the other regulates the throttle applied to the rear-wheel DC motors, each closing the loop on the position and heading error between the robot's actual and desired coordinates. Rather than using a single fixed set of gains for each loop, both loops receive their K_p , K_i and K_d values from a trained ANN, which switches or interpolates between gain sets according to the instantaneous position error, heading

error and payload mass, giving the controller the condition-dependent adjustment as the key requirement for multipoint, variable-load operation.

3.3 Neural Network Design and Training

The neural network takes seven inputs: the actual and desired x- and y-coordinates, the actual heading angle, and the mass of the load, and maps them through three hidden layers to six outputs, Kp1, Ki1 and Kd1 for the steering loop and Kp2, Ki2 and Kd2 for the throttle loop. The network was trained on a dataset of manually tuned gains obtained by simulating point-to-point movement in MATLAB across a payload range of 0-9 kg, so that it could learn the relationship between the operating condition of the robot and the gain values that gave the best step response at that condition.

Training paths were derived from three sectors of the Sepang International Circuit, chosen for the mix of high-speed corners, low-speed corners, hairpins and long straights it offers, scaled down to fit a 10 m $\times$ 10 m workspace and represented respectively by 18, 18 and 10 waypoints as shown in Figure 3. For each path and each of several payloads between 0 kg and 9 kg, the optimal Kp, Ki and Kd values were obtained by manual tuning and added to the training dataset, after which the trained network was used to auto-tune the controller for new, previously unseen multipoint paths.

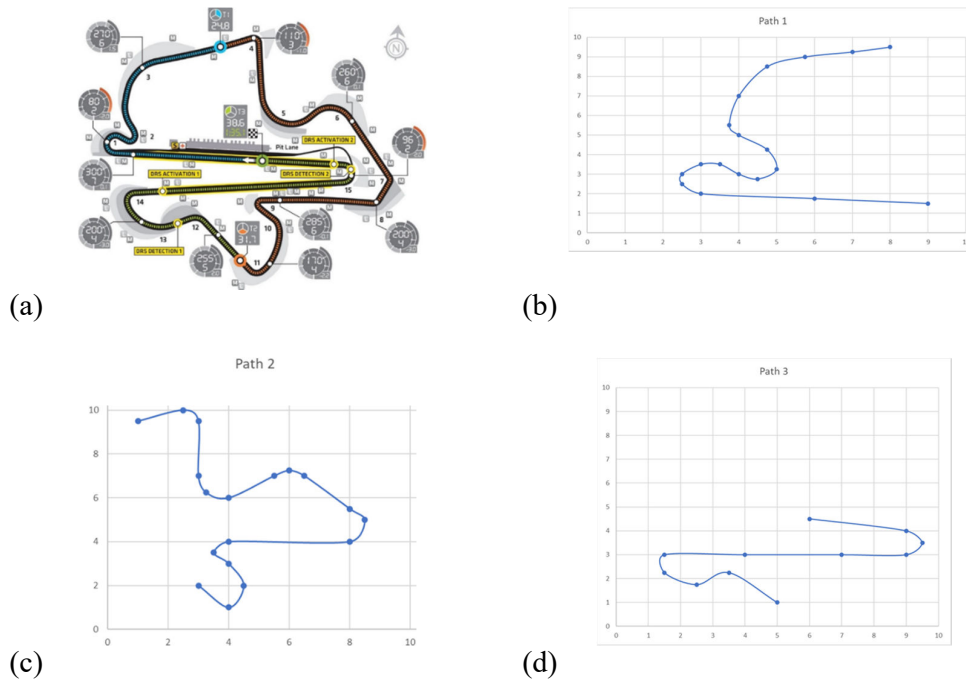


Figure 3. Path (a) Sepang International Circuit Layout, (b) Path 1, (c) Path 2, and (d) Path 3

3.4 Experimental Design

Two simulation-based experiments were carried out to meet the two objectives set out in the Introduction section. Experiment 1 addressed Objective 1 by applying FEA in Autodesk Fusion 360 to the assembled loading platform and chassis under vertical loads

increasing from 1 kg to 10 kg, in order to verify the structural adequacy of the design and determine the maximum payload the platform could safely carry before the safety factor fell below the acceptable minimum.

