الجمهورية الجزائرية الديمقراطية الشعبية

DEMOCRATIC AND POPULAR ALGERIAN REPUBLIC

وزارة التعليم العالي والبحث العلمي

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

جامعة الشهيد حمه لخضر الوادي

ECHAHID HAMMA LAKHDAR UNIVERSITY OF EL-OUED

كلية التكنولوجيا

Faculty of Technology

قسم الميكانيك

Department of Mechanical Engineering

In order to obtain an Academic Master's diploma in Technology

Specialty: Electromechanics

Them

Weight Adaptive Path Tracking Control for Autonomous Vehicles Based on PSO

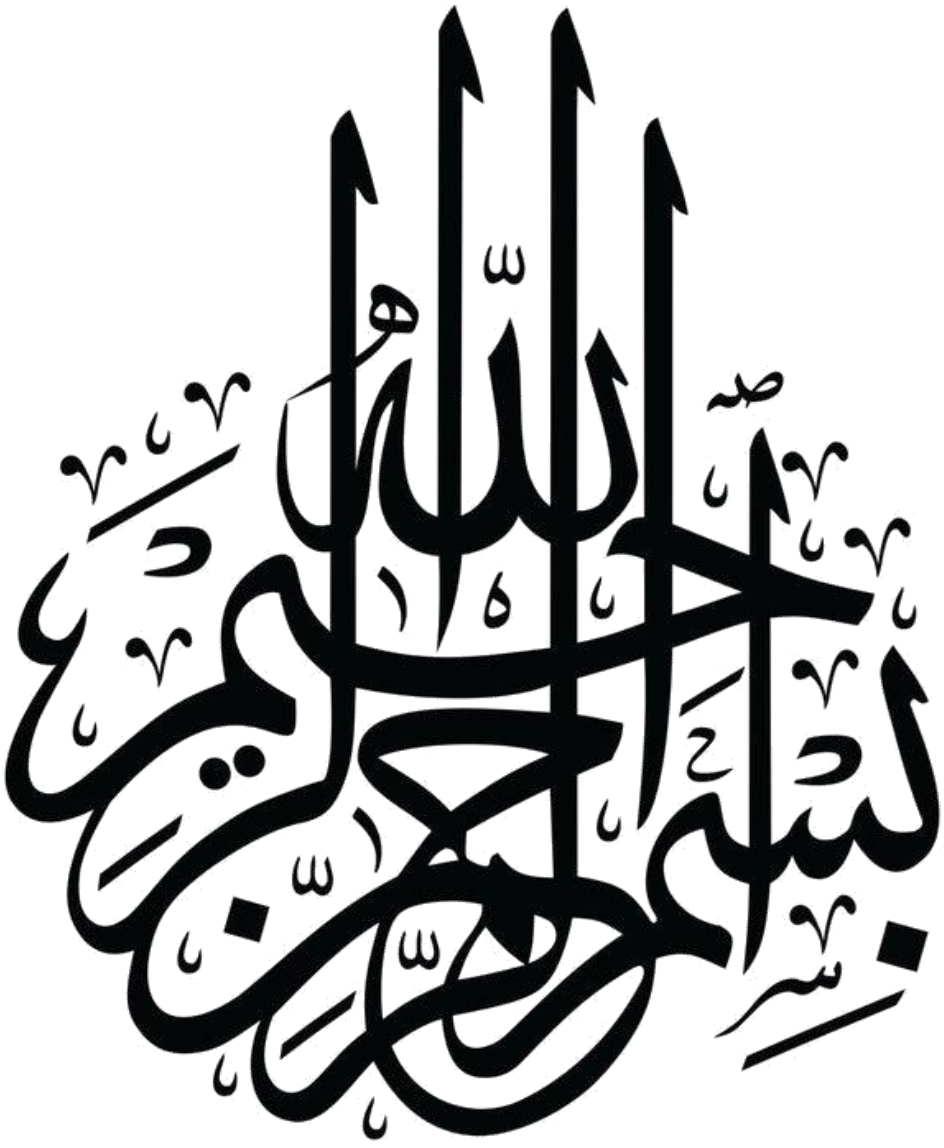
Presented by:

- **Zerig Ziad**
- **Ben Amara Aymen**
- **Tioua Yassine**

Jury Members:

President:	**Dr. Hamza Messei Aoune**	**MCA**	**El-Oued University**
Examiner:	**Dr. El Bachir Zin**	**MCA**	**El-Oued University**
Supervisor:	**Mr. Marouane Chetioui**	**MCA**	**El-Oued University**

University year: 2025/2026


بسم الله الرحمن الرحيم

Acknowledgments

Praise be to God, by whose blessing good deeds are perfected. To proceed:

In loyalty, appreciation, and acknowledgment of kindness, we extend our sincere thanks to those devoted individuals who spared no effort in assisting us in the field of academic research. We especially mention our supervisor, the virtuous Dr. Marouane chetioui, who guided, advised, and helped us in our studies and in completing our thesis. May God reward him with all good.

We also do not forget to extend our sincere thanks to all the professors who helped us in completing this research.

Finally, we offer our heartfelt gratitude to everyone who lent us a helping hand in completing this thesis in the best possible manner.

DEDICACES

We dedicate the fruit of our effort to those whom God has commanded us to honor with kindness and righteousness: our noble parents, may God prolong their lives and clothe them with health and well-being.

To those who have never faded from our memory and imagination: our brothers and sisters, each by name, with tenderness and affection.

To those who guided us along the paths of knowledge and learning: our teachers, in appreciation and respect.

To uncles and aunts (paternal and maternal), with love and family ties.

To all friends and colleagues, without exception.

To all these people, we dedicate this humble effort, asking God to make it beneficial for all Muslims.

Abstract

This thesis addresses the path-tracking problem for autonomous vehicles by proposing a Curvature-Aware Model Predictive Control (CAMPC) framework optimized using Particle Swarm Optimization (PSO). Conventional MPC controllers rely on fixed, manually tuned weighting parameters, which often result in suboptimal performance when operating under different road geometries and driving conditions. To improve tracking accuracy and control stability, the proposed CAMPC incorporates road curvature information directly into the prediction model and cost function.

The MPC weighting parameters are optimized offline using PSO within the CARLA simulation environment. During the optimization process, candidate weight sets are evaluated through repeated closed-loop simulations, where a cost function combining trajectory-tracking errors and control effort is minimized. The resulting optimal weight vector is then stored and employed by the MPC controller during real-time operation.

The proposed framework is evaluated in CARLA using three trajectories of increasing complexity, including a roundabout, a multi-curve route, and a long curvy road. The performance of the optimized CAMPC is compared with that of a conventional PID controller. Experimental results show that the proposed approach significantly improves trajectory-tracking performance, reducing maximum cross-track error by 51–61%, maximum heading error by 85–86%, and average speed error by 22–31%. In addition, the optimized controller decreases braking effort and eliminates steering oscillations, resulting in smoother and more stable vehicle behavior. These results demonstrate that PSO-based offline optimization provides an effective method for selecting MPC weighting parameters and enables robust path tracking for autonomous driving applications.

Keywords: Autonomous Vehicles, Model Predictive Control (MPC), Path Tracking, Particle Swarm Optimization (PSO).

الملخص

تتناول هذه الأطروحة مشكلة تتبع المسار للمركبات ذاتية القيادة من خلال اقتراح إطار عمل للتحكم التنبؤي النموذجي المُراعي لانحناء الطريق (CAMPC)، والمُحسَّن باستخدام خوارزمية تحسين سرب الجسيمات (PSO). تعتمد وحدات التحكم التنبؤية النموذجية التقليدية على معلمات ترجيح ثابتة يتم ضبطها يدويًا، مما يؤدي غالبًا إلى أداء دون المستوى الأمثل عند التشغيل في ظل ظروف هندسية مختلفة للطرق وظروف قيادة متنوعة. ولتحسين دقة التتبع واستقرار التحكم، يدمج إطار عمل CAMPC المقترح معلومات انحناء الطريق مباشرةً في نموذج التنبؤ ودالة التكلفة.

يتم تحسين معلمات ترجيح التحكم التنبؤي النموذجي خارجياً باستخدام خوارزمية PSO ضمن بيئة محاكاة CARLA. خلال عملية التحسين، يتم تقييم مجموعات الأوزان المرشحة من خلال عمليات محاكاة متكررة ذات حلقة مغلقة، حيث يتم تقليل دالة التكلفة التي تجمع بين أخطاء تتبع المسار وجهد التحكم. ثم يتم تخزين متجه الوزن الأمثل الناتج واستخدامه بواسطة وحدة التحكم التنبؤية النموذجية أثناء التشغيل في الوقت الفعلي.

يتم تقييم إطار العمل المقترح في CARLA باستخدام ثلاثة مسارات ذات تعقيد متزايد، تشمل دوارًا، وطريقًا متعدد المنحنيات، وطريقًا طويلًا متعرجًا. تُقارن أداء وحدة التحكم المُحسّنة CAMPC مع أداء وحدة تحكم PID التقليدية. تُظهر النتائج التجريبية أن النهج المقترح يُحسّن بشكل ملحوظ أداء تتبع المسار، حيث يُقلل من أقصى خطأ في المسار الجانبي بنسبة 51–61%، وأقصى خطأ في الاتجاه بنسبة 85–86%، ومتوسط خطأ السرعة بنسبة 22–31%. إضافةً إلى ذلك، تُقلل وحدة التحكم المُحسّنة من جهد الكبح وتُزيل تذبذبات التوجيه، مما يُؤدي إلى سلوك أكثر سلاسة واستقرارًا للمركبة. تُوضح هذه النتائج أن التحسين غير المتصل بالإنترنت القائم على خوارزمية PSO يُوفر طريقة فعالة لاختيار معلمات ترجيح MPC، ويُمكّن من تتبع المسار بكفاءة عالية لتطبيقات القيادة الذاتية.

كلمات مفتاحية: المركبات ذاتية القيادة، التحكم التنبؤي بالنموذج (MPC)، تتبع المسار، خوارزمية تحسين سرب الجسيمات (PSO).

Table of Contents

Table of Contents

List of Figures

List of Tables

Introduction general

Introduction General

In recent years, autonomous vehicles have attracted significant attention from both researchers and industrial companies due to their potential to transform modern transportation systems. The development of self-driving technologies aims to improve road safety, reduce traffic congestion, decrease fuel consumption, and provide better driving comfort. Since human error is considered one of the main causes of road accidents, autonomous driving systems are increasingly viewed as an effective solution for creating safer and more efficient transportation environments. The continuous progress in artificial intelligence, computer vision, embedded systems, and control engineering has accelerated the evolution of autonomous vehicles and brought them closer to real-world deployment.

An autonomous vehicle relies on several interconnected components to operate correctly in complex environments. Among these components, path tracking plays a fundamental role because the vehicle must follow a predefined trajectory accurately while preserving stability and passenger comfort. This task becomes more difficult in situations involving curved roads, lane changes, intersections, and sudden steering maneuvers. Any deviation from the desired path may affect vehicle safety and overall driving performance. For this reason, designing a robust and accurate control strategy remains one of the major challenges in autonomous driving research.

Model Predictive Control (MPC) is widely used in autonomous vehicle applications because of its ability to predict future vehicle behavior and manage system constraints efficiently. MPC calculates optimal control actions over a prediction horizon in order to minimize tracking errors and improve vehicle stability. Nevertheless, the effectiveness of MPC strongly depends on the selection of appropriate weights in the cost function. Poorly selected weights may produce unstable steering behavior, excessive oscillations, or inaccurate trajectory tracking. In practical applications, finding suitable weights manually is not straightforward, especially under changing driving conditions and different road geometries. Therefore, optimization methods have become an important tool for improving MPC performance automatically.

In this work, a Curvature-Aware Model Predictive Control (CAMPC) approach combined with Particle Swarm Optimization (PSO) and Neural Networks is proposed for autonomous vehicle path tracking in the CARLA simulator. The PSO algorithm is used during the offline phase to determine suitable MPC weights for different driving scenarios. The generated data

are then used to train a neural network capable of predicting optimal weights directly from the vehicle state. During online operation, the trained neural network replaces the optimization process and provides adaptive weights in real time, allowing the MPC controller to react efficiently to different road curvatures and driving situations.

This memoire is divided into three chapters. The first chapter presents a general overview of autonomous vehicles and their main functional layers, including perception, planning, and control. It also introduces the kinematic bicycle model and explains the Curvature-Aware Model Predictive Control approach together with its cost function formulation. The second chapter focuses on the proposed optimization and learning framework based on Particle Swarm Optimization and Neural Networks for adaptive weight tuning. The third chapter presents the obtained simulation results in the CARLA environment and discusses the performance of the proposed method in terms of trajectory tracking accuracy, stability, and control quality under different driving scenarios.

