



Article

Research on the Performance of Vehicle Lateral Control Algorithm Based on Vehicle Speed Variation

Weihai Zhang, Jinbo Wang * and Tongjia Pang

Department of Transportation, School of Automotive Engineering, Shandong Jiaotong University, No. 5001 Haitang Road, Changqing University Science Park, Jinan 250300, China

* Correspondence: wangjinbo@sdjtu.edu.cn

Abstract: Analyzing the performance characteristics and applicable environments of intelligent vehicle lateral control algorithms, this paper establishes the Model Predictive Control (MPC), Pure Pursuit (PP), and Linear Quadratic Regulator (LQR) algorithms and uses 2022b MATLAB software to simulate the lateral error, heading error, and algorithm execution time at speeds of 3 m/s, 7 m/s, and 10 m/s. Urban low-speed scenarios (3 m/s) require high-precision control (such as obstacle avoidance), while high-speed scenarios (10 m/s) require strong stability. Existing research mostly focuses on a single speed and lacks a quantitative comparison across multiple operating conditions. Although MPC has high accuracy, its time consumption fluctuates greatly. LQR has strong real-time performance but a wide range of heading errors. PP has poor low-speed performance but controllable high-speed time consumption growth. It is necessary to define the applicable scenarios of each algorithm through quantitative data. In response to the lack of multi-speed domain quantitative comparison in existing research, this paper conducts multi-condition simulations using MPC, PP, and LQR algorithms and finds that at a low speed of 3 m/s, the peak lateral error of PP (0.45 m) is 55% and 156% higher than MPC (0.29 m) and LQR (0.176 m), respectively. At a speed of 10 m/s, the lateral error standard deviation of MPC (0.08 m) is reduced by 68% compared to PP (0.25 m). In terms of algorithm time consumption, LQR maintains full-speed domain stability (0.11–0.44 ms), while PP time increases by 95% with speed from 3 m/s to 10 m/s. The results show that in terms of lateral error, the MPC and LQR algorithms perform more stably, while the PP algorithm has a larger error at low speeds. Regarding heading error, all algorithms have a relatively large error range, but the MPC and LQR algorithms perform slightly better than the PP algorithm at high speeds. In terms of algorithm execution time, the LQR algorithm has the shortest and most stable execution time, the MPC algorithm has a relatively longer execution time, and the PP algorithm's execution time varies at different speeds. Through this simulation, if high control accuracy and stability are pursued, the MPC or LQR algorithm can be considered; if real-time performance and computational efficiency are more important, the PP algorithm can be considered.

Keywords: model predictive control algorithm; pure pursuit algorithm; linear quadratic regulator algorithm; lateral error; heading error; algorithm execution time



Academic Editor: Kan Akatsu

Received: 28 February 2025

Revised: 29 March 2025

Accepted: 30 April 2025

Published: 4 May 2025

Citation: Zhang, W.; Wang, J.; Pang, T. Research on the Performance of Vehicle Lateral Control Algorithm Based on Vehicle Speed Variation. *World Electr. Veh. J.* **2025**, *16*, 259. <https://doi.org/10.3390/wevj16050259>

Copyright: © 2025 by the authors. Published by MDPI on behalf of the World Electric Vehicle Association. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the rapid development of autonomous driving technology, vehicle lateral control algorithms have become the core element of autonomous driving systems, and their performance and stability are directly related to the safety and reliability of autonomous driving

vehicles. The main goal is to ensure that the vehicle can travel smoothly and accurately along the predetermined trajectory, whether on urban roads, highways, or facing complex and changing driving environments. The global market size of autonomous vehicles is expected to jump from USD 76 billion in 2021 to USD 400 billion in 2030, and the penetration rate of technologies above the L3 level will reach 15%, covering core scenarios such as urban transportation, logistics, and transportation. With 33 million autonomous vehicles expected to be put into use in 2035, traditional control strategies have made it difficult to meet the demands of a dynamic environment. Research shows that 34% of traffic accidents are caused by lateral deviation, with a 2.7-fold increase in collision risk when the lateral error exceeds 0.3 m in curved scenes, and a heading angle deviation of 0.2 rad leading to 41% of lane-keeping failures. Therefore, developing high-precision lateral control algorithms based on MPC, LQR, etc., to achieve functions such as lane keeping and emergency obstacle avoidance has become a key technological direction for improving the safety of autonomous driving. This study compared the performance of MPC, PP, and LQR algorithms and found that in the 10 m/s high-speed scenario, the lateral error range of the PP algorithm ($-0.78\text{ m}\sim 0.056\text{ m}$) may exceed the safety threshold ($\pm 0.3\text{ m}$), while MPC algorithm can stabilize the error within the range of $-0.07\text{ m}\sim 0.26\text{ m}$, significantly reducing the risk of lateral collision. Furthermore, the LQR algorithm ensures controllable errors ($-0.073\text{ m}\sim 0.266\text{ m}$) while taking only $0.11\sim 0.44\text{ ms}$ to compute, meeting the real-time requirements for emergency obstacle avoidance (usually requiring $<50\text{ ms}$). These results indicate that high-precision lateral control algorithms are the key technological path to solving the problem of autonomous driving accidents. Lateral control is not only crucial to ensuring driving safety but also key to improving vehicle driving stability. In recent years, many scholars have been committed to closely combining the design of intelligent driving vehicle systems with vehicle dynamic models, and deeply exploring and optimizing lateral motion control strategies.