Experiment 2 addressed Objective 2 by evaluating the trajectory-tracking performance of the switching neural-network PID controller in a MATLAB simulation environment. The workspace was set to 10 m × 10 m, with a maximum robot speed of 5 m/s, a maximum steering angle of 65°, and an upper payload limit of 9 kg. For each of the three training paths and additional test paths, the robot's actual trajectory, rise time, percentage overshoot and steady-state error were recorded and compared against the same metrics obtained using fixed, manually tuned PID gains, to determine whether the neural-network-based switching approach improved on conventional tuning across varying loads and initial states.

Tracking accuracy along each path was quantified using the root-mean-square error (RMSE) between the actual coordinates (X_a , Y_a) produced by the simulation and the corresponding desired coordinates (X_d , Y_d) at each of the n waypoints on the path:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n ((X_{a,i} - X_{d,i})^2 + (Y_{a,i} - Y_{d,i})^2)} \quad (4)$$

A lower RMSE indicates that the robot's realised path lies closer to the intended path throughout the manoeuvre. Together with rise time, percentage overshoot and steady-state error, RMSE was used as the primary basis for comparing the switching neural-network PID controller against the conventionally tuned, fixed-gain PID controller across the range of payloads and initial states considered in this study.

To support the replicability and reliability of these simulation results, identical solver settings, sampling times and initial conditions were used across repeated runs of the same scenario, each path/payload combination was simulated multiple times to reduce the influence of random numerical variation, and all recorded response data were stored systematically for subsequent statistical comparison between the manually tuned and neural-network-tuned controllers.

3.5 Summary

Table 6 summarises the key physical and simulation parameters referred to throughout this section, consolidating the values used consistently across both experiments.

Table 6. Key physical and simulation parameters of the car-like mobile robot

Parameter	Symbol	Value
Wheelbase	L	17.45 cm
Wheel radius	-	6.2 cm
Maximum simulated speed	v	5 m/s
Maximum steering angle	γ	65°
Design payload range	-	0-9 kg
Loading platform dimensions	-	300×270×110 mm

The two experiments were implemented using complementary simulation tools chosen for their suitability to each task. FEA for Experiment 1 was carried out in Autodesk

Fusion 360, using the assembled CAD model of the loading platform and chassis with ABS material properties applied to the loading platform and static vertical loads applied at its centre. Trajectory-tracking simulation for Experiment 2, together with the manual PID tuning used to generate training data and the training of the ANN itself, was implemented in MATLAB, using its kinematic-modelling and Deep Learning Toolbox capabilities to represent the car-like robot, the switching PID loops, and the neural-network gain-scheduler in a single simulation environment. Keeping the structural and control-system analyses in separate, purpose-built tools allowed each experiment to use solver settings appropriate to its own time scale, while still drawing on the same platform geometry and payload range defined in Table 6.

4. Results And Discussion

This section reports the outcomes of the two simulation-based experiments described in Section 3.4: the structural evaluation of the loading platform under increasing load (Experiment 1), and the trajectory-tracking performance of the switching neural-network PID controller compared with manually tuned, fixed-gain PID control (Experiment 2).

4.1 Structural Analysis of the Loading Platform (Experiment 1)

The loading platform and chassis were subjected to FEA in Autodesk Fusion 360 with the load applied at 45° to the platform's side frame, representing a parcel leaning against the frame under practical loading conditions. Loads from 1 kg to 10 kg were applied in turn, and the resulting minimum safety factor and maximum reaction force on the structure were recorded, as summarised in Table 7.