Chapter1

Autonomous vehicle pipeline with CAMPC

Introduction

This first chapter establishes the theoretical foundation upon which the entire thesis is built, presenting the fundamental concepts of autonomous vehicles and their control system architectures. It aims to provide the reader with the necessary scientific background to understand the challenges associated with path tracking, the importance of vehicle dynamic modeling, and the role of Model Predictive Control in enhancing autonomous driving performance. The chapter begins with a comprehensive definition of autonomous vehicles and their core components, then proceeds to review the typical architecture of autonomous driving systems, with particular emphasis on the control module as the final command executor and the translator of abstract plans into physical vehicle motions. The chapter also addresses vehicle modeling, introducing the kinematic bicycle model as a balanced compromise between dynamic fidelity and computational efficiency, followed by an exploration of curvature-based speed profile design and the mathematical formulation of path tracking errors. Finally, the Curvature-Aware Model Predictive Control (CAMPC) approach is presented as the central control strategy adopted in this work, along with a discussion of its advantages and limitations.

1.1 Introduction to Autonomous Vehicle

An autonomous vehicle (AV), also known as a self-driving vehicle or driverless vehicle, is an intelligent transportation system capable of perceiving its surrounding environment, making driving decisions, and controlling vehicle motion without continuous human intervention. Autonomous vehicles integrate several advanced technologies such as sensors, artificial intelligence, computer vision, machine learning, localization systems, and control algorithms in order to perform driving tasks automatically and safely. These tasks include lane keeping, obstacle avoidance, speed regulation, path planning, steering, braking, and navigation in complex traffic environments.

Figure 1.1: example of real autonomous vehicle.

Autonomous driving systems (ADS) represent a transformative integration of sensing, perception, planning, and control technologies designed to enable vehicles to navigate complex environments with minimal or no human intervention. These systems rely on a layered architecture that processes raw sensor data into actionable control commands, ensuring safety, efficiency, and robustness under dynamic conditions such as varying traffic, road geometries, and environmental disturbances. The control layer, in particular, serves as the final execution stage, translating high-level plans into low-level actuator inputs while accounting for vehicle dynamics and operational constraints. [2]

The architecture of an autonomous vehicle is typically modular, comprising three primary subsystems: (i) the algorithm subsystem (encompassing sensing, perception, localization, prediction, planning, and control), (ii) the client subsystem (including the real-time operating system and hardware platform for onboard computation), and (iii) the cloud subsystem (providing high-definition mapping, simulation, data storage, and model training). Sensors such as LiDAR, cameras, radar, GPS/IMU, and ultrasonic devices feed into perception and localization modules, which generate a comprehensive environmental model and precise vehicle pose estimation. Planning modules then produce reference paths or trajectories, while the control module executes these plans. This modular design allows for scalability and fault tolerance, contrasting with end-to-end learning-based architectures that map raw sensor inputs directly to control outputs [3].

In practice, the modular architecture dominates commercial and research prototypes due to its interpretability and verifiability. For instance, perception fuses multi-modal sensor data to detect objects and map the environment, localization employs techniques like GNSS fusion or

LiDAR-based map matching for centimeter-level accuracy, and motion planning generates collision-free trajectories considering vehicle kinematics and traffic rules. The control module interfaces directly with actuators (steering, throttle, and braking), closing the loop by minimizing deviations from the planned path while respecting real-time constraints. This hierarchical structure ensures that uncertainties at higher levels (e.g., prediction errors) are mitigated at the control level through feedback and predictive optimization. [4]

The role of the control module is pivotal as the lowest-level executor in the ADS pipeline, responsible for converting reference trajectories or paths generated by the planning layer into precise actuator commands. It must handle nonlinear vehicle dynamics, actuator limitations, and external disturbances (such as road friction variations or wind) while guaranteeing stability, comfort, and adherence to safety bounds. Unlike higher-level modules focused on global decision-making, the control module operates at high frequency (often 10–100 Hz) to provide immediate corrective actions, acting as a bridge between abstract plans and physical vehicle behavior. In path-tracking applications, it optimizes control inputs (e.g., steering angle and longitudinal acceleration) over a prediction horizon, making it especially suited for techniques like Model Predictive Control (MPC). [5]

The path tracking problem is formally defined as the task of designing a controller that enables the vehicle to follow a predefined reference path (or time-parameterized trajectory) with minimal lateral and heading errors, subject to vehicle dynamics constraints and safety requirements. Mathematically, given a reference path $ref(s)$ parameterized by arc length s, the controller minimizes cross track error cte and heading angle error e_ψ while respecting bounds on states (e.g., yaw rate, sideslip angle) and inputs (e.g., steering rate,throttle,brake). This problem distinguishes between path following (position-only tracking without explicit timing) and trajectory tracking (time-dependent reference). It is inherently challenging due to nonholonomic vehicle constraints, parameter uncertainties (tire cornering stiffness, mass), and real-time computational demands, particularly at high speeds or in curved roads [6].

Effective solutions to the path tracking problem, such as MPC, leverage a predictive vehicle model (often a bicycle or dynamic model) to forecast future states and optimize a cost function that penalizes tracking errors and control effort. This formulation inherently handles multivariable interactions and hard constraints, outperforming classical methods like PID or pure pursuit in complex scenarios. The introduction of autonomous vehicle control thus sets

the foundation for advanced techniques like MPC, which address the inherent limitations of traditional controllers by incorporating vision and constraint awareness.

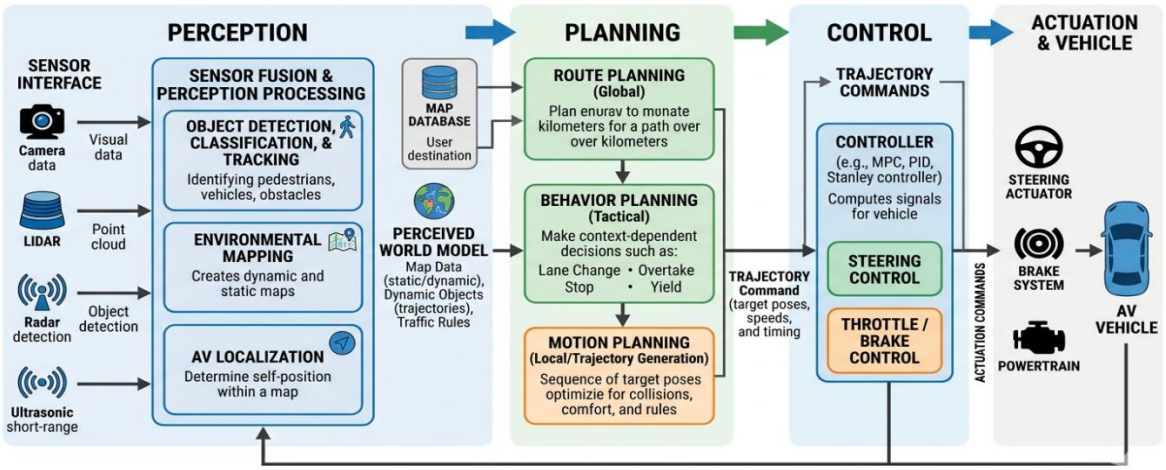


Figure 1.2: Overview of the Autonomous Vehicle Software Stack, illustrating the interaction between Perception, Planning, and Control modules.

1.2 Vehicle Modeling for Control Design

Vehicle modeling plays a fundamental role in the design of control systems for autonomous vehicles, serving as the cornerstone for predictive strategies such as Model Predictive Control (MPC). An accurate mathematical representation of vehicle dynamics allows the controller to forecast future states over a finite horizon, optimize control inputs, and explicitly enforce constraints related to actuator limits, stability, and safety. Without a suitable model, MPC cannot reliably minimize tracking errors or handle the multivariable, constrained nature of path tracking in real-time environments, leading to degraded performance or instability under disturbances like varying road conditions or high speeds. The choice of model balances fidelity with computational efficiency, as overly complex models hinder real-time solvability while overly simplistic ones fail to capture essential behaviors. [7]

The kinematic bicycle model is a widely employed simplification in autonomous vehicle control, particularly for MPC-based path tracking at low-to-moderate lateral accelerations. It reduces the four-wheel vehicle to a single-track (bicycle-like) representation by lumping the left and right wheels on each axle into one front wheel and one rear wheel. This abstraction captures the essential nonholonomic constraints of wheeled locomotion while remaining analytically tractable for optimization. The model is derived purely from geometric and kinematic relationships, ignoring tire-road interaction forces, and is especially popular in

research and prototyping because it enables fast prediction and real-time implementation in embedded systems. [8]

In the kinematic bicycle model, the primary state variables are the global position coordinates of the rear axle (x,y) the vehicle heading angle ψ, and the longitudinal velocity at the rear axle v. An auxiliary state, the sideslip angle β at the center of gravity, is often included to improve accuracy without introducing full dynamics. The steering behavior is controlled by the front wheel steering angle δ_f and the longitudinal behavior is controlled by acceleration a (which combines throttle and braking commands). These states and inputs fully describe the vehicle's planner motion under the bicycle model approximation, with the wheelbase $l = l_r + l_f$ (distances from the center of gravity to the front and rear axles, respectively) as the key geometric parameter. [9]

The continuous-time equations governing the kinematic bicycle model (with the rear axle as reference point and small-angle approximations where applicable) are expressed as:

$$\begin{cases} \dot{x} = v\cos(\psi + \beta) \\ \dot{y} = v\sin(\psi + \beta) \\ \dot{\psi} = \dfrac{v}{l_r + l_f}\cos(\beta)\tan(\delta_f) \\ \dot{v} = a \end{cases} \quad (1)$$

These differential equations are discretized (typically via forward Euler integration) for use in the MPC optimization problem. [10]

The kinematic bicycle model relies on several key assumptions that enable its simplicity: (i) no lateral tire slip (wheels roll purely in their heading direction), (ii) negligible pitch, roll, and vertical dynamics, (iii) constant or slowly varying longitudinal velocity in many implementations, and (iv) small steering and sideslip angles allowing linear approximations such as $\tan(\delta_f) \approx \delta_f$.

These assumptions hold under moderate operating conditions where lateral acceleration remains below approximately 0.5 g (where g is gravitational acceleration) and tire forces stay within the linear regime. [11]

Despite its advantages, the kinematic bicycle model has notable limitations that restrict its applicability. It neglects tire compliance, load transfer, and dynamic effects such as yaw-rate coupling or sideslip-induced forces, making it inaccurate during high-speed cornering, emergency maneuvers, or on low-friction surfaces where slip angles become significant. At

higher lateral accelerations (>0.5 g), the model can generate infeasible trajectories or unstable predictions, necessitating fallback to dynamic bicycle models (incorporating Pacejka tire formulas) or hybrid approaches. In MPC contexts, these limitations are mitigated by constraint tuning, model adaptation, or switching to higher-fidelity models when conditions exceed the kinematic regime, ensuring robustness while preserving computational efficiency for real-time path tracking. [12]

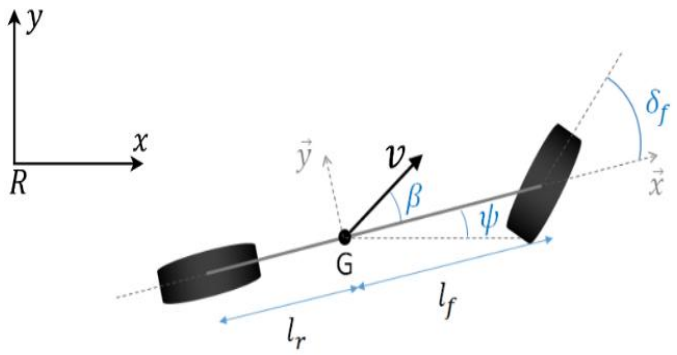


Figure 1.3: Kinematic Bicycle Model Representation for Vehicle Dynamics.

1.3 Curvature and Speed Profile Design

Speed profile and curvature are widely used in the autonomous navigation domain to ensure safe, smooth, and efficient vehicle motion along complex road geometries. By incorporating the geometric characteristics of the path, autonomous vehicles can anticipate upcoming turns, reduce lateral acceleration, and optimize passenger comfort. This curvature-aware speed planning strategy has been widely validated in path tracking, motion planning, and highway automation systems [13][14][15].

Curvature-based speed design relies on a fundamental lateral dynamics constraint that relates the vehicle's lateral acceleration to the square of its longitudinal velocity and the curvature of the reference path:

$$a_y = v^2 k \qquad (2)$$

where a_y is lateral acceleration, v is the longitudinal speed, and k is the curvature of the road. This expression comes directly from the steady-state cornering dynamics of the bicycle

model and reflects the physical relationship between turning radius, speed, and lateral force demand. As curvature increases the required lateral acceleration grows quadratically with speed.