The lateral control algorithms of vehicles mainly include the model predictive control algorithm (MPC), pure tracking algorithm (PP), Linear Quadratic Regulator algorithm (LQR), and Stanley algorithm (Stanley), etc. Zhu Xingjian et al. [1] designed a weight-based coefficient adaptive MPC system to verify its reliability and effectiveness in the practical applications of unmanned vehicles. Zhiheng et al. [2] added dynamic constraints of centroid side deflection angle, acceleration, and tire side deflection angle based on the traditional model prediction and tracking algorithm, and established a trajectory planning layer on the trajectory tracking control layer. The results show that the model prediction control algorithm with dynamic constraints has the advantage of tracking under harsh operating conditions and has good adaptability to external environmental interference and internal factors of the vehicle. Sun Yinjian [3] uses the 6-degree-of-freedom vehicle dynamic model and the magic formula tire model as the basis, and applies the model predictive control theory, combined with the soft constraint of the lateral deflection angle, and designs a trajectory tracking controller based on the linear time-varying model prediction control algorithm, and passes the previous. The wheel actively turns to control the vehicle and realizes trajectory tracking while ensuring the stability of the vehicle. V Changoski [4] and others analyze future self-driving cars' line-controlled steering systems by implementing model predictive control controllers, which will optimize the vehicle's expected behavior. Zhu Zhongwen et al. [5] studied the nonlinear Model Predictive Control (MPC) trajectory tracking control strategy to address the issues of insufficient trajectory tracking accuracy and poor stability of autonomous vehicles under complex paths. The control effect was improved, and it showed good robustness to different vehicle speeds. Xie X.Y. et al. [6], considering the problem of long calculation time for MPC solution and low online real-time performance, proposed a model predictive trajectory tracking control method based on

variable step size of the control time domain. By changing the step size of the control time domain, the calculation time for the solution was reduced. Kayacan E et al. [7], taking into account the disturbances during the motion process, combined the linear model predictive controller with feedforward and robust control actions to ensure control accuracy and stability. Liu Yifan et al. [8] proposed a pure tracking control with dynamic adjustment of pre-sight distance, using the speed and lateral deviation as the input of the fuzzy controller, and the pre-sight distance as the output, eliminating the fixed pre-sight distance brought by ordinary pure tracking models. Disadvantages effectively improve control accuracy. Liu Weidong [9] et al. proposed a path-tracking controller based on improved pure tracking, established a road pre-aim model combining planning paths and planning speed information, and further conducted closed-loop testing and verification on the real vehicle platform. The experimental results show that the improved pure tracking algorithm has good tracking accuracy and steering smoothness. Meng Wang et al. [10] designed a path-tracking control strategy. Based on improved pure tracking algorithms, the strategy flexibly adjusts the forward distance based on data of vehicle speed and road curvature radius to ensure that in various road environments, all are applicable below. This approach significantly reduces the problem of large path-tracking errors that occur during curve driving. Chen Ridong [11] took the Pure Pursuit (PP) algorithm as the research object. Innovatively, the steering angle compensation of the low-speed PP controller was solved based on a two-point preview. Moreover, a high-speed PP controller based on the preview function and steering angle constraints, as well as a vehicle speed controller based on the track curvature, were proposed. Ni et al. [12] constructed the robot's motion model through the Lagrangian function method and further optimized the LQR controller using a rolling optimization strategy based on lateral tracking error, thereby enhancing its tracking accuracy. On the other hand, Trapanese et al. [13] incorporated feedforward control into traditional LQR controllers, a move that significantly improved the accuracy of lateral tracking. Gao Linlin et al. [14] proposed an improved LQR lateral motion control method with feedforward control to achieve the improvement of the LQR controller and improve the adaptability and control accuracy of the controller. In order to improve the vehicle's lateral stability and tracking accuracy of smart cars under high curvature and high-speed operating conditions, Chen Liang et al. [15] proposed an intelligent automobile linear secondary regulator (LQR) based on the optimal front wheel lateral bias force. The lateral control method of path tracking shows that, while reducing the path tracking error, the control method can also effectively improve the lateral stability of the vehicle. Lu Yang et al. [16] addressed the issues of low control accuracy and poor stability in the trajectory tracking of autonomous driving using the Linear Quadratic Regulator (LQR). These issues stem from the difficulty in selecting the weight coefficient matrices Q and R . To solve this, they proposed an LQR lateral trajectory tracking control method based on an improved genetic optimization algorithm. Zhang Yajuan et al. [17] proposed an Adaptive Linear Quadratic Regulator (ALQR) controller. Under high- and low-road surface adhesion coefficient conditions, the tracking accuracy of the ALQR controller is on average 65.867% higher than that of the LQR controller. In particular, on low-adhesion road surfaces, the LQR controller neglects the significant decline in tracking performance caused by changes in tire stiffness. The proposed ALQR controller can calculate the optimal front-wheel steering angle in real time by using the real-time updated tire cornering stiffness, thus ensuring the vehicle's tracking accuracy and stability.

Although there have been numerous studies exploring the performance of lateral control algorithms for autonomous driving, such as MPC, PP, and LQR, most literature focuses on single-speed scenarios (such as fixed low or high speeds) and lacks systematic comparisons across multiple-speed domains (3 m/s to 10 m/s), resulting in a lack of

universal basis for algorithm selection. Existing research focuses more on lateral errors and neglects the comprehensive trade-off between heading errors and computational time, which makes it difficult to meet the requirements of precision and real-time collaborative optimization in practical engineering; The comparison of algorithm performance relies heavily on qualitative descriptions (such as ‘good’ or ‘poor’), lacking quantitative indicators such as standard deviation, peak error, and time consumption growth rate, which cannot provide clear boundary data.