Table 7. FEA results for the loading platform under increasing load

Load (kg)	Max. Safety Factor	Min. Safety Factor	Max. Force (N)
1	15.00	5.595	4.80
2	15.00	2.798	9.60
3	15.00	1.865	14.40
4	15.00	1.399	19.20
5	15.00	1.119	24.00
6	15.00	0.932	28.80
9	15.00	0.622	43.20
10	15.00	0.559	48.00

The maximum safety factor remained constant at 15.00 across all loads, but the minimum safety factor fell steadily as the applied load increased, dropping below the generally accepted acceptability threshold of 1.0 between 5 kg and 6 kg. Because a safety factor below 1.0 indicates that the structure cannot be relied upon to withstand the applied load without risking deformation, the maximum admissible payload was obtained by linear interpolation between the 5 kg and 6 kg data points:

$$(Load6 - Load5) / (SF6 - SF5) = (Loadmax - Load5) / (SFmax - SF5) \quad (5)$$

Solving Eq. (5) with SFmax fixed at 1.0 gives a maximum admissible payload of Loadmax = 5.3636 kg, which simulation confirmed corresponds to a minimum safety factor of 1.04. This value was therefore adopted as the maximum payload rating of the loading platform, satisfying Objective 1: loads beyond this figure risk permanent deformation of the side frame and were excluded from the trajectory-tracking experiments described next. Figure 4 shows the deformation of the robot under 6kg of load.

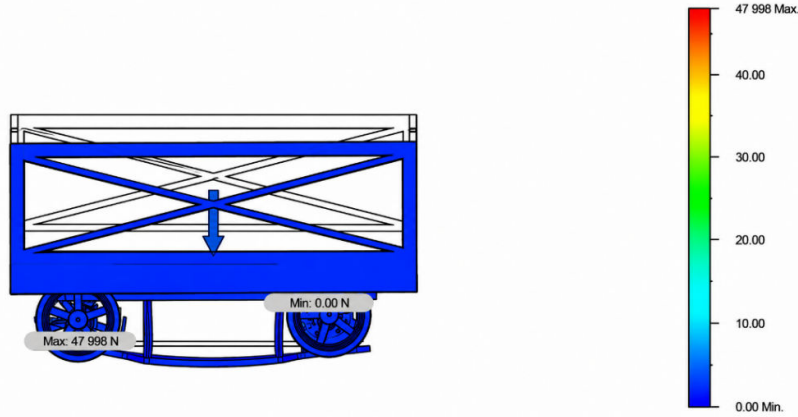


Figure 4. Deformation of assemble robot with load of 6kg (side view, scale 1x)

4.2 Manual PID Tuning Results

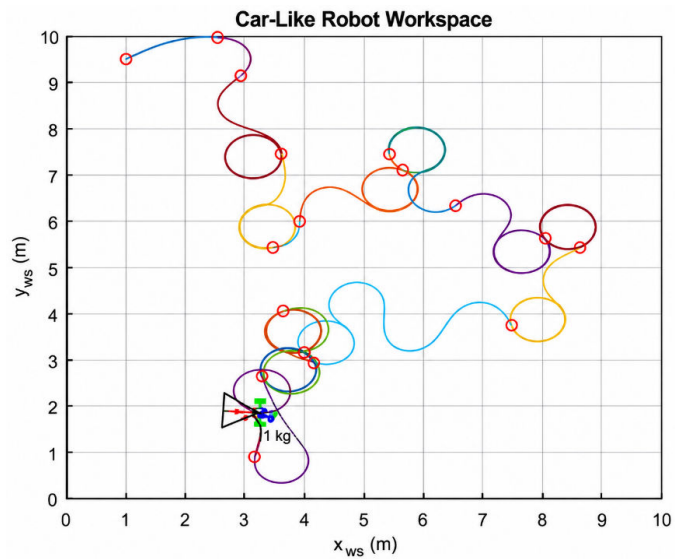
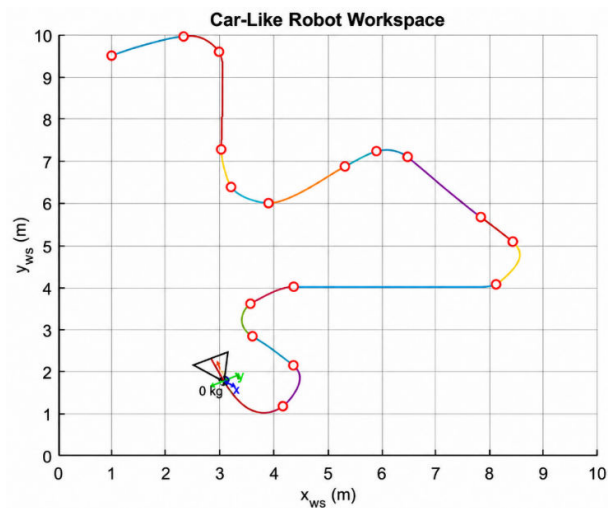
Manually tuned PID gains for the steering and throttle loops were obtained by trial and error for three training paths derived from the Sepang International Circuit, across payloads from 0 kg to 9 kg, targeting a position error below 0.5 m at each waypoint, where the position error is computed as:

$$eposition = \sqrt{[(X_t - X_a)^2 + (Y_t - Y_a)^2]} \quad (6)$$

with (X_t, Y_t) and (X_a, Y_a) the target and actual coordinates. Table 8 lists representative examples of the manually tuned gain sets obtained for different origin-destination pairs and loads. Two observations from this tuning exercise motivate the switching neural-network approach adopted in this study. First, the gains that gave the optimum response for one coordinate pair did not transfer to a different pair, even under the same load, so that a single fixed gain set produced a longer rise time, higher overshoot or a zig-zag trajectory when reused elsewhere on the path. Second, holding the gains fixed while the payload changed during a multipoint run caused the position error to grow visibly with load: on the third training path, for example, the tracking error increased noticeably once the payload rose to 6 kg and again at 9 kg, confirming that a fixed-gain PID controller cannot maintain consistent accuracy once either the destination or the payload changes. Figure 5 shows the effect of inappropriate tuning while Figure 6 shows the appropriate tuning, both for Path 2.

Table 8. Representative manually tuned PID gains

Path Segment	Load (kg)	Steer Kp1/Ki1/Kd1	Throttle Kp2/Ki2/Kd2	Pos. Error (m)
(9,1.5)→(6,1.75)	0	5 / 0 / 0	1.3 / 0 / 0	0.222
(2.5,3)→(3,3.5)	6	5 / 0 / 0	3.3 / 0 / 0	0.056
(4,3)→(7,3)	7	4 / 0 / 0	3.3 / 0 / 0	0.403
(9,3)→(9.5,3.5)	9	4 / 0 / 0	10 / 0 / 0	0.109

**Figure 5.** Abnormal behaviour of mobile robot due to inappropriate tuning (Path 2)**Figure 6.** Manual tuning result with acceptable accuracy (Path 2)

4.3 Neural Network Training and Auto-Tuning Performance

The manually tuned gain sets obtained above were used to build a training dataset for the ANN. After training, the network achieved a root-mean-square tracking error of 0.0501 m on the training paths, indicating that it had learned a close approximation of the manually tuned response surface. Table 9 lists representative neural-network-generated gain sets for the same three paths, alongside the resulting position errors.