To respect the vehicle's physical limits and ensure passenger comfort, the maximum allowable speed along the path is therefore computed by imposing a constraint on lateral acceleration, typically defined by tire friction limits or comfort thresholds. This gives the curvature-limited speed formulation:

$$v = \sqrt{\frac{a_{y,max}}{|k|}} \qquad (3)$$

Where $a_{y,max}$ denotes the lateral acceleration, v signifies the vehicle's longitudinal velocity, and k represents the curvature of the trajectory. Increased curvature (i.e., sharper curves) results in greater lateral acceleration at a specific speed. To ensure stability and comfort, the vehicle must decrease speed as curvature intensifies, while it may safely accelerate on straight or gently curved sections. This curvature-dependent speed regulation provides several advantages:

- Safety**:** Prevents exceeding lateral tire-force limits, reducing the risk of skidding.
- Comfort**:** Minimizes sudden lateral jerks and ensures smooth moving into curves.
- Energy Efficiency**:** Reduces unnecessary braking and acceleration cycles.
- Improved Tracking**:** Provides the controller with feasible velocities, preventing saturation of steering actuators during tight curves.

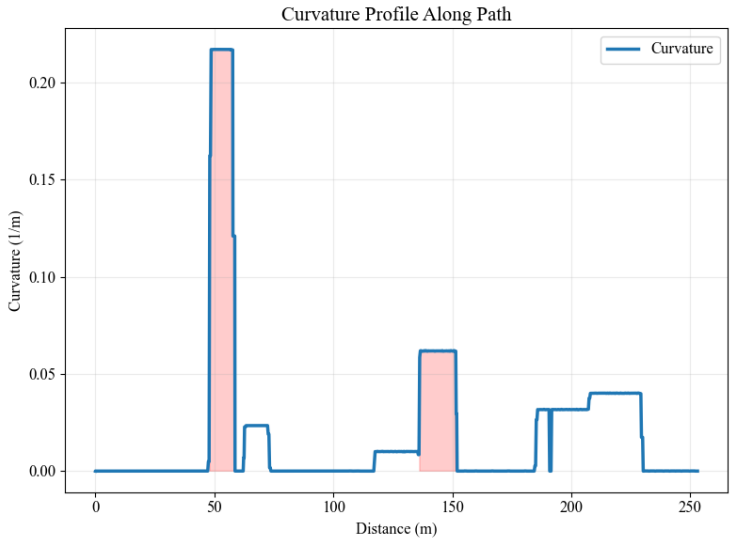


Figure 1(a). Curvature Profile Along the Trajectory

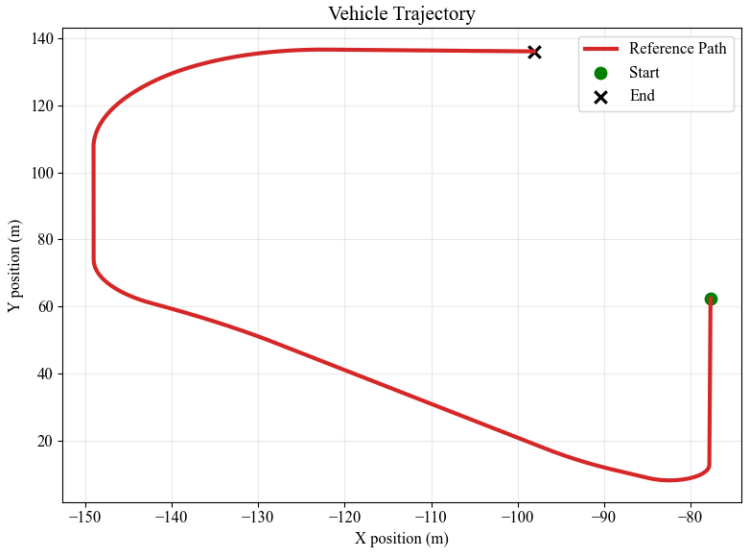


Figure 1(b). Reference Trajectory of the Autonomous Vehicle

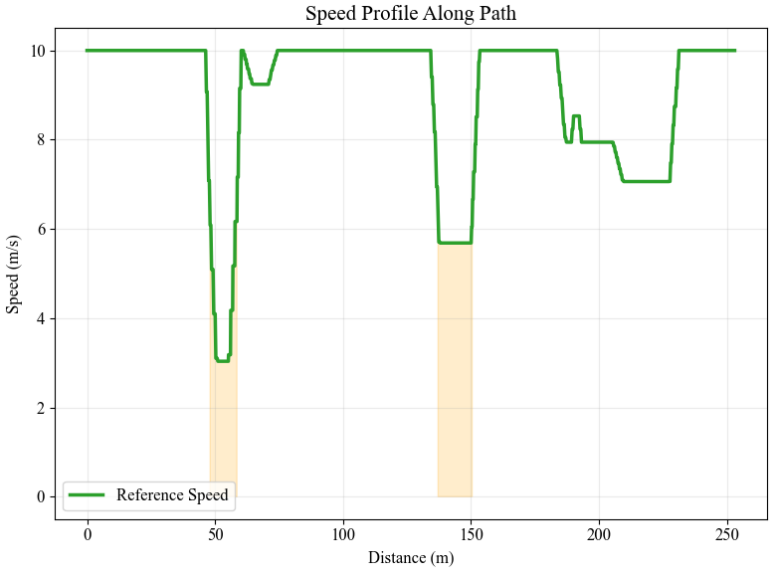


Figure 1(c). Speed Profile Adapted to Path Curvature

Figure **1(a)** depicts the curvature profile of the reference path, revealing the geometric complexity encountered along the route. Several sharp peaks appear at locations corresponding to tight turns, while extended flat regions represent straight or gently curving sections. These variations highlight the importance of curvature-aware trajectory planning for autonomous vehicles.

Figure **1(b)** presents the reference trajectory in Cartesian coordinates, which includes straight segments, wide-radius bends, and pronounced turning maneuvers. This diverse geometric structure ensures a comprehensive evaluation of the proposed controller under different driving conditions.

Figure **1(c)** shows the curvature-based speed profile generated from the lateral acceleration constraint. As expected, the vehicle's speed decreases markedly in high curvature regions corresponding to the peaks in Figure 1(a) and increases along straight stretches of the path. This adaptive behavior demonstrates how curvature-aware speed planning improves safety and smoothness by proactively regulating velocity according to road geometry.

1.4 Path Tracking Error Formulation

In path-tracking control, the vehicle must accurately follow the reference trajectory while maintaining the correct orientation relative to the desired path. To quantify the deviation between the vehicle motion and the reference trajectory, two principal error metrics are

commonly defined: the cross-track error (CTE) and the heading error (e_ψ). These quantities are computed using the vehicle center position (x_c, y_c) and two successive reference waypoints W_t and W_{t+1}, as illustrated in Figure 3. The formulation of these tracking errors is fundamental in Model Predictive Control (MPC) for autonomous vehicles because it converts the geometric deviation between the current vehicle state and the reference trajectory into measurable state variables incorporated into the optimization cost function and system constraints. In most practical implementations, the errors are expressed in a local coordinate system attached to the reference path, such as the Frenet–Serret frame, allowing the controller to predict and minimize future deviations while accounting for upcoming road geometry. This representation improves tracking accuracy, enhances vehicle stability, and maintains computational efficiency, particularly during high-speed maneuvers or curved-road driving conditions where global Cartesian coordinates become less suitable. Furthermore, the use of properly defined tracking errors facilitates the partial decoupling of lateral and longitudinal vehicle dynamics in MPC-based autonomous driving systems [12].

1.4.1 Cross Truck Error

The cross-track error measures the **lateral displacement** between the vehicle's center (x_c, y_c) and the reference line segment joining W_t and W_{t+1} This segment can be expressed analytically as:

$$ax + by + c = 0 \qquad (4)$$

and the CTE is defined as the **signed perpendicular distance** from the vehicle to that line:

$$\text{CTE} = \frac{ax_c + by_c + c}{\sqrt{a^2 + b^2}} \qquad (5)$$

The sign indicates whether the vehicle is located to the left or to the right of the reference trajectory according to the direction of the normal vector $\hat{n}$. Consequently, this term captures how far the vehicle has drifted laterally from the intended path.

1.4.2 heading error

The heading error quantifies the angular misalignment between the vehicle's current yaw angle and the desired orientation of the reference trajectory. It reflects the difference between the vehicle heading and the tangent direction of the reference path, making it a fundamental

state variable in path-tracking control. Even when the lateral deviation is negligible, an incorrect heading can progressively drive the vehicle away from the desired trajectory. To determine this error, the reference heading angle ψr\psi_rψr is first computed from the vector connecting two successive waypoints $W_t(x_t, y_t)$ to $W_{t+1}(x_{t+1}, y_{t+1})$

$$\psi_r = arctan2(y_{t+1} - y_t\ , x_{t+1} - x_t) \quad (6)$$

Then, the heading error is defined as the difference between the vehicle's yaw angle ψ and the reference heading ψ_r:

$$e_\psi = \psi - \psi_r \qquad (7)$$

This quantity indicates how much the vehicle orientation deviates from the desired path direction. A small heading error implies proper alignment with the trajectory, whereas larger values require corrective steering actions to restore stability and tracking accuracy. In MPC formulations, the heading error is strongly coupled with the lateral tracking error through the vehicle velocity component perpendicular to the path, and it explicitly appears in the state-space prediction model used by the controller. For computational efficiency and real-time implementation, small-angle approximations are often employed during model linearization within the prediction horizon [15].

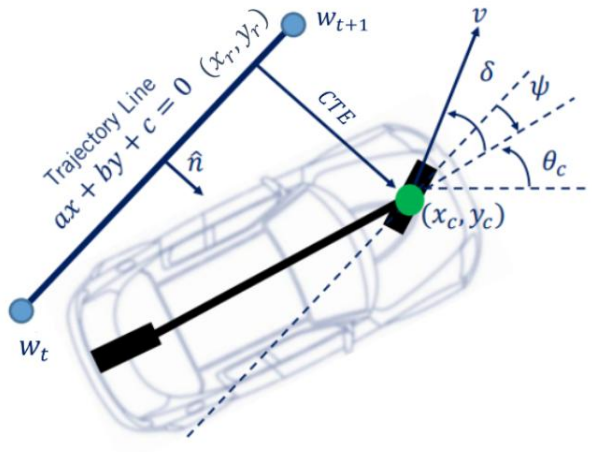


Figure1.4. Illustration of path tracking errors, CTE, and heading error (e_(ψ,t)) relative to successive waypoints

1.4.3 Speed Error

In addition to geometric tracking performance, the vehicle must regulate its longitudinal motion to follow the desired speed profile prescribed by the path planner. The velocity-tracking error is defined as:

$$e_v = v_r - v_c \qquad (8)$$

Where:

- v_r is the reference velocity generated by the curvature-based speed planner, and
- v_c is the measured vehicle velocity at time t.

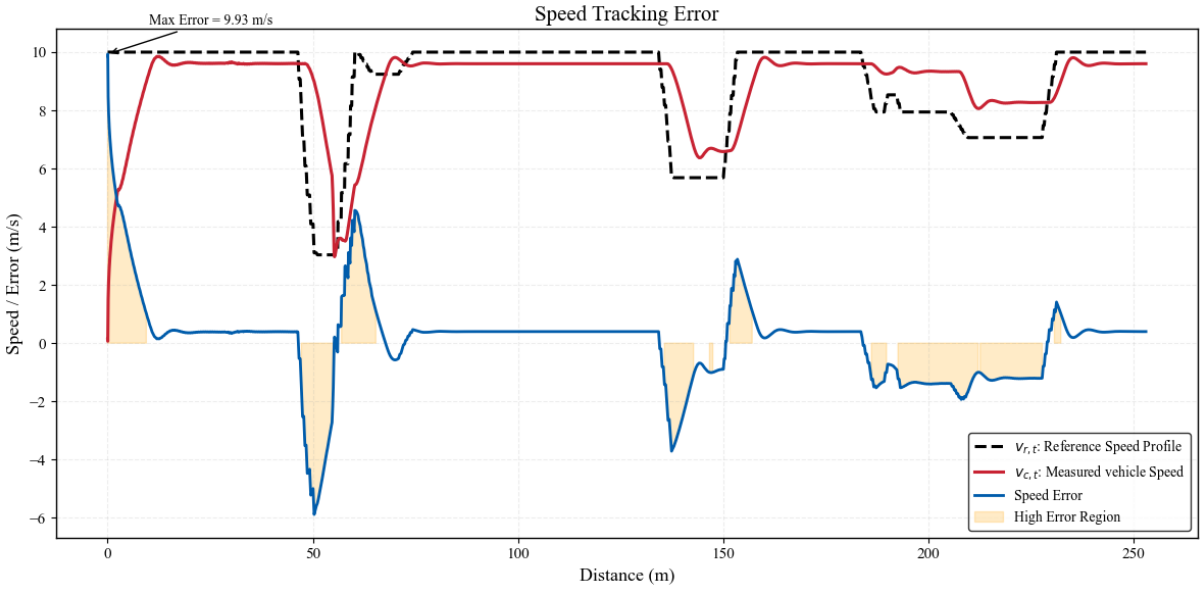


Figure 1.5 Speed-tracking performance along the test route. The dashed curve represents the reference speed profile, and the red curve shows the measured vehicle speed. The blue curve depicts the instantaneous speed-tracking error, and the shaded regions highlight intervals where the magnitude of this error exceeds 1 m/s. The maximum error observed along the route is also indicated.