2. Vehicle Kinematics Model

To implement control of the vehicle’s motion, the first step is to build a data-based vehicle motion model. Among them, the second-degree-of-freedom prediction model is widely used as a common and concise vehicle model. We use the tool of the two-degree-of-freedom prediction model to build a lateral motion control model of the vehicle, which is a kinematic model. The following Figure 1 shows the two-degree-of-freedom prediction models. In addition to the degree-of-freedom prediction model, dynamic models, data-driven models, geometric path-tracking models, and sliding mode control models can also be used to construct lateral motion control, each with unique applicable scenarios and limitations. The dynamic model considers more degrees of freedom, such as longitudinal/lateral velocity, yaw rate, tire sideslip angle, suspension dynamics, etc., and introduces a tire force model (such as the Pacejka magic formula) to describe nonlinear characteristics. It has higher accuracy in high-speed and high-dynamic scenarios (such as emergency obstacle avoidance and track driving) and can capture the behavior of vehicles under extreme conditions. However, it has high computational complexity, requires powerful processor support, and parameter calibration is difficult (such as tire stiffness and friction coefficient). Data-driven models use neural networks, reinforcement learning, and other methods to directly learn dynamic characteristics from vehicle driving data, without the need for explicit modeling. They are adaptable to complex environments such as icy and snowy roads and tire wear, and can handle nonlinearity and uncertainty that traditional models find difficult to describe. They do not rely too much on a large amount of high-quality data, have limited generalization ability, and may be affected by model size in real time. The geometric path-tracking model directly calculates steering instructions based on geometric relationships (such as forward distance and curvature), such as the Pure Pursuit algorithm, which is computationally efficient and suitable for low-speed scenarios with gentle path curvature (such as warehouse AGVs and park shuttle buses). The sliding mode control model achieves robust control by designing a sliding surface, which has a strong fault tolerance for model uncertainty and external disturbances. It can still maintain stable tracking under changes in tire–road friction coefficient or crosswind interference. However, the control output may have high-frequency oscillations, which require the use of filters to increase system complexity.

The degree-of-freedom prediction model is widely used in vehicle lateral control, mainly for the following reasons:

(1) High computational efficiency

The degree-of-freedom prediction model significantly reduces model complexity by simplifying vehicle dynamics (usually only considering key degrees of freedom such as lateral displacement and yaw angle).

(2) Balance accuracy and practicality

Although simplification of degrees of freedom may sacrifice some dynamic details, it can fully describe the vehicle’s motion characteristics in low-speed to medium-speed scenarios, such as urban roads.

(3) Mature theoretical foundation

The degree-of-freedom prediction model is based on rigid body kinematics equations, with a concise mathematical form and easy linearization. In addition, its parameters have clear physical meanings, making it easy to calibrate and verify, and reducing the debugging cost of actual deployment.

(4) Adapt to mainstream control methods

MPC, LQR, and other control algorithms rely on predictive models for state estimation and optimization. Due to its low dimensional nature, the degree-of-freedom prediction model can be directly embedded into the control framework to avoid unsolvable optimization problems or computational delays caused by high model complexity.

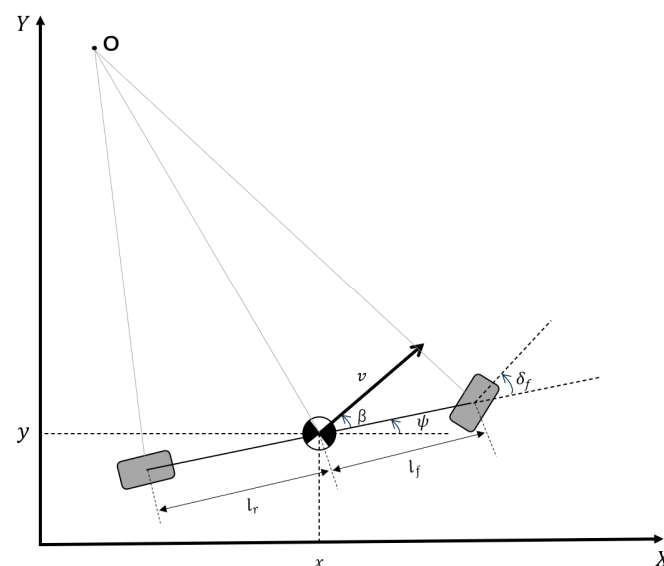


Figure 1. Two-degree-of-freedom prediction model.

Where v : The speed of the center of mass of the vehicle, which represents the actual speed of the vehicle during driving.

ψ : yaw angle (the angle between the longitudinal axis of the vehicle and the X-axis), representing the direction of the vehicle.

β : lateral deflection angle of the centroid (the angle between the longitudinal axis and the speed of the vehicle).

δ_f : Front wheel angle.

When establishing a vehicle kinematic model, the assumptions made are [18]:

1. On the premise of ignoring the movement of the vehicle in the vertical direction (i.e., the Z-axis), our explanation of the vehicle focuses on a two-dimensional plane scene;
2. Under the assumption, that the rotation angle of the two front wheels or two rear wheels of the vehicle is consistent with the rotation speed, and the steering angle is very small, we can use the same properties of the two front wheels or rear wheels. Consider them as a whole, i.e., representing it with a front or rear wheel. In this way, the original four-wheel vehicle model can be reasonably simplified into a two-wheel vehicle model;
3. At low speeds, we ignore the lateral force exerted by the tire, based on the premise that the front wheel angle is the same as the front wheel speed, and the rear wheel is the same. This assumption is valid because at low-speed driving, the lateral force generated by the tire is very small and almost negligible, so it can be approximately considered that the front wheel angle is consistent with the wheel speed direction;

4. If the vehicle's driving speed changes are gradual, meaning the acceleration remains at a low level, then the load transfer between the front and rear axles of the vehicle can be ignored;
5. Assuming that the entire vehicle and the suspension system are rigid, meaning the wheels do not have slip movement, a yaw angle can be used to represent the actual movement direction of the vehicle.