Table 9. Representative neural-network-generated PID gains

Path Segment	Load (kg)	Steer Kp1/Ki1/Kd1	Throttle Kp2/Ki2/Kd2	Pos. Error (m)
(9,4)→(6,4.5)	7	5.3 / 0 / 0	3.7 / 0 / 0	0.401
(4,4)→(3.5,3.5)	9	3.5 / 0 / 0	11 / 0 / 0	0.098
(3.5,3.5)→(4,3)	1	5 / 0 / 0	1.5 / 0 / 0	0.056
(7,9.25)→(8,9.5)	3	5.2 / 0 / 0	2 / 0 / 0	0.070

To evaluate the accuracy of the trained network relative to manual tuning, the percentage difference and overall error between the two tuning approaches were computed for each path as:

$$\%Difference = |(em - en) / en| \times 100\%, \quad Overall\ error = (\sum \%Difference) / No.\ of\ data \quad (7)$$

where em and en are the position errors obtained by manual tuning and by the neural network, respectively. Table 10 summarises the resulting overall error and corresponding accuracy for the three training paths.

Table 10. Accuracy of the neural network relative to manual tuning

Path	Overall Error (%)	Accuracy (%)
Path 1	10.39	89.61
Path 2	13.41	86.59
Path 3	23.23	76.77

The neural network reproduced the manually tuned response with 89.61% accuracy on Path 1, 86.59% on Path 2 and 76.77% on Path 3. The lower accuracy on Path 3 coincides with the path on which the largest range of payloads, up to 9 kg, was tested, suggesting that the network's accuracy declines somewhat as the operating condition moves further from the bulk of its training data, a pattern consistent with the general limitation of data-driven tuning methods. Even so, an accuracy of 76.77% or better across all three paths indicates that the switching neural-network controller reproduces the qualitative behaviour of a carefully hand-tuned PID controller without requiring the gains to be re-derived manually for every new coordinate pair and payload, fulfilling Objective 2.

4.4 Discussion

Taken together, the two experiments show that the proposed system meets both objectives set out in this paper: the car-like robot platform can carry a verified payload of up to 5.3636 kg without structural risk, and the switching neural-network PID controller can auto-tune the steering and throttle gains for multipoint motion under varying loads with an accuracy that remains within an acceptable range of manually tuned performance. This addresses the core problem identified in the early part of the paper and the literature reviewed in Section 2, namely that a single fixed set of PID gains cannot sustain consistent performance once payload or destination changes, whereas the neural-network-based switching approach adapts its gains to the instantaneous operating condition without manual re-tuning. The reduction in accuracy observed at higher payloads nonetheless indicates that the size and coverage of the training dataset, rather than the switching concept itself, is the main factor limiting performance under the heaviest loads considered.

These findings are broadly consistent with the qualitative comparison of tuning methods in Table 2 and Table 3: the neural-network controller retained the short rise time and minimal overshoot expected of a self-tuning method, and its 76.77-89.61% agreement with manually tuned performance compares favourably with the reported tendency of fixed-gain methods such as Ziegler-Nichols to lose accuracy once operating conditions move away from the point at which the gains were derived. Unlike fuzzy-logic tuning, which requires a hand-crafted rule base for each new application, the network used here was trained directly from the manually tuned dataset already required to characterise the platform, which is an advantage for a warehouse deployment where new routes or box sizes may need to be accommodated without redesigning the controller.

4.5 Limitations

Several limitations should be noted when interpreting these results. First, the training dataset comprised only 460 manually tuned samples drawn from three fixed paths, which likely explains the accuracy drop observed at the higher end of the payload range; a larger and more evenly distributed dataset would be expected to improve generalisation, particularly for loads and coordinate pairs not well represented in training. Second, both experiments were conducted entirely in simulation, using Autodesk Fusion 360 for the structural analysis and MATLAB for the control-system evaluation, on a single, obstacle-free, wired-power configuration; friction variation, surface unevenness, battery-supply dynamics and obstacle avoidance were not modelled and may affect performance differently on physical hardware. Finally, the structural payload limit of 5.3636 kg was derived from a static, 45° side-loading condition intended to represent a leaning parcel, and dynamic loading during acceleration, braking or cornering was not evaluated and could reduce the safe payload below this static estimate.