1.5 Curvature-Aware Model Predictive Control (CAMPC)

Model Predictive Control (MPC) optimizes future control actions by predicting the vehicle's behavior over a finite horizon. Our proposed **Curvature-Aware MPC (CAMPC)** integrates curvature information directly into the cost function, enabling the controller to anticipate sharp turns and reduce lateral instability. The used cost function is as follows:

$$J_{CAMPC} = \sum_{k=0}^{N-1}(\mathrm{w}_{cte}e_{cte,k}^2 + w_\psi e_{\psi,k}^2 + w_v e_{v,k}^2 + w_\delta \delta_k^2 + w_a a_k^2 + w_b b_k^2 + w_\kappa \kappa_t^2 + w_{\Delta\delta}(\delta_k - \delta_{k-1})^2 + w_{a^+}(a_k - a_{k-1})^2) \quad (10)$$

where:

1. **State Tracking Costs**:

   $\mathrm{w}_{cte}(cte_k)^2$: Penalizes CTE (deviation from the reference path).

   $w_\psi\left(e_{\psi,k}\right)^2$: Penalizes heading angle error (deviation from the desired orientation).

2. **Velocity Tracking Cost**:

   $w_v\left(e_{v,k}\right)^2$: Penalizes deviation from the reference velocity v_{ref},where $e_{v,k} = v_k - v_{ref,k}$.

3. **Control Input Costs**:

   $w_\delta \delta_k^2$: Penalizes steering effort.

   $w_{\mathrm{a}}.\, a_k^2$: Penalizes throttle input (positive a_t).

   $w_b.\, b_k^2$: Penalizes braking input (negative a_t).

4. **curvature integration**:

   κ_t : curvature of the reference path at time t.

   w_{κ}: Weight penalizing high curvature zones.

5. **Control Smoothing Costs:**

   $w_{\Delta\delta}.\,(\delta_k - \delta_{k-1})^2$: Penalizes abrupt changes in steering angle.

   $w_{a^+}.\,(a_k - a_{k-1})^2$: Penalizes abrupt changes in throttle.

Where the state vector is:

$$X_t = [x_t \;\; y_t \;\; \psi_t \;\; v_t \;\; cte_t \;\; e_{\psi,t}\,]^T \qquad (11)$$

Where (x_t, y_t) is the global position, ψ_t the heading (yaw), v_t the longitudinal speed, cte_t, The signed lateral offset from the reference path and $e_{\psi,t}$ The heading difference between the vehicle and the path tangent.

The control inputs are explained as follows:

$$u_t = [a_t \;\; \delta_{f,t}]^T \qquad (12)$$

With a_t the longitudinal acceleration and $\delta_{f,t}$ is the steering angle of the front-wheel. The distances from the center of gravity to the front and rear axles are l_f and l_r, respectively [44].

1.6 Limitations of Fixed-Weight of MPC and proposed method

Model Predictive Control (MPC) is widely recognized as an effective control strategy for autonomous vehicle path tracking due to its ability to predict future vehicle behavior while explicitly handling system dynamics and actuator constraints. However, the performance of MPC strongly depends on the selection of its weighting parameters. In many practical implementations, these weights are determined manually through trial-and-error procedures

and remain fixed throughout vehicle operation. Although fixed-weight MPC can provide satisfactory performance under specific conditions, its effectiveness may decrease when the vehicle encounters different road geometries, speeds, and driving scenarios.

One of the main limitations of fixed-weight MPC is the difficulty of selecting a single set of parameters capable of providing optimal performance across all driving conditions. A controller tuned for straight roads may not provide the same level of accuracy on highly curved trajectories, while a configuration optimized for aggressive cornering may produce unnecessary control activity on straight segments. As a result, tracking performance, control smoothness, and vehicle stability can vary significantly depending on the driving environment.

Another challenge is the sensitivity of MPC performance to the selected weighting parameters. Small changes in these values can lead to noticeable differences in lateral tracking accuracy, heading regulation, and actuator behavior. Consequently, obtaining a suitable set of weights often requires extensive testing and expert knowledge. This tuning process becomes increasingly difficult as the complexity of the driving scenario grows, particularly in urban environments containing curves, intersections, and varying speed requirements.

To address these limitations, this work proposes a PSO-based optimization framework for determining the MPC weighting parameters. Instead of relying on manual tuning, Particle Swarm Optimization (PSO) is employed to automatically search for the weight vector that provides the best closed-loop performance. During the optimization process, candidate weight sets are evaluated through repeated simulations in the CARLA environment. For each candidate solution, the MPC controller performs trajectory tracking while a cost function measures tracking accuracy and control effort.

The optimization procedure is conducted entirely offline. Through iterative updates of particle positions and velocities, PSO explores the search space and progressively converges toward an optimal set of MPC weights. The resulting weight vector is then stored and used by the MPC controller during online operation. This approach eliminates the need for manual parameter tuning and allows the controller to operate with a set of weights selected according to a systematic optimization process.

By combining MPC with offline PSO-based weight optimization, the proposed method aims to improve trajectory-tracking accuracy, control smoothness, and overall vehicle stability while maintaining the real-time capabilities required for autonomous driving applications. The

formulation of the optimization problem and the integration of PSO with the MPC controller are presented in the following chapter.

Chapter Conclusion

In this chapter, we have laid the essential theoretical groundwork for the development of an advanced control system for autonomous vehicles. We began by defining autonomous vehicles and presenting the modular architecture of autonomous driving systems, highlighting the pivotal role of the control module as the lowest-level executor responsible for converting reference trajectories into precise actuator commands. The path tracking problem was then formally introduced, emphasizing its inherent challenges including nonholonomic constraints, parameter uncertainties, and real-time computational demands.

We subsequently explored vehicle modeling for control design, presenting the kinematic bicycle model as the chosen representation due to its balance between accuracy and computational efficiency. The curvature and speed profile design was discussed, demonstrating how geometric path characteristics can be exploited to generate safe and comfortable speed profiles through the fundamental relationship ay=v2κ*ay*=*v*2*κ*. The path tracking error formulation was then detailed, defining the cross-track error, heading error, and speed error as the key performance metrics for controller evaluation.

The chapter culminated in the introduction of the Curvature-Aware Model Predictive Control (CAMPC) approach, which explicitly incorporates road curvature information into the MPC cost function, enabling the controller to anticipate sharp turns and improve tracking performance. Finally, we identified the limitations of fixed-weight MPC and proposed a PSO-based optimization framework to systematically determine optimal weighting parameters, setting the stage for the detailed methodology presented in the following chapter.

Chapter 2

MPC Weights optimization by PSO

2.1 Problem Formulation

Model Predictive Control (MPC) computes the vehicle control inputs by solving an optimization problem over a finite prediction horizon at each control step. The quality of the generated control actions depends strongly on the weighting parameters used in the MPC cost function. These weights determine the relative importance of trajectory-tracking accuracy and control effort.

In most MPC implementations, the weighting parameters are selected manually and remain fixed during operation. Although this approach can provide satisfactory performance for a specific driving scenario, it may not be suitable for all road conditions. Autonomous vehicles operate in environments that include straight roads, sharp turns, intersections, and varying speed limits. A set of weights that performs well in one situation may lead to degraded performance in another. Consequently, selecting appropriate MPC weights is a challenging task.

The objective of this work is to determine a set of MPC weights that minimizes trajectory-tracking errors while maintaining smooth and stable control actions. Let $w = [w_1, w_2, \ldots\ldots, w_8]$ be the vector of MPC weighting parameters. For a given candidate weight vector (W), the MPC controller is executed in the CARLA simulation environment, and the resulting tracking performance is evaluated using a cost function (J). The optimization problem can be expressed as:

$$W^* = \arg\min_{w} J(w) \text{ subject to } \quad W_{min} \leq W \leq W_{max}$$

where J is the closed-loop cost computed from trajectory-tracking errors and control effort during a CARLA simulation run. The optimal weight vector W^* is obtained offline using Particle Swarm Optimization and subsequently used by the MPC controller during online operation.

The cost function combines trajectory-tracking errors and control effort. During each optimization iteration, the MPC controller uses the candidate weights generated by the Particle Swarm Optimization (PSO) algorithm. The CARLA environment provides the vehicle states and reference trajectory, while the cost evaluation module computes the corresponding performance index. This cost is then returned to the PSO algorithm, which updates the candidate solutions and continues the search process.

After convergence, the PSO algorithm produces the optimal weight vector (W^*). These optimized weights are stored and subsequently used by the MPC controller during the online

control phase, allowing the vehicle to perform real-time trajectory tracking without requiring additional optimization.

2.2 Proposed System Architecture

Before detailing each component, it is instructive to present the overall architecture that integrates them. Figure 2.1 illustrates the two-phase design: an offline optimization phase responsible for generating optimal weights, and an online control phase that uses the saved optimal weights for robust path tracking in real time.

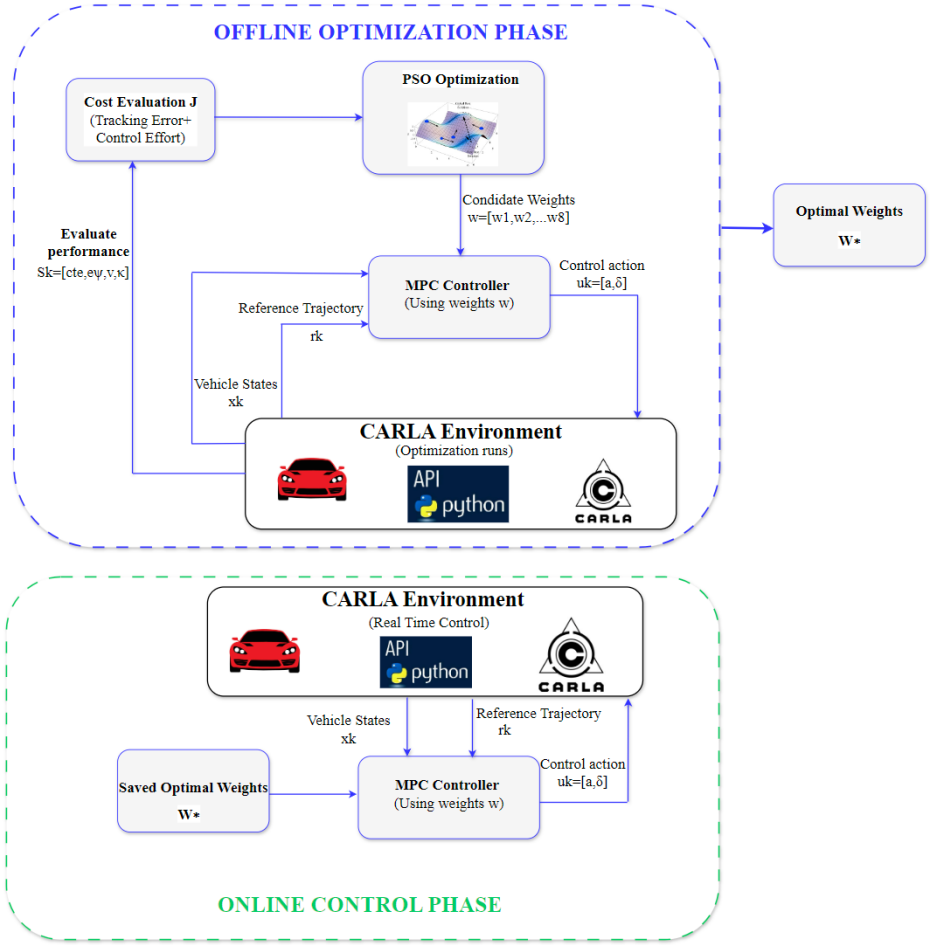


Figure 2.1 Offline - online architecture of the proposed PSO-tuned MPC controller. PSO optimizes the MPC weighting parameters in CARLA through repeated simulation-based evaluations of tracking performance and control effort. The resulting optimal weights W^* are subsequently used by the MPC controller for real-time vehicle trajectory tracking during online operation.