Figure 2 is the kinematic model obtained by satisfying the assumptions according to Figure 1. (X_f, Y_f) are the front wheels of the car, (X_r, Y_r) are the rear wheels of the car, δ is the steering angle of the car, l is the wheelbase, and R is the radius of the trajectory. We usually use two coordinate systems: one is the inertial coordinate system, which is the reference system that the inertial navigation system depends on; the other is the vehicle body coordinate system, which is mainly used to characterize the relative motion state of the vehicle itself. In Figure 2, the inertial coordinate system is XOY , and the vehicle body coordinate system is based on the V_r direction as the front direction. φ is the angle between the two coordinate systems and is negative clockwise.

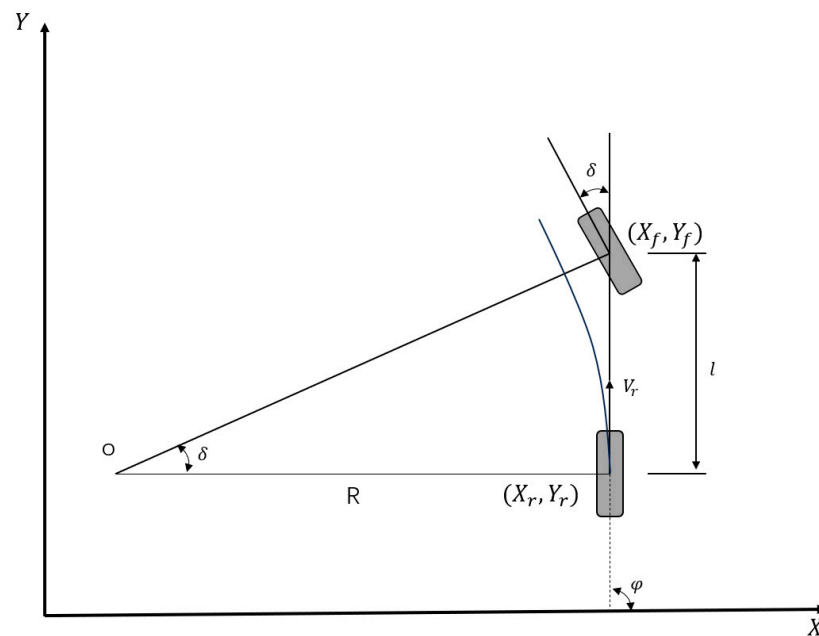


Figure 2. Vehicle motion model.

According to Figure 2, the kinematic model can be derived as

$$\begin{bmatrix} \dot{X}_r \\ \dot{Y}_r \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos\varphi \\ \sin\varphi \\ \tan\delta/l \end{bmatrix} V_r \quad (1)$$

The commonly used controls in autonomous driving are vehicle speed V_r and angular velocity ω_r , so the kinematic model we use for vehicle speed and angular velocity is:

$$\begin{bmatrix} \dot{X}_r \\ \dot{Y}_r \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos\varphi \\ \sin\varphi \\ 0 \end{bmatrix} V_r + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega_r \quad (2)$$

The state quantity ι is $[X_r, Y_r, \varphi]^T$, the control quantity K is $[V_r, \delta]$, and the angular velocity is $\omega_r = \frac{v_r}{l} \tan \delta$.

3. Horizontal Control Algorithm

3.1. Model Predictive Control (MPC)

Model Predictive Control (MPC) has a unique theoretical basis and core advantages, as it achieves multi-objective and multi-constrained control through rolling time-domain optimization. In terms of multivariable control and constraint handling, it can explicitly handle physical constraints such as steering angle restrictions to avoid control instructions exceeding limits, making it suitable for complex path-tracking scenarios. In terms of dynamic adaptability, the prediction model can be updated online to adapt to changes in vehicle load and external disturbances (such as crosswinds). There is also the ability for long-term optimization, which can predict future multi-step trajectories, optimize steering smoothness, and reduce passenger discomfort caused by sudden turns. In terms of applicable scenarios, the literature [19] indicates that it is particularly suitable for highway lane keeping and emergency obstacle avoidance in high-precision, high-speed scenarios; combined with the magic formula tire model, it can effectively handle the nonlinear characteristics of tire lateral force and improve control accuracy. MPC uses known models, the current state of the system, and future control amounts to predict the future output of the system, and it rolls solutions to the constraint optimization problem to achieve the purpose of control. There are three key steps in model prediction control, namely: prediction model, rolling optimization, and feedback correction.

As shown in Figure 3, the reference trajectory is represented by curve 1, where k on the horizontal axis represents the current point in time. The controller will calculate the output curve 2 in the prediction time domain $[k, k + p]$ based on the current system status, control inputs, and prediction model. By solving an optimization problem, we can obtain curve 4 of the predicted control quantity from the starting point of the longitudinal axis. Then, the first value in the control quantity curve is selected as the actual executed control quantity for the prediction of the $k + 1$ time. This process is repeated continuously to achieve continuous and effective control of the controlled object. According to the kinematic modeling analysis of the aforementioned vehicle, the system exhibits nonlinear features and it is difficult to directly draw its solution. In order to improve the feasibility of computer solutions, the first step is to linearize the system; that is, implement a linear transformation. Based on the state quantity $\tilde{i} = [\tilde{X}_r \quad \tilde{Y}_r \quad \tilde{\varphi}]^T$, the control quantity $\tilde{\kappa} = [\tilde{V}_r \quad \tilde{\delta}]$. We use the method of Taylor series expansion to pair (1) calculate and obtain the linearized equation:

$$\dot{\tilde{i}} = A\tilde{x} + B\tilde{\kappa} \quad (3)$$

$$\text{where } A = \begin{bmatrix} 0 & 0 & -V_r \sin \theta_r \\ 0 & 0 & V_r \cos \theta_r \\ 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} \cos \theta_r & 0 \\ \sin \theta_r & 0 \\ \tan \delta / l & V_r / l \cos^2 \delta_r \end{bmatrix}.$$