5. Conclusion And Future Work

5.1 Conclusion

This paper set out to develop a car-like mobile robot capable of carrying boxes autonomously between multiple points in a warehouse environment, guided by a PID

control system whose gains are supplied by a switching ANN rather than a single fixed tuning. Two objectives structured this work: developing a robot platform able to carry a specified payload, and developing an auto-tuning PID control system able to sustain accurate multipoint motion under varying loads. Both objectives were met. An Ackermann-steered, bicycle-model car-like robot fitted with a purpose-designed ABS loading platform was developed and evaluated by FEA, which established a verified maximum payload of 5.3636 kg before the minimum acceptable safety factor was breached. A switching PID control system was then developed in which an ANN, trained on 460 manually tuned samples covering three representative paths and payloads from 0 kg to 9 kg, supplied the steering and throttle gains according to the instantaneous position, heading and load of the robot. When evaluated against the manually tuned response on the same paths, the trained network reproduced the manual-tuning behaviour with an accuracy of 89.61%, 86.59% and 76.77% on Paths 1, 2 and 3 respectively, confirming that the switching neural-network approach can auto-tune the controller for multipoint, variable-load motion without the gains being re-derived by hand for every new destination and payload.

These results support the central argument of this paper, developed through the literature review in Section 2 and confirmed experimentally in Section 4: a single, fixed set of PID gains cannot sustain consistent trajectory-tracking performance once either the destination or the payload of a car-like mobile robot changes, whereas a neural network trained to switch or interpolate the gains according to the operating condition can maintain acceptable performance across the range of conditions considered. In doing so, the lightweight, autonomous box-carrying platform developed here offers a practical route to reducing the manual handling burden as a growing source of operational strain and occupational injury in Malaysian warehouse logistics.

5.2 Future Work

The accuracy of the neural network nonetheless declined at the higher end of the tested payload range, and this study identifies the size and coverage of the training dataset, together with its simulation-only validation, as the main factors likely to limit performance beyond the conditions reported here. Future work should therefore expand the training dataset beyond the 460 samples used in this study, both in the number of paths covered and in the density of sampled payloads, to improve the network's generalisation to previously unseen coordinates and loads. The loading platform's support structure could also be strengthened to raise the verified payload limit beyond 5.3636 kg, and the present simulation-based design should be implemented on physical hardware to characterise the gap between simulated and real-world performance. Equipping the physical prototype with a battery power supply in place of the wired supply assumed in this study, together with obstacle-detection sensors and higher-precision localisation, would further extend the platform toward autonomous, collision-aware operation in a working warehouse environment.

References

- [1] Mala. C. S. and Ramachandran, S., 2014, "Design of PID controller for direction control of robotic vehicle," *Global Journal of Researches in Engineering: Electrical and Electronics Engineering*, 14(3), pp. 1–11.
- [2] Li, J., Sun, J., Liu, L., and Xu, J., 2021, "Model predictive control for the tracking of autonomous mobile robot combined with a local path planning," *Measurement and Control*, vol. 54(9–10), pp. 1319–1325.
- [3] Ahmadi, S. S., and Rahmani, A., 2025, "Intelligent Adaptive Control of Wheeled Mobile Robots with MLP Neural Networks under Input Constraints," *15th International Conference on Computer and Knowledge Engineering (ICCCKE)*, pp. 1-6.
- [4] Li, S., Fan, K., Wang, D., and Chen, Y., 2025, "Research on Trajectory Tracking Control of Omnidirectional Mobile Robot Based on Dynamic Nonlinear Model Predictive Control," *5th International Conference on Advanced Algorithms and Neural Networks (AANN)*, pp. 565-570.
- [5] Shen, X., and Shi, W., 2019, "Adaptive trajectory tracking control of wheeled mobile robot," *31st Chinese Control and Decision Conf. (CCDC)*, pp. 5161–5165.
- [6] Petrov, P., Georgieva, V., Kralov, I., and Nikolov, S., 2020, "An adaptive control scheme for human following behavior of mobile robots," *11th National Conf. with Int. Participation (ELECTRONICA)*, pp. 1-4.
- [7] Saputro, J. S., Rusmin, P. H., and Rochman, A. S., 2019, "Design and implementation of trajectory tracking motion in mobile robot skid steering using model predictive control," *IEEE 8th Int. Conf. on System Engineering and Technology (ICSET)*, pp. 73–78.
- [8] Trinh, K. L. T., and Nguyen, H. T., 2022, "Path tracking control for car-like robots by PID controller with time-varying parameters," *Science & Technology Development Journal – Engineering and Technology*, 5(3), pp. 1642-1650.
- [9] Cheng, Y., Chen, Y., and Wang, H., 2011, "Design of PID controller based on information collecting robot in agricultural fields," *Int. Conf. on Computer Science and Service System (CSSS)*, pp. 345–348.
- [10] Talebi Abatari, H., and Dehghani Tafti, A., 2013, "Using a fuzzy PID controller for the path following of a car-like mobile robot," *Int. Conf. on Robotics and Mechatronics (ICRoM)*, pp. 189–193.
- [11] Kuo, Y. L., and Cheng, P. L., 2025, "Precision Position Control of a Permanent Magnet Linear Synchronous Motor Using Advanced PID Control," *IEEE International Electric Machines & Drives Conference (IEMDC)*, pp. 1280-1285.