The proposed architecture consists of two main operational phases: an offline optimization phase and an online control phase. During the offline phase, a Particle Swarm Optimization (PSO) algorithm searches for the optimal weighting parameters of the Model Predictive Controller (MPC) by repeatedly evaluating candidate solutions in the CARLA simulation environment. For each candidate weight vector, the MPC controller performs trajectory tracking, while a cost evaluation module computes a performance index based on tracking error and control effort. Through iterative optimization, the PSO algorithm identifies the optimal weight vector (W^*). In the online phase, the optimized weights are stored and directly employed by the MPC controller for real-time vehicle control. The CARLA environment closes the control loop by providing vehicle states and reference trajectories, while the MPC controller generates the corresponding steering and acceleration commands.

Table 2.1 summarizes the role of each component in the proposed framework.

Phase	**Component**	**Function**
Offline Optimization	PSO Optimizer	Searches the MPC weight space and generates candidate weight vectors $w = [w_1, w_2, \dots., w_8]$.
Offline Optimization	Cost Evaluation J	Computes the objective function based on trajectory-tracking error and control effort. The resulting cost is fed back to the PSO optimizer.
Offline Optimization	MPC Controller	Applies each candidate weight vector generated by PSO and computes the control actions $u_k = [a, \delta]$ for trajectory tracking.

Offline Optimization	CARLA Environment	Executes simulation runs, provides vehicle states x_k and reference r_k trajectories, and evaluates the closed-loop performance associated with each candidate weight vector.
Offline Optimization Output	Optimal Weights W^*	Best weight vector obtained after PSO convergence and minimization of the cost function.
Online Control	Saved Optimal Weights W^*	Precomputed optimal MPC weights loaded for real-time operation.
Online Control	MPC Controller	Uses the optimized weight vector W^*, current vehicle states, and reference trajectory to compute real-time steering and acceleration commands.
Online Control	CARLA Environment	Receives control inputs, updates vehicle dynamics, and provides feedback states to close the control loop.

The key design principle is the separation of the computationally expensive weight search (which can take minutes per scenario offline) from the real-time control loop (which must operate at 10–20 Hz). The neural network acts as a compressed, differentiable representation of the PSO results, enabling the system to benefit from near-optimal weights without incurring the cost of solving an optimization problem at every timestep.

2.3 Particle Swarm Optimization

2.3.1 Definition and Operating Principle

Particle Swarm Optimization is a population-based stochastic optimization algorithm introduced by Kennedy and Eberhart in 1995 [24]. It is inspired by the collective motion of bird flocks and fish schools: individual agents called particles move through the search space by combining their own exploration history with information shared across the swarm. Unlike gradient-based methods, PSO requires no differentiability of the objective function and is therefore well suited to the black-box cost evaluation inherent in closed-loop MPC tuning.

A swarm of P particles is initialized with random positions $x_i \epsilon R^m$ and velocities $v_i \epsilon R^m$ within the admissible MPC weight space. At each iteration t, the velocity and position of particle i are updated according to:

$$v_i^{t+1} = \omega v_i^t + c_1 r_1 (p_i^t - x_i^t) + c_2 r_2 (g^t - x_i^t), \qquad (2.2)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \qquad (2.3)$$

In Eqs. (2.2) and (2.3), ω denotes the inertia weight, which controls the influence of the particle's previous velocity on its current movement. The parameters c1 and c2 are the cognitive and social acceleration coefficients, respectively, while r1 and r2 are independent random numbers uniformly distributed in the interval [0,1]. The term p_i represents the personal-best position previously found by particle i, whereas g^t denotes the global-best position identified by the entire swarm. The inertia weight ω is commonly decreased linearly throughout the optimization process to gradually shift the search from global exploration toward local exploitation [25].

The fitness of each particle is evaluated using the MPC closed-loop cost function J(W). After evaluating all particles, the personal-best and global-best solutions are updated, and the particle positions and velocities are recalculated according to Eqs. (2.2) and (2.3). This iterative process continues until a predefined convergence criterion is satisfied or the maximum number of iterations is reached.

2.3.2 Domains of Application

PSO has been applied across a broad range of engineering optimization problems due to its simplicity, low parameter count, and robustness to non-convex, multi-modal, and noisy objective landscapes. In control engineering, PSO has been used to tune PID gains [26],

optimize fuzzy membership functions, calibrate neural network weights, and, more recently, tune MPC cost matrices for robot navigation and autonomous driving. In the autonomous vehicle domain, fixed-weight MPC tuning via PSO has been demonstrated to outperform manual calibration on standard benchmarks, motivating its adoption here as the offline optimization engine [27].

2.3.3 PSO for MPC Weight Optimization

In the proposed framework, each PSO particle represents a candidate weight vector $W = [w_{cte}, w_{\psi}, w_{v}, w_{\delta}, w_{\mathrm{a}}, w_{b}, w_{\Delta\delta}, w_{a^{+}}]$ penalizing cross-track error, heading error, speed error, steering rate, acceleration rate, braking rate , abrupt changes in steering angle, abrupt changes in throttle angle, respectively.

The objective of the PSO algorithm is to determine the weight vector that minimizes the overall closed-loop tracking cost. The optimization problem is formulated as:

$$W^{*} = arg \min_{W \epsilon\, \Omega} J_{PSO}(W)$$

where Ω denotes the feasible search space of the MPC weights and $J_{PSO}(W)$ represents the closed-loop performance cost obtained from the CARLA simulation. The final output of the optimization process is the optimal weight vector (W^{*}), which is stored for use during online control phase.

For each particle, the corresponding weight vector is applied to the MPC controller and evaluated through a complete closed-loop simulation in the CARLA environment. The particle fitness is computed using the following objective function:

$$J_{PSO} = \alpha_1 RMSE(cte) + \alpha_2 RMSE(e_{\psi}) + \alpha_3 RMSE(e_{v}) + \alpha_4 \overline{|\Delta\delta|} + \alpha_5 \overline{|\Delta a|} + \alpha_6 \overline{|b|} \quad (2.4)$$

Where:

$$RMSE(cte) = \sqrt{\frac{1}{T}\sum_{k=1}^{T} cte_k^2}$$

$$RMSE(e_{\psi}) = \sqrt{\frac{1}{T}\sum_{k=1}^{T} e_{\psi,k}^2}$$

$$RMSE(e_v) = \sqrt{\frac{1}{T}\sum_{k=1}^{T} e_{v,k}^2}$$

And:

$$\overline{|\Delta\delta|} = \frac{1}{T}\sum_{k=1}^{T} |\delta_k - \delta_{k-1}|$$

$$\overline{|\Delta a|} = \frac{1}{T}\sum_{k=1}^{T} |a_k - a_{k-1}|$$

$$\overline{|b|} = \frac{1}{T}\sum_{k=1}^{T} |b_k|$$

Where T is the total number of controls timesteps (samples) used to evaluate one complete simulation run.

The summation is taken over all control timesteps within the evaluated driving scenario. For each particle, a complete closed-loop simulation is executed in the CARLA environment to assess the performance of the corresponding MPC weight vector. As a result, the optimization process requires significant computational effort, since every candidate solution must be evaluated through a full simulation run. This optimization is carried out offline, ensuring that it does not impact the real-time operation of the vehicle. The PSO algorithm continues updating particle positions and velocities until either the maximum number of iterations is reached or the improvement in the global best solution falls below a predefined convergence criterion. Once the optimization converges, the resulting optimal weight vector W^* is retained and used by the MPC controller during online vehicle operation.

2.4 Online Control Phase

2.4.1 Overall System Flow

After the offline optimization phase, the optimal MPC weight vector W^* obtained using PSO is stored and used during real-time vehicle operation. During each control timestep (k), the CARLA environment provides the current vehicle state.

$$X_t = [x_k \;\; y_k \;\; \psi_k \;\; v_k \;\; cte_k \;\; e_{\psi,k}]^T$$

together with the reference trajectory over the prediction horizon (). The MPC controller uses the optimized weight vector (W^*), the current vehicle state, and the reference trajectory R_k to formulate and solve a constrained finite-horizon optimization problem. The resulting control command $u_k = [a_k \ \delta_k]^T$ is then applied to the vehicle, where a_k and δ_k denote the acceleration and steering commands, respectively. The updated vehicle state is subsequently returned by the CARLA simulator, closing the control loop.

2.4.2 Online Control Algorithm

The online control procedure executed at each timestep (k) is summarized as follows:

Step 1: State acquisition: Obtain the current vehicle state (X_k) and the reference trajectory (R_{k+N}) from the CARLA environment.

Step 2: Weight loading.: Load the optimized MPC weight vector (W^*) obtained during the offline PSO optimization phase.

Step 3: MPC optimization: Solve the constrained optimal control problem

$$U^* = arg \min_U J_{MPC}(X_k, R_{k:k+1}, W^*)$$

subject to the vehicle dynamics and actuator constraints. This yields the optimal control sequence u^*_{k+N}

Step 4: Control application: Apply the first element of the optimal sequence, $u_k = u^*_k$, to the vehicle actuators (throttle a and steering angle δ).

Step 5: Loop update: Advance to timestep (k+1) and repeat the procedure.

The primary computational task during online operation is the solution of the MPC optimization problem at each control step. Since the weighting parameters have already been optimized offline, no additional parameter tuning or optimization is required during deployment. Consequently, the online controller retains the standard MPC structure while benefiting from weight values selected through the PSO optimization process. This separation between offline optimization and online control enables efficient real-time operation while maintaining improved trajectory-tracking performance.

2.5 Chapter Summary

This chapter presented the design of the proposed PSO-optimized MPC framework for autonomous vehicle trajectory tracking. The proposed approach addresses one of the main limitations of conventional MPC controllers, namely the difficulty of selecting appropriate

weighting parameters through manual tuning. Instead of relying on trial-and-error calibration, the proposed method employs Particle Swarm Optimization (PSO) to automatically determine a set of MPC weights that provides improved tracking performance and control smoothness.

The chapter first introduced the formulation of the MPC controller and the vehicle model used for trajectory tracking. The MPC cost function was then defined to account for cross-track error, heading error, speed tracking error, steering effort, throttle effort, braking effort, and control smoothness. Subsequently, the PSO algorithm was described as an offline optimization tool for searching the MPC weight space and identifying the weight vector that minimizes a closed-loop performance cost evaluated in the CARLA simulation environment.

The complete optimization framework was presented, including the particle representation, the PSO update equations, the objective function design, and the interaction between the PSO optimizer, the MPC controller, and the CARLA simulator. Through repeated simulation-based evaluations, the optimization process converges to an optimal weight vector that balances trajectory-tracking accuracy and control effort.

Finally, the online control phase was described. During vehicle operation, the MPC controller uses the optimized weight vector obtained during the offline optimization stage to generate steering, throttle, and braking commands in real time. Since the computationally intensive optimization process is performed offline, the online controller retains the standard MPC structure and satisfies the real-time requirements of autonomous driving applications.

The performance of the proposed PSO-optimized MPC framework is evaluated in Chapter 3 using several driving scenarios of varying complexity in the CARLA simulation environment.

Chapter 3

Simulation Results and Performance Analysis

3.1 Introduction

This chapter presents the simulation results obtained from the comparative evaluation of two control strategies: the Proportional-Integral-Derivative (PID) controller and the Curvature-Adaptive Model Predictive Controller (CAMPC). The experiments were conducted on a vehicle model navigating three predefined routes of increasing geometric complexity - a roundabout, a sophisticated multi-curve route, and a long curvy road - at a reference speed of 10 m/s throughout. Performance is assessed using four primary indicators: trajectory tracking accuracy, speed tracking fidelity, cross-track error (CTE), and orientation error ($e\psi$), alongside actuator usage metrics.

The chapter is organized as follows. Section 3.2 analyzes the PSO optimization convergence used to tune the CAMPC parameters. Sections 3.3, 3.4, and 3.5 present the qualitative six-panel diagnostic for each of the three routes. Section 3.6 consolidates all routes into a unified quantitative comparison table. Section 3.7 provides a cross-route discussion and Section 3.8 concludes the chapter.

3.2 PSO-Based Parameter Optimization

3.2.1 Optimization Objective

The CAMPC cost-function weights governing the relative importance of lateral error, heading error, speed deviation, and control effort were determined via Particle Swarm Optimization (PSO). Manual tuning of this weight set is impractical given the nonlinear coupling between vehicle dynamics and path geometry across heterogeneous road profiles. PSO was therefore used to minimize a composite cost criterion that aggregates tracking errors over a full representative scenario.