Discretize (3):

$$\tilde{i}(k+1) = A_{k,t} \tilde{i}(k) + B_{k,t} \tilde{\kappa}(k) \quad (4)$$

where $A_{k,t} = \begin{bmatrix} 1 & 0 & -V_r T \sin \varphi_r \\ 0 & 1 & V_r T \cos \varphi_r \\ 0 & 0 & 1 \end{bmatrix}$, $B_{k,t} = \begin{bmatrix} T \cos \varphi_r & 0 \\ T \sin \varphi_r & 0 \\ T \tan \delta_r / l & T V_r / l \cos^2 \delta_{f_r} \end{bmatrix}$, $\tilde{i}(k)$ is the state quantity at the current sampling moment, $\tilde{\kappa}(k)$ is the current control quantity, $\tilde{i}(k+1)$ is the state quantity at the next sampling moment, and T is the sampling period.

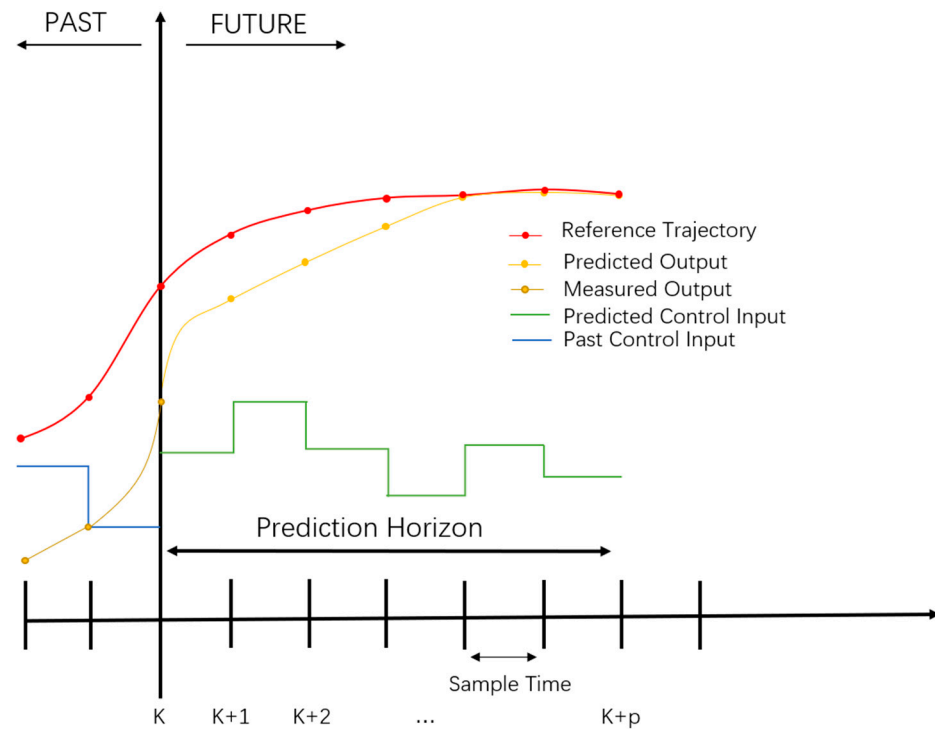


Figure 3. MPC schematic diagram.

In order to convert the control quantity into the control quantity increment, a new state vector $\xi(k|t) = [x(k) \quad u(k-1)]^T$ is constructed, and we can obtain:

$$\xi(k+1) = \tilde{A}_k \xi(k) + \tilde{B}_k \Delta u(k) \quad (5)$$

$$\eta(k) = \tilde{C}_k \xi(k) \quad (6)$$

where $\tilde{A}_k = \begin{bmatrix} A_k & B_k \\ 0 & I_m \end{bmatrix}$, $\tilde{B}_k = \begin{bmatrix} B_k \\ I_m \end{bmatrix}$, $\tilde{C}_k = [C_k \quad 0]$ and I_m represents the one-dimensional and two-dimensional identity matrices. According to (5) and (6) state prediction, we can obtain

$$\begin{aligned} \xi(k+1) &= \tilde{A}_k \xi(k) + \tilde{B}_k \Delta u(k) \\ \xi(k+2) &= \tilde{A}_k \xi(k+1) + \tilde{B}_k \Delta u(k+1) = \tilde{A}_k^2 \xi(k) + \tilde{A}_k \tilde{B}_k \Delta u(k) + \tilde{B}_k \Delta u(k+1) \\ \xi(k+N_p) &= \tilde{A}_k^{N_p} \xi(k) + \tilde{A}_k^{N_p-1} \tilde{B}_k \Delta u(k) + \cdots + \tilde{B}_k \Delta u(k+N_p-1) \end{aligned} \quad (7)$$