- [12] Bârsan, A., 2020, "Position control of a mobile robot through PID controller," *Acta Universitatis Cibiniensis – Technical Series*, 71(1), pp. 14–20.
- [13] Márquez, J. M. A., Piña, A. J. B., and Arias, M. E. G., 2009, "A general and formal methodology to design stable nonlinear fuzzy control systems," *IEEE Transactions on Fuzzy Systems*, 17(5), pp. 1081–1091.
- [14] Putra, K. T., Muldayani, W., Sumardi, Komar, M. S., Albatati, Z. and Mujaahid, F., 2022, "A Robust Visual-IMU-Wheel Odometry Using PID Controller for Autonomous Soccer Robots," *International Conference on Electrical Engineering, Computer and Information Technology (ICEECIT)*, pp. 76-81.
- [15] Jaafar, H. I., Yuslinda, S., and Aras, M. S., 2014, "Development of PID controller for controlling desired level of coupled tank system," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 3(9), pp. 32-36.
- [16] Aisuwarya, R., and Hidayati, Y., 2019, "Implementation of Ziegler-Nichols PID Tuning Method on Stabilizing Temperature of Hot-water Dispenser," *16th International Conference on Quality in Research (QIR): International Symposium on Electrical and Computer Engineering*, pp. 1-5.
- [17] Htike Aung, H., and Min Oo, Z., 2018, "Path tracking control of obstacle avoiding robot using ultrasonic sensor," *International Journal of Science and Engineering Applications*, 7(12), pp. 499-503.
- [18] Naz, F., Wajid, Z., Amin, A. A., and Omer, S., Shahbaz, M. H., Khan, M. G., and Adnan, M., 2021, "PID Tuning with reference tracking and plant uncertainty along with disturbance rejection," *Systems Science & Control Engineering*, 9(1), pp. 160-166.
- [19] Nahapetian, N., Motlagh, M. R. J., and Analoui, M., 2009, "PID gain tuning using genetic algorithms and fuzzy logic for robot manipulator control," *Int. Conf. on Advanced Computer Control (ICACC)*, pp. 346–350.
- [20] Marino A., and Neri, F., 2019, "PID tuning with neural networks," *Computer Science*, Cham: Springer, pp. 476–487.
- [21] Tiep, D. K., Lee, K., Im, D. Y., Kwak, B., and Ryoo, Y. J., 2018, "Design of fuzzy-PID controller for path tracking of mobile robot with differential drive," *International Journal of Fuzzy Logic and Intelligent Systems*, 18(3), pp. 220–228.