The algorithm evolves a swarm of candidate weight vectors over 50 trials, using inertia, cognitive, and social components to balance exploration and exploitation. The best cost observed by any particle at each trial is recorded to monitor convergence.

3.2.2 Convergence Results

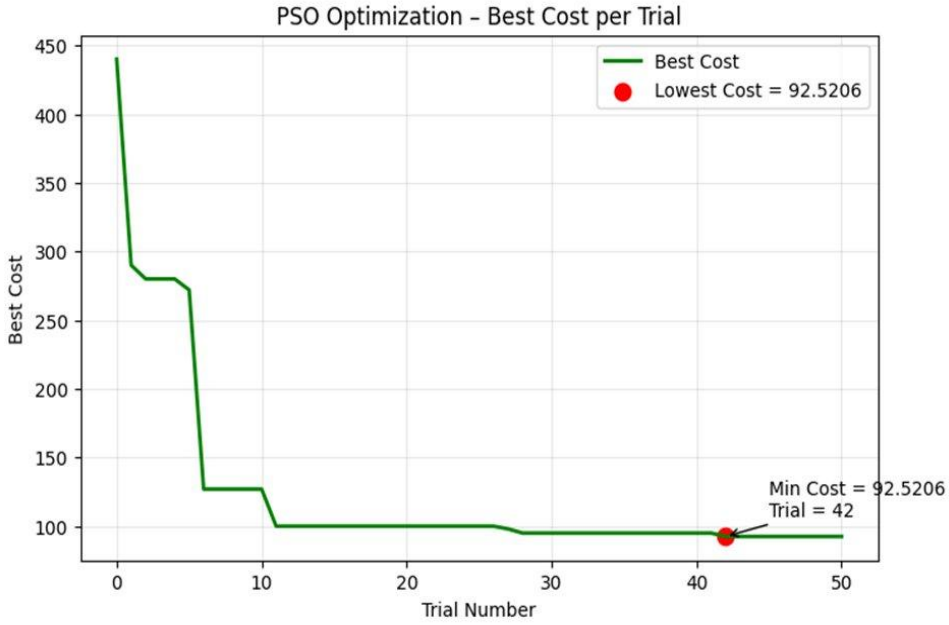


Figure 3.1 — PSO convergence: best cost per trial. The global minimum of 92.5206 is reached at Trial 42 (red dot).

Figure 3.1 shows three well-defined convergence phases. During trials 0–12, the cost drops sharply from approximately 441 to 100, reflecting rapid global exploration by the swarm. Between trials 12 and 28, a slower refinement phase reduces the cost to around 96 as particles concentrate around a promising subspace. From trial 28 onward, the cost stabilizes, with the definitive minimum of 92.5206 recorded at trial 42. The total cost reduction of 79% confirms that the optimized weight set substantially outperforms any random initialization and provides a reproducible, objective parameter configuration for all subsequent simulations.

3.3 Trajectory 1: Roundabout

3.3.1 Route Geometry

The first route replicates a roundabout circuit. The path is a closed circular loop of approximately 90 m radius, introducing sustained high curvature (κ reaching 0.10 m^{-1}) throughout most of the route. Unlike the pulse-curvature profiles of straight-plus-corner routes, the roundabout demands continuous lateral corrections over an extended arc, representing one of the most demanding tests for a lateral controller operating at fixed speed.

3.3.2 Six-Panel Diagnostic

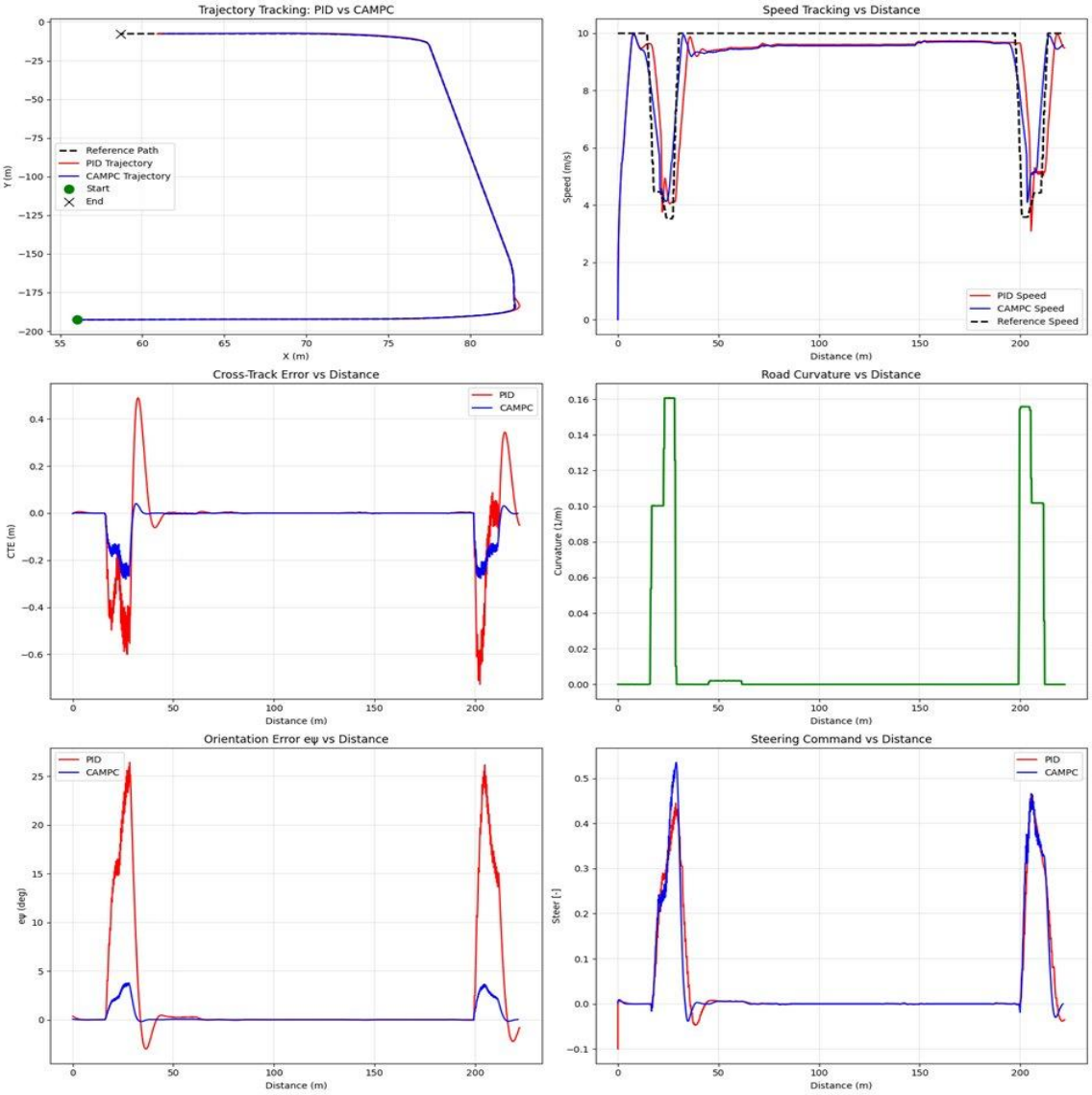


Figure 3.2 — Six-panel performance comparison for Trajectory 1 (Roundabout). PID in red, CAMPC in blue.

Trajectory Tracking

The X-Y plot (top-left) shows both controllers completing the full roundabout loop. The CAMPC trajectory (blue) closely follows the reference path (dashed black), while the PID trajectory (red) presents a visible outward drift during the tightest section of the circle, particularly on the lower-left arc. This outward bulge is consistent with the known overshoot behavior of reactive controllers on constant-curvature curves, where the accumulated lateral error is not corrected until the vehicle has already passed the apex.

Speed Tracking

The speed panel (top-right) reveals repeated deceleration events throughout the roundabout, driven by the continuous curvature. PID exhibits greater speed oscillation - with undershoots dropping to approximately 6 m/s - while CAMPC maintains speed within a narrower band. The CAMPC speed profile is noticeably smoother, avoiding the abrupt recoveries visible in the PID signal. Average speed error with CAMPC (0.803 m/s) is 29% lower than with PID (1.131 m/s).

Cross-Track Error

The CTE panel (middle-left) illustrates the lateral deviation over the full route. PID generates high-frequency oscillations reaching ±0.30 m throughout the curved portion, a signature of the proportional-derivative gains being insufficient to handle the sustained centripetal demand. CAMPC limits the CTE to a maximum of 0.176 m - a 51% reduction - and produces a notably cleaner transient profile. The residual CAMPC oscillations visible between distances 50 -130 m corresponds to the most curved section of the loop and are quickly damped.

Orientation Error

Heading error follows a pattern correlated with curvature. PID reaches a peak of 15.7° at the entry to the tight arc, indicating a significant misalignment between the vehicle heading and the path tangent. CAMPC constrains the heading error below 2.3° throughout, an improvement of nearly 85%. The flat CAMPC e_ψ curve confirms that the predictive horizon allows the controller to align the vehicle heading in advance of curvature transitions.

Steering Command and Actuator Usage

Both controllers produce similar peak steering magnitudes on this route. However, the PID steering signal contains high-frequency chattering throughout the arc (visible as rapid sign changes between distances 50-130 m), whereas CAMPC steering is smooth and continuous. This chattering behavior in PID increases actuator wear without contributing to lateral accuracy. The average braking command under CAMPC (0.009) is one-tenth of that under PID (0.091), indicating that the predictive speed regulation largely eliminates the need for emergency deceleration.

3.4 Trajectory 2 - Sophisticated Route

3.4.1 Route Geometry

The second route is a multi-segment path combining straight sections with a sequence of five distinct curved zones of varying curvature magnitude (κ ranging from 0.035 to 0.095 m^{-1}). The total route length is approximately 200 m. The irregular curvature profile - evident in the staircase-shaped Road Curvature vs Distance panel - makes this route particularly challenging, as neither the curvature amplitude nor the duration of each curve is uniform. A controller must therefore adapt its lateral strategy dynamically rather than responding to a single repeatable geometry.

3.4.2 Six-Panel Diagnostic

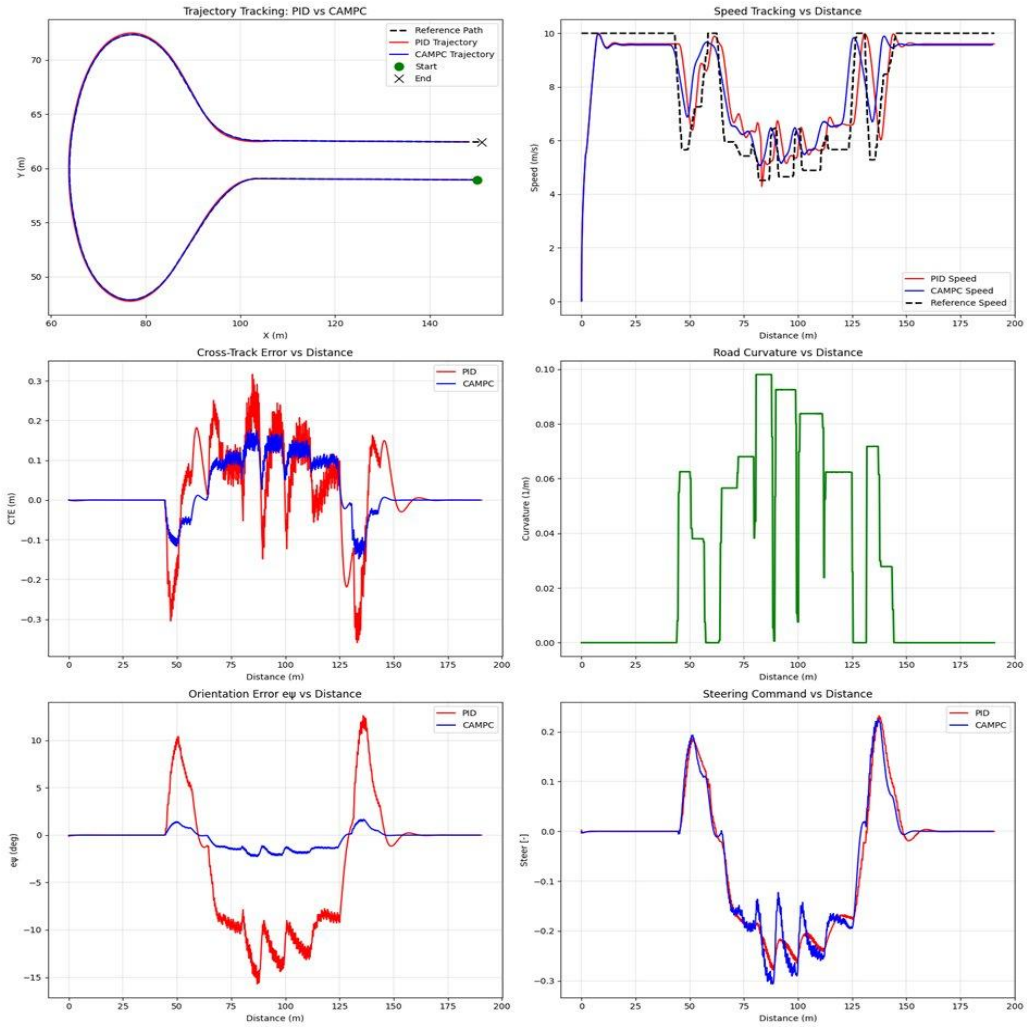


Figure 3.3 — Six-panel performance comparison for Trajectory 2 (Sophisticated Route). PID in red, CAMPC in blue.