According to (7), the system output of the new state space equation can be calculated as follows:

$$\begin{aligned} \eta(k+1) &= \tilde{C}_k \xi(k+1) = \tilde{C}_k \tilde{A}_k \xi(k) + \tilde{C}_k \tilde{B}_k \Delta u(k) \\ \eta(k+2) &= \tilde{C}_k \tilde{A}_k^2 \xi(k) + \tilde{C}_k \tilde{A}_k \tilde{B}_k \Delta u(k) + \tilde{C}_k \tilde{B}_k \Delta u(k+1) \\ \eta(k+N_p) &= \tilde{C}_k \tilde{A}_k^{N_p} \xi(k) + \tilde{C}_k \tilde{A}_k^{N_p-1} \tilde{B}_k \Delta u(k) + \cdots + \tilde{C}_k \tilde{B}_k \Delta u(k+N_p-1) \end{aligned} \quad (8)$$

Express the output of the system's future moments in the form of a matrix:

$$Y = \psi \xi(k) + \Theta \Delta U \quad (9)$$

$$\text{where } Y = \begin{bmatrix} \eta(k+1) \\ \eta(k+2) \\ \vdots \\ \eta(k+N_c) \\ \vdots \\ \eta(K+N_p) \end{bmatrix}, \psi = \begin{bmatrix} \tilde{C}_k \tilde{A}_k \\ \tilde{C}_k \tilde{A}_k^2 \\ \vdots \\ \tilde{C}_k \tilde{A}_k^{N_c} \\ \vdots \\ \tilde{C}_k \tilde{A}_k^{N_p} \end{bmatrix},$$

$$\Theta = \begin{bmatrix} \tilde{C}_k \tilde{B}_k & 0 & 0 & 0 \\ \tilde{C}_k \tilde{A}_k \tilde{B}_k & \tilde{C}_k \tilde{A}_k^2 \tilde{B}_k & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{C}_k \tilde{A}_k^{N_c-1} \tilde{B}_k & \tilde{C}_k \tilde{A}_k^{N_c-2} \tilde{B}_k & \cdots & \tilde{C}_k \tilde{A}_k \tilde{B}_k \\ \tilde{C}_k \tilde{A}_k^{N_c} \tilde{B}_k & \tilde{C}_k \tilde{A}_k^{N_c-1} \tilde{B}_k & \cdots & \tilde{C}_k \tilde{A}_k^2 \tilde{B}_k \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{C}_k \tilde{A}_k^{N_p-1} \tilde{B}_k & \tilde{C}_k \tilde{A}_k^{N_p-2} \tilde{B}_k & \cdots & \tilde{C}_k \tilde{A}_k^{N_p-N_c-1} \tilde{B}_k \end{bmatrix}, \Delta U = \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_c) \end{bmatrix}$$

Define the reference output vector $Y_{ref}(k) = [\eta_{ref}(k+1), \dots, \eta_{ref}(k+N_p)]^T$, let $E = \psi \tilde{\zeta}(k)$. Substitute (9) into available

$$J = \Delta U^T (\Theta^T Q_Q \Theta + R_R) \Delta U + 2(E^T Q_Q \Theta - Y_{ref}^T Q_Q \Theta) \Delta U + \boxed{E^T Q_Q E + Y_{ref}^T Q_Q Y_{ref} - 2Y_{ref}^T Q_Q E} \quad (10)$$

where $Q_Q = \mathbf{I}_{N_p} \otimes \mathbf{Q}$, $R_R = \mathbf{I}_{N_p} \otimes \mathbf{R}$, $\otimes$ represents the Kroneck product. The part inside the rectangular box in the formula is a constant and can be ignored during optimization and solution, so the performance evaluation function can be written as

$$J = \Delta U^T (\Theta^T Q_Q \Theta + R_R) \Delta U + 2(E^T Q_Q \Theta - Y_{ref}^T Q_Q \Theta) \Delta U \quad (11)$$

Let $H = \Theta^T Q_Q \Theta + R_R$, $g = \Theta^T Q_Q (E - Y_{ref})$. Then, Formula (11) can be rewritten as (12)

$$J = 2 \left(\frac{1}{2} \Delta U^T H \Delta U + g^T \Delta U \right) \quad (12)$$

In optimization solution, Equation (12) is equivalent to

$$J = \frac{1}{2} \Delta U^T H \Delta U + g^T \Delta U \quad (13)$$

considering that there is the following relationship between the control amount and the control increment:

$$\mathbf{u}(k+i) = \mathbf{u}(k+i-1) + \Delta \mathbf{u}(k) \quad (14)$$

Before optimization, the variables in the constraints need to be unified, and the increments of the control quantity and the control quantity can be converted into the following form:

$$U_{\min} \leq A_k \Delta U + U_t \leq U_{\max} \quad (15)$$

$$U_{\min} \leq A_k \Delta U + U_t \leq U_{\max} \quad (16)$$

$$\Delta U_{\min} \leq \Delta U_t \leq \Delta U_{\max} \quad (17)$$

where $U_t = \mathbf{1}_N \otimes \mathbf{u}(k-1)$, $\mathbf{1}_N$ is the column vector with row number N_c , $\mathbf{u}(k-1)$ is the actual control amount at the previous moment, and $U_{\min}$ and $U_{\max}$ are, respectively, the

control in the control time domain. The minimum and maximum value sets of quantities, $\Delta U_{\min}$ and $\Delta U_{\max}$ are the minimum and maximum values sets of control increments in the control time domain, respectively.

$$A_k = \underbrace{\begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix}}_{N_c \times N_c}$$

At this point, the optimization problem-solving method of model prediction control can be transformed into a standard quadratic planning problem:

$$\begin{aligned} \min_{\Delta U} & \frac{1}{2} \Delta U^T H \Delta U + g^T \Delta U \\ & \Delta U_{\min} \leq \Delta U \leq \Delta U_{\max} \\ & U_{\min} \leq A_k \Delta U + U_t \leq U_{\max} \end{aligned} \quad (18)$$