Trajectory Tracking

The X–Y trajectory plot shows CAMPC maintaining close alignment with the reference path through all five curve segments. The PID trajectory deviates visibly at the third and fourth curves (approximately x = 90 -130 m), where the curvature peaks most sharply at 0.095 m^{-1}. At these segments, the PID path bulges outward by up to 0.30 m before recovering, while CAMPC maintains a near-perfect overlay with the reference. This confirms that CAMPC adapts its steering gain dynamically with curvature, whereas PID applies a fixed gain that proves inadequate at higher curvature levels.

Speed Tracking

The speed profile for this route reveals the most significant inter-controller difference observed across all three trajectories. PID speed drops repeatedly to approximately 6 m/s at each curve, with slow recovery on the short straight segments between them. CAMPC, by contrast, maintains speed above 9 m/s through all but the sharpest curves, where minor undershoots of approximately 0.5 m/s are observed. The average speed error for CAMPC (0.595 m/s) is 31% lower than for PID (0.860 m/s), and the maximum undershoot is comparable for both controllers (≈9.995 m/s max error), though this figure reflects the initial acceleration phase rather than steady-state behavior.

Cross-Track Error

The CTE panel shows the clearest performance separation of all three routes. PID generates oscillations in the range ±0.30 m across the entire curved portion, with a worst-case peak of −0.343 m near the tightest curve at distance 130 m. CAMPC holds CTE within ±0.15 m for most of the route, with maximum deviation of 0.283 m. The mean CTE under CAMPC (0.033 m) is 47% below that of PID (0.062 m), representing a consistent and uniform improvement across all curve segments rather than a selective advantage at specific zones.

Orientation Error

The heading error confirms the trajectory observations. PID peaks at 26.7° near the sharpest curve — an angular deviation large enough to represent a meaningful safety concern at 10 m/s. The corresponding CAMPC maximum is 3.8°, a reduction of approximately 86%. The CAMPC e_ψ curve is nearly flat throughout the straight sections and exhibits only brief, small-amplitude pulses at curvature transitions, rapidly returning to zero within 10–15 m.

Steering Command and Actuator Usage

On this route, both controllers generate similar peak steering amplitudes (approximately ±0.25), reflecting the moderate curvature levels. The key distinction lies in the frequency content: PID produces a noisy, oscillatory steering signal throughout the curved zones, while CAMPC output is smooth and contains no high-frequency components. Average absolute steering under CAMPC (0.060) is marginally lower than PID (0.065), and average braking is reduced from 0.052 to 0.015 a 71% reduction further confirming the efficiency of joint longitudinal-lateral optimization.

3.5 Trajectory 3: Curvy Route

3.5.1 Route Geometry

The third route is the longest and geometrically richest of the three, spanning approximately 600 m and incorporating eight distinct curvature events with peak values reaching $\kappa = 0.16$ m^{-1} — the highest curvature tested in this study. The route includes both gradual transitions and abrupt curvature steps, as seen in the Road Curvature vs Distance panel. The extended route length ensures that sustained performance is evaluated, not merely transient behavior at isolated corners.

3.5.2 Six-Panel Diagnostic

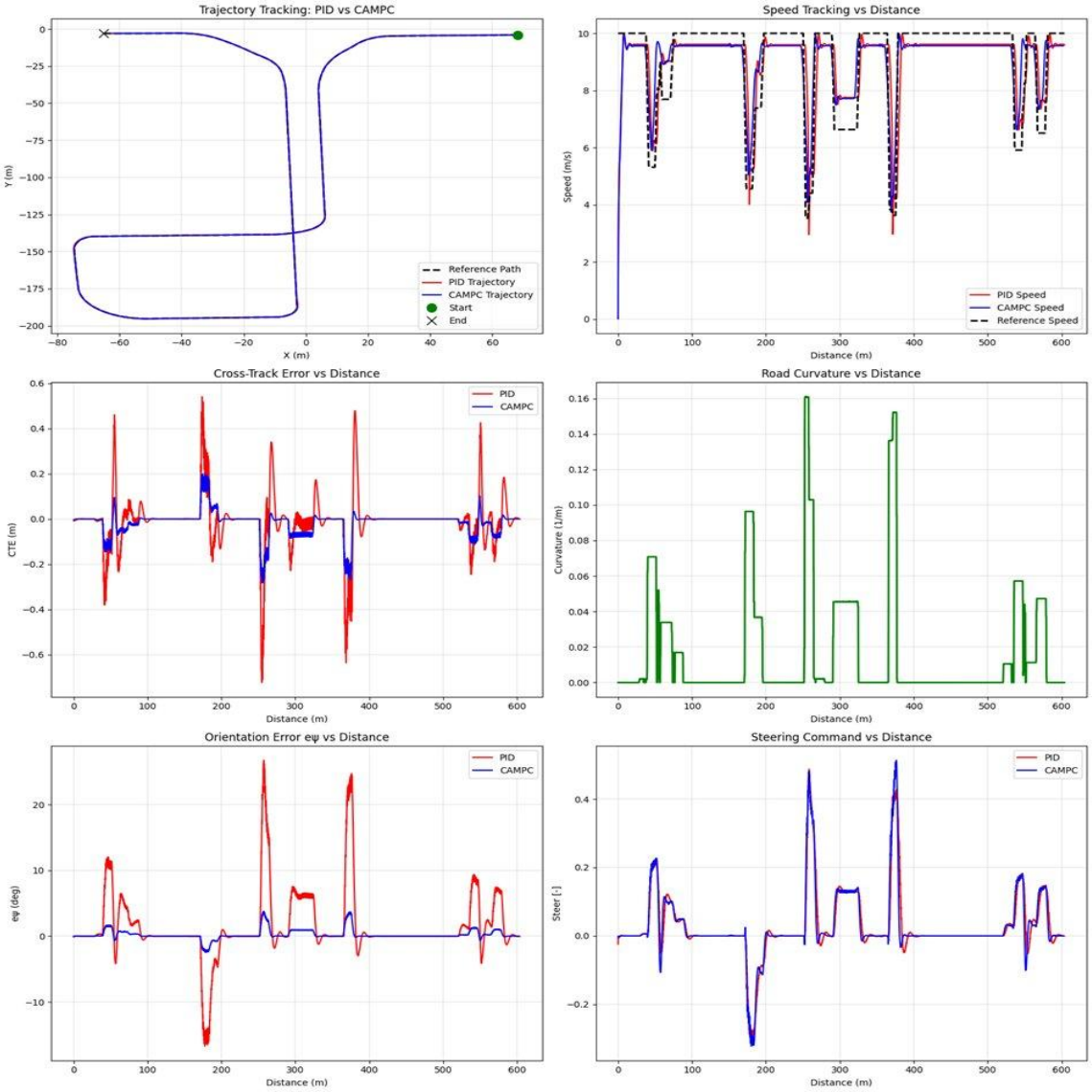


Figure 3.4 — Six-panel performance comparison for Trajectory 3 (Curvy Route). PID in red, CAMPC in blue.

Trajectory Tracking

The X–Y trajectory plot for this route spans a large coordinate range (X: −80 to +65 m, Y: −200 to 0 m), covering a complex geometric path that includes both wide sweeping curves and tight corners. CAMPC (blue) maintains continuous overlap with the reference path across all eight curvature events. PID (red) tracks the reference adequately on the straight segments and gentler curves but exhibits measurable overshoot at the high-curvature corners near distances 240–300 m, where κ peaks at 0.16 m^{-1}. At these points, the PID trajectory deviates

laterally before converging back, a pattern consistent with the reactive gain structure being unable to pre-compensate for abrupt curvature changes.

Speed Tracking

Speed regulation on this route presents a more uniform pattern than on the sophisticated route, due to the wider spacing between curvature events. Both controllers maintain speed near 10 m/s on the long straight segments. At curvature peaks, PID again exhibits larger undershoots approximately 3.5 m/s at the most demanding corners while CAMPC limits deceleration to below 1 m/s. The average speed error for CAMPC (0.848 m/s) is 22% below that of PID (1.089 m/s). The speed regulation advantage is consistent and repeatable across all eight curvature events.

Cross-Track Error

The CTE panel for Trajectory 3 shows PID generating spikes of up to ±0.50 m at the high-curvature corners near distances 240, 280, and 350 m, with secondary oscillations persisting for 20–30 m after each corner exit. CAMPC maintains the CTE within ±0.10 m for the majority of the route, with isolated peaks not exceeding 0.280 m at the sharpest corners. The average CTE for CAMPC (0.034 m) is 55% lower than for PID (0.075 m). The sustained low-error profile of CAMPC over a 600 m route demonstrates that its performance advantage is not a transient effect but a structurally consistent property of the predictive control law.

Orientation Error

The heading error panel confirms the pattern established on the previous two routes. PID peaks at 26.4° at the most demanding corner, while CAMPC reaches a maximum of only 3.8°. The ratio of improvement (approximately 7×) is consistent with Trajectory 2, despite the higher peak curvature, suggesting that the CAMPC horizon length is well-calibrated for the curvature scales present in these routes. Between corners, CAMPC heading error returns rapidly to zero on each straight segment, while PID shows residual oscillations that only attenuate gradually.

Steering Command and Actuator Usage

On this route, CAMPC applies noticeably higher peak steering commands at the sharpest corners (0.45 vs. 0.43 for PID), consistent with the proactive pre-steering strategy. This early, larger steering input reduces the peak lateral error at the cost of a slightly higher actuator demand at corner entry. Post-corner, CAMPC steering returns to zero more rapidly than PID, avoiding the residual corrections that characterize the reactive controller. Average braking is reduced from 0.056 (PID) to 0.022 (CAMPC) — a 61% reduction — and average throttle is reduced from 0.454 to 0.441, confirming overall energy efficiency gains with the predictive strategy.

3.6 Consolidated Quantitative Performance Summary

Table 3.1 consolidates all nine-performance metrics across the three routes and both controllers. The values are extracted directly from the simulation logs and represent full-route statistics rather than segment-level measurements.

Metric	Trajectory 1: Roundabout		Trajectory 2: Sophisticated Route		Trajectory 3: Curvy Route	
	PID	CAMPC	PID	CAMPC	PID	CAMPC
Avg CTE (m)	0.0752	0.054	0.0624	0.033	0.0752	0.034
Max CTE (m)	0.358	0.176	0.723	0.283	0.728	0.280
Avg eψ (deg)	5.327	0.756	3.319	0.450	3.684	0.481
Max eψ (deg)	15.713	2.281	26.722	3.755	26.432	3.775
Avg Speed Err (m/s)	1.131	0.803	0.860	0.595	1.089	0.848
Max Speed Err (m/s)	9.996	9.996	9.995	9.995	9.997	9.997
Avg |Steer| [-]	0.105	0.099	0.065	0.060	0.070	0.061
Avg Throttle [-]	0.447	0.421	0.436	0.420	0.454	0.441
Avg Brake [-]	0.091	0.009	0.052	0.015	0.056	0.022

Table 3.1 — Full quantitative performance comparison: PID vs CAMPC across three routes and nine metrics.

3.6.1 Cross-Metric Analysis

Examining the table by row reveals consistent trends across all three trajectories. Max CTE is reduced by 51-61% across routes, and Max eψ is reduced by 85-86%, with the largest absolute improvements on Trajectories 2 and 3 where the higher curvature amplifies the PID error. Average CTE improvements range from 28% (Trajectory 1) to 55% (Trajectory 3), and average eψ improvements range from 86-88%.

Speed metrics show that average speed error is reduced by 22–31% depending on the route, while the maximum speed error which reflects the initial acceleration transient is near-identical for both controllers, confirming that the start-up behavior is not influenced by the control architecture. Actuator metrics reveal a consistent pattern: CAMPC uses less average braking (reductions of 71-90%), less average throttle (reductions of 3-6%), and similar or

slightly lower average steering, confirming that the predictive optimization produces globally more efficient control actions.