For the above formula, you can use the QP solver *quadprog* provided by MATLAB to obtain a series of control input increments in the control time domain:

$$\Delta \mathbf{U}^* = [\Delta \mathbf{u}^*(k), \Delta \mathbf{u}^*(k+1), \dots, \Delta \mathbf{u}^*(N_c-1)]^T \quad (19)$$

According to the basic principles of model predictive control, the first element $\Delta \mathbf{u}^*(k)$ in the control sequence is used as the actual control input increment on the system. That is,

$$\mathbf{u}(k) = \mathbf{u}(k-1) + \Delta \mathbf{u}^*(k) \quad (20)$$

The system executes this amount of control until the next moment. At a new moment, the system re-predicts the output of the next time domain based on the state information and obtains a new control increment sequence through the optimization process. This cycle is repeated until the system completes the control process.

3.2. Pure Tracking Algorithm (Pure Pursuit, PP)

The PP algorithm selects the target point at the pre-sight distance in front of the vehicle, allowing it to travel along the predetermined trajectory. It uses geometric relationships to calculate the steering angle required for the vehicle to point to the target point and then adjust the vehicle direction according to the calculated steering angle, keeping it on course toward the target point. The pure tracking algorithm (PP) calculates steering instructions based on geometric forward viewpoints and has a unique theoretical basis and core advantages. It is computationally efficient, with a complexity of $O(1)$ and low average time consumption, making it very suitable for low computing platforms such as warehouse AGVs. Moreover, it does not require a dynamic model and only relies on the geometric relationship of the path, which makes debugging simple. In terms of applicable scenarios, PP is suitable for low-speed flexible control, such as park logistics vehicles performing well when the path curvature is gentle. At the same time, it can also serve as a redundant control strategy, acting as a backup algorithm for MPC or LQR, to reduce system load in simple scenarios. However, PP also has limitations, as its high-speed performance is poor [20].

In Figure 4, l_d is the distance from the current rear axle center A of the car to the target point C, and α is the angular deviation.

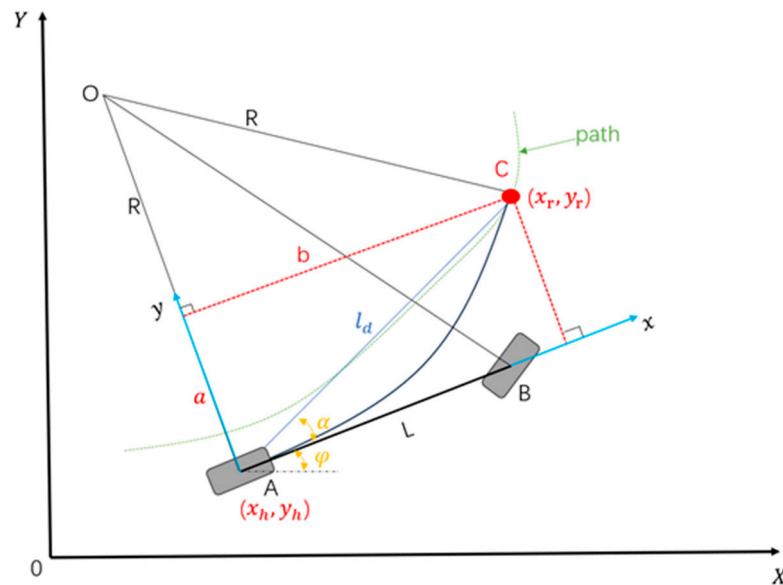


Figure 4. PP geometric model diagram.

Figure 4 can be obtained according to the sine theorem:

$$\frac{l_d}{\sin(2\alpha)} = \frac{R}{\sin(\frac{\pi}{2} - \alpha)}$$

Simplify:

$$k = \frac{2\sin\alpha}{l_d} \quad (21)$$

where k is the arc of the vehicle traveling in the Ackermann motion model, then the expression of the front wheel angle δ is:

$$\delta = \arctan(kL)$$

L is the vehicle wheelbase. In summary, there is

$$\delta(t) = \arctan \frac{2L\sin(\alpha)}{l_d} \quad (22)$$

3.3. Linear Quadratic Regulator (LQR)

The Linear Quadratic Regulator (LQR) exhibits significant core advantages based on a unique theoretical foundation. Its core lies in achieving control optimization through a quadratic cost function, which performs excellently in real time. By solving the Riccati equation offline, LQR has stable online computation time, which makes it highly suitable for the deployment requirements of embedded systems. Meanwhile, LQR has parameter robustness and can effectively tolerate model linearization errors, making it widely used in mass-produced vehicle controllers. In terms of applicable scenarios, LQR performs outstandingly in medium- and low-speed scenarios. However, LQR also has certain limitations, as its dependence on linear models is more evident [20].

To talk about this LQR, we must start with LQ Optimal Control. We study a linear time-varying system:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \quad (23)$$

The core of optimal control is to make the system run efficiently by minimizing some costs. When this cost is embodied in a quadratic functional form and the system itself is

linear, such problems are called linear quadratic problems. The controller designed for such problems, i.e., the solution, is called the linear secondary regulator (LQR). LQR uses a full-state feedback mechanism to determine the optimal control input by weighted summing of different states. It is part of optimal control theory and adopts a control architecture with full-state feedback. By constructing a quadratic cost function covering future state changes in the system and the amount of actuator input, LQR can calculate the gain matrix that minimizes the total cost, thus obtaining the desired control sequence. The cost of designing an infinite-time LQR problem is:

$$J = \int_0^{\infty} (x^T Q x + u^T R u) dt, Q = Q^T, R = R^T, Q \geq 0, R > 0 \quad (24)$$

Generally speaking, the Q matrix and the R matrix are set as unit diagonal matrices. The elements on these diagonal matrices represent the weight coefficients of different state variables and control variables, respectively. The larger the weight, the more we express that during the design process, the greater the attention this variable receives. The integral symbol represents the accumulation process of the cost function value from the beginning of control to the final moment. Since it is a quadratic form, the cost always remains within the positive value range and eventually approaches 0. Our design goal is to find a series of control sequences that minimize this cumulative cost.