3.7 Discussion

3.7.1 Predictive Horizon vs Reactive Gain

The results across all three routes demonstrate a fundamental structural advantage of model predictive control over fixed-gain PID for curved-road tracking. The PID controller operates on the current error state: it steers in proportion to the present cross-track and heading errors, with a derivative term that reacts to the rate of change of error. This architecture has no mechanism to account for path geometry ahead of the vehicle, meaning that the lateral demand of an upcoming curve only registers in the control law after the vehicle has already begun to deviate.

CAMPC, by contrast, minimizes a predicted cost over a finite horizon that explicitly includes the curvature of the path ahead. As the vehicle approaches a corner, the optimizer identifies the need for pre-steering and begins applying lateral inputs before the cross-track error develops. The practical result is that the peak lateral error at corner entry is dramatically reduced, and the post-corner recovery is faster and smoother. This advantage scales with curvature magnitude: on the gentle curves of the roundabout, the improvement is moderate; on the high-curvature corners of Trajectory 3, it is decisive.

3.7.2 Joint Speed and Lateral Optimization

A secondary but important advantage of CAMPC is the joint optimization of speed and lateral control within a single cost function. In the PID framework, the longitudinal controller operates independently of the lateral controller. When the lateral error grows as occurs at every corner entry the independent speed controller continues to target 10 m/s, leading to unnecessary speed during the highest-demand phase of the turn. CAMPC implicitly coordinates these two degrees of freedom: when the predicted lateral error at a corner is high, the optimizer naturally prefers a lower speed setpoint to reduce the required centripetal force, resulting in the smoother, less aggressive decelerations observed across all routes.

3.7.3 Actuator Efficiency and Practical Implications

The significant reduction in braking demand under CAMPC between 61% and 90% depending on the route has practical implications for brake wear, passenger comfort, and energy consumption. The reduction in throttle demand (3 to 6%) is more modest but consistent. These efficiency gains arise not from any explicit energy term in the cost function but as a natural byproduct of smoother, better-coordinated control actions. In a real deployment, these savings translate directly into longer component service intervals and improved range for electric vehicles.

3.7.4 Limitations and Open Questions

Despite the consistent advantages demonstrated, several limitations must be acknowledged. First, all simulations were conducted under idealized conditions: no sensor noise, no actuator lag, and perfect model knowledge. The CAMPC performance is sensitive to the accuracy of the vehicle model used within the optimizer; model mismatch due to tire nonlinearities or load transfer at high speeds could degrade tracking accuracy. Second, the computational cost of solving the MPC optimization at each time step was not evaluated in this study and represents an important practical constraint for embedded real-time deployment. Third, the PSO tuning was performed on a single representative route; the generalizability of the resulting weight set to routes with significantly different geometries warrants further investigation.

3.8 Conclusion

This chapter presented a systematic simulation-based comparison of PID and CAMPC controllers across three road geometries of increasing complexity. The results are consistent, reproducible, and unambiguous in their direction. The following conclusions can be drawn:

1. **The PSO-optimized CAMPC weight set is robust across diverse geometries.** Tuned on a representative scenario with a cost of 92.52, the resulting parameter set delivered consistent performance improvements across all three routes without per-route retuning, confirming the generalization capability of the PSO-based calibration approach.

2. **CAMPC substantially reduces lateral tracking errors.** Maximum cross-track error was reduced by 51-61% and maximum heading error by 85-86% across all three routes. These improvements hold at all curvature levels tested, from the moderate $\kappa = 0.10$ m^{-1} of the roundabout to the demanding $\kappa = 0.16$ m^{-1} peaks of the curvy route.

3. Speed regulation and lateral control benefit from joint optimization. Average speed error under CAMPC is 22-31% lower than under PID, and the speed profile is markedly smoother at every curvature transition. The coordinated longitudinal-lateral strategy avoids the abrupt decelerations that characterize the independent PID architecture.

4. Actuator usage is more efficient under CAMPC. Average brake demand is reduced by 61 - 90% and throttle by 3-6%, while steering actuation remains comparable. The smooth, proactive steering profile of CAMPC eliminates the high-frequency chattering seen in PID, reducing actuator wear and improving ride quality.

5. The performance gap widens with curvature severity. The relative advantage of CAMPC over PID increases as route curvature increases, confirming that the predictive horizon is the determinant factor. For low-curvature or straight-road driving, both controllers are broadly equivalent; the CAMPC advantage is most critical precisely in the scenarios where control quality matters most high-speed cornering.

These findings provide a strong and well-supported basis for adopting CAMPC as the preferred control architecture for autonomous vehicle path tracking in environments where road geometry is varied and curvature demands are non-trivial. The next chapter will address the real-time implementation aspects of CAMPC, including computational complexity, embedded hardware requirements, and horizon length sensitivity analysis.

General Conclusion

General conclusion

This research falls within the field of advanced autonomous vehicle control, proposing an adaptive approach to improve trajectory tracking performance. Faced with the limitations of conventional fixed-gain controllers, particularly their inability to adapt to abrupt changes in road curvature and driving conditions, we developed a predictive control architecture based on a model, incorporating dynamic adaptation of the cost function weights. The main objective was to reconcile lateral tracking accuracy, vehicle stability, and passenger comfort, while respecting the real-time constraints inherent to embedded systems.

In the first chapter, we laid the theoretical groundwork for the autonomous vehicle's functional chain, detailing the various layers from perception to control. Particular attention was paid to bicycle kinematic modeling, chosen for its balance between dynamic accuracy and low computational complexity. We also introduced the CAMPC (Curvature-Aware Model Predictive Control) controller, whose originality lies in the explicit integration of road curvature into the cost function, thus enabling the anticipation of sharp turns. This chapter highlighted the sensitivity of MPC performance to manual weight adjustments, which underscores the need for a systematic optimization method.

The second chapter presented the main contribution of this thesis: a hybrid method combining particle swarm optimization (PSO) and neural networks for adaptive tuning of the MPC weights. The approach unfolds in two phases. First, an offline phase where PSO explores the weight space for each driving condition encountered in the CARLA simulation environment, generating a database of pairs (vehicle state, optimal weights). Then, a dense neural network is trained on this data to learn the nonlinear correspondence between the current vehicle state (lateral error, orientation error, speed, curvature, and its derivative) and the optimal MPC weights. This clever design shifts the computational burden from offline to ultrafast online inference, compatible with the real-time constraints of autonomous driving.

The third chapter experimentally validated the proposed approach through a series of simulations on three courses with contrasting geometries: a roundabout, a complex winding road, and a long, highly curved road. The quantitative results demonstrate a clear superiority of the adaptive CAMPC compared to a conventional PID controller and a fixed-weight MPC. Specifically, the maximum lateral tracking error (CTE) was reduced by 51% to 61%, the maximum orientation error by 85% to 86%, and the average speed error by 22% to 31%, depending on the course. Furthermore, the approach enabled a significant reduction in

braking effort (up to 90%) and remarkable smoothing of steering inputs, limiting the chattering observed with PID controllers. These results confirm that integrating curvature and adapting weights in line improves safety, comfort, and energy efficiency.

In conclusion, this thesis makes a relevant contribution to the field of autonomous vehicle control by demonstrating that adaptive weight tuning of a mass-propulsion system (MPC), guided by curvature and learned by a neural network, is not only feasible in real time but also significantly more efficient than classical approaches. However, some limitations remain: the work was conducted in an ideal simulation environment (CARLA) without sensor noise or actuator latency, and the robustness of the approach to variations in vehicle parameters (mass, adhesion) was not tested. For future research, it would be interesting to extend this method to more complex dynamic models, to integrate a continuous online adaptation loop for the neural network (incremental learning), and to validate the approach on a scaled-down vehicle or in a more realistic simulator including external perturbations.

Bibliographical references

Références bibliographiques

[1] SAE International, *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles (SAE J3016)*, 2021

[2] Liu, S., Li, L., Tang, J., Wu, S., & Gaudiot, J.-L. (2020). *Creating Autonomous Vehicle Systems* (2nd ed.). Morgan & Claypool Publishers.

[3] Stano, P., Montanaro, U., Tavernini, D., Tufo, M., Fiengo, G., Novella, L., & Sorniotti, A. (2023). Model predictive path tracking control for automated road vehicles: A review. *Annual Reviews in Control*, 55, 194–236.

[4] Stano, P., et al. (2023). Model predictive path tracking control for automated road vehicles: A review. *Annual Reviews in Control*, 55, 194–236.

[5] Rajamani, R. (2012). *Vehicle Dynamics and Control* (2nd ed.). Springer.

[6] Stano, P., et al. (2023). Model predictive path tracking control for automated road vehicles: A review. *Annual Reviews in Control*, 55, 194–236.

[7] Rajamani, R. (2012). *Vehicle Dynamics and Control* (2nd ed.). Springer.

[8] Polack, P., Altché, F., d'Andréa-Novel, B., & de La Fortelle, A. (2017). The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles. *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 812–818.

[9] Kong, J., Pfeiffer, M., Schildbach, G., & Borrelli, F. (2015). Kinematic and dynamic vehicle models for autonomous driving control design. *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 1094–1099. (arXiv preprint available at)

[10] Nan, J., et al. (2024). Model predictive control for autonomous vehicle path tracking through optimized kinematics. *Results in Engineering*, 24, 103123.

[11] Kong, J., et al. (2015). Kinematic and dynamic vehicle models for autonomous driving control design. *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*.

[12] Stano, P., et al. (2023). Model predictive path tracking control for automated road vehicles: A review. *Annual Reviews in Control*, 55, 194–236.

[13] P. Falcone, F. Borrelli, J. Asgari, H. Tseng and D. Hrovat, "Predictive Active Steering Control for Autonomous Vehicle Systems," *Proc. American Control Conference*, 2007, pp. 4153–4158.

[14] M. Werling, J. Ziegler, S. Kammel and L. Gröll, "Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenet Frame," *Proc. IEEE Intelligent Vehicles Symposium*, 2010, pp. 416–423.

[15] J. Ziegler and C. Stiller, “Spatiotemporal State Lattices for Fast Trajectory Planning in Dynamic On-Road Driving Scenarios,” *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 1879–1884.

[13] Duan, X., et al. (2021). Implementing trajectory tracking control algorithm for autonomous vehicle. *IEEE Conference Proceedings*.

[14] Nan, J., et al. (2024). Model predictive control for autonomous vehicle path tracking through optimized kinematics. *Results in Engineering*, 24, 103123.

[15] Rajamani, R. (2012). *Vehicle Dynamics and Control* (2nd ed.). Springer.

[16] Stano, P., et al. (2023). Model predictive path tracking control for automated road vehicles: A review. *Annual Reviews in Control*, 55, 194–236.

[17] Stano, P., et al. (2023). Model predictive path tracking control for automated road vehicles: A review. *Annual Reviews in Control*, 55, 194–236.

[18] Chen, Z., et al. (2024). Prediction horizon-varying model predictive control (MPC) for autonomous vehicle control. *Electronics*, 13(8), 1442.

[19] Nan, J., et al. (2024). Model predictive control for autonomous vehicle path tracking through optimized kinematics. *Results in Engineering*, 24, 103123.

[20] Vu, T. M., et al. (2021). Model predictive control for autonomous driving vehicles. *Electronics*, 10(21), 2593.

[21] Domina, Á., & Tihanyi, V. (2023). Model predictive controller approach for automated vehicle’s path tracking. *Sensors*, 23(15), 6862.

[22] Stano, P., et al. (2023). Model predictive path tracking control for automated road vehicles: A review. *Annual Reviews in Control*, 55, 194–236.

[23] Nan, J., et al. (2024). Model predictive control for autonomous vehicle path tracking through optimized kinematics. *Results in Engineering*, 24, 103123.

[24] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proc. IEEE Int. Conf. Neural Networks, vol. 4, Perth, Australia, 1995, pp. 1942–1948.

[25] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in Proc. IEEE World Congr. Comput. Intell., Anchorage, AK, USA, 1998, pp. 69–73.

[26] A. I. Selvakumar and K. Thanushkodi, "A new particle swarm optimization solution to nonconvex economic dispatch problems," IEEE Trans. Power Syst., vol. 22, no. 1, pp. 42–51, Feb. 2007.

[27] H. Peng, T. Hao, J. Fu, and J. Hu, "Adaptive model predictive control for autonomous vehicles using deep reinforcement learning," IEEE Trans. Veh. Technol., vol. 70, no. 4, pp. 3339–3349, Apr. 2021.

[28] M. Zanon, S. Gros, and M. Diehl, "A tracking MPC formulation that is locally equivalent to economic MPC," J. Process Control, vol. 45, pp. 30–42, 2016.

[29] Yurtsever, E., Lambert, J., Carballo, A., & Takeda, K. (2020). A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access*, 8, 58443–58469.