The optimal control sequence $u = -Kx$ is brought into the cost function, and x is separated to obtain:

$$J = \int_0^{\infty} (x^T Q x + u^T R u) dt = \int_0^{\infty} (x^T (Q + K^T R K) x) dt \quad (25)$$

Assume that the original integral function is:

$$-x^T P x = \int_0^{\infty} (x^T (Q + K^T R K) x) dt \quad (26)$$

where P is a symmetric matrix, $P = P^T > 0$.

Because x and u are functions about time t , we can obtain the time derivative on both sides of the above equation at the same time:

$$-\frac{d}{dt}(x^T P x) = x^T (Q + K^T R K) x = -(\dot{x}^T P x + x^T P \dot{x}) \quad (27)$$

The system equation becomes

$$\dot{x} = (A - BK)x \quad (28)$$

Combined with the Formulas (26) and (27) as follows:

$$-x^T (Q + K^T R K) x = x^T \left[(A - BK)^T P + P(A - BK) \right] x \quad (29)$$

Satisfy P as follows: $A^T P + PA + Q - K^T B^T P B K^T R K = 0$, $-(Q + K^T R K) = (A - BK)^T P + P(A - BK)$.

$K = R^{-1} B^T P$ is brought into the above formula (not deriving for the time being) to obtain the algebraic Riccati equation:

$$PA + A^T P - P B R^{-1} B^T P + Q = 0 \quad (30)$$

4. MPC, PP, and LQR Experimental Verification

The experiment sets the vehicle to start on the preset path and then gradually approaches the preset path. The simulation is designed to enable the car to reach the endpoint

as smoothly and accurately as possible along the designated trajectory. The simulation experiment scene is constructed as a composite closed path without obstacle interference, which consists of straight line segments alternating with bends of constant curvature. The initial position of the experimental object is set as a lateral offset of 2 m with respect to the path coordinate system, and the trajectory planning algorithm achieves progressive convergence with the preset reference path. Its discretization iteration period is set to $\Delta t = 200$ ms. During this simulation, the vehicle kinematics model is used to update the vehicle's position and posture, and the speed is 3 m/s, 7 m/s, and 10 m/s, respectively, and the time interval is set to 0.2 s. The selection of a speed range of 3 m/s (10.8 km/h) to 10 m/s (36 km/h) is mainly based on the following three considerations: firstly, this range covers typical autonomous driving scenarios, and low-speed scenarios (3 m/s) correspond to urban congestion, autonomous parking in parking lots, park logistics vehicles, etc. For example, L4 level autonomous driving defined in SAE J3016 standard [21] in the United States usually requires a speed of less than 15 km/h (≈ 4.17 m/s) in "restricted areas" (such as parking lots). The medium- to high-speed scenario (7–10 m/s) corresponds to urban main roads (speed limit of 30–50 km/h) and highway ramp scenarios. According to the ISO 11270:2020 standard [22], the typical test speed range for autonomous vehicles on structured roads (such as urban roads) is 20–40 km/h (≈ 5.56 – 11.11 m/s). Secondly, it complies with international standards and testing specifications. ISO 11270: The 2020 "Performance Requirements for Lane Keeping Assistance Systems (LKAS) in Intelligent Transportation Systems" recommends a testing speed range of 10–40 km/h (≈ 2.78 – 11.11 m/s), covering the 3–10 m/s range of this study [22]. The Chinese GB/T 40429-2021 Road Test Capability Evaluation Method for autonomous vehicles clearly requires that the test scenario should include "low speed (≤ 15 km/h)" and "medium speed (15–60 km/h)" sections [23]. Thirdly, in line with academic research consensus, similar speed range validation lateral control algorithms have been widely used in research in recent years. For low-speed (3–5 m/s) aspects, Li et al. [24] used the 3 m/s validation PP algorithm in warehouse AGV control; in terms of medium to high speed (7–10 m/s), Zhang et al. [25] studied the lateral error standard deviation of MPC at a speed of 10 m/s and found that it can effectively handle acceleration constraints to ensure ride comfort. Three algorithms were set up using MATLAB software for experiments, and the trajectory data (PP, LQR, MPC) of the three algorithms were loaded and drawn, each using a different line type and color. The simulation results are as follows:

Table 1 is obtained by statistically analyzing the lateral error in Figure 5, the heading error in Figure 6, and the time consumption in Figure 7 under the condition of 3 m/s; Table 2 is obtained by statistically analyzing the lateral error in Figure 8, the heading error in Figure 9, and the time consumption in Figure 10 under the condition of 7 m/s; Table 3 is obtained by statistically analyzing the lateral error in Figure 11, the heading error in Figure 12, and the time consumption in Figure 13 under the condition of 10 m/s. It can be seen from the three tables.

Table 1. Comparison of MPC, PP, and LQR data with vehicle speed of 3 m/s.

Algorithm	Lateral Error	Heading Error	Time-Consuming
MPC	−0.29~0.27	−0.687~1.98	0.108~0.50
PP	−0.45~0.033	−0.038~1.958	0.119~0.39
LQR	−0.176~0.009	−0.42~1.98	0.11~0.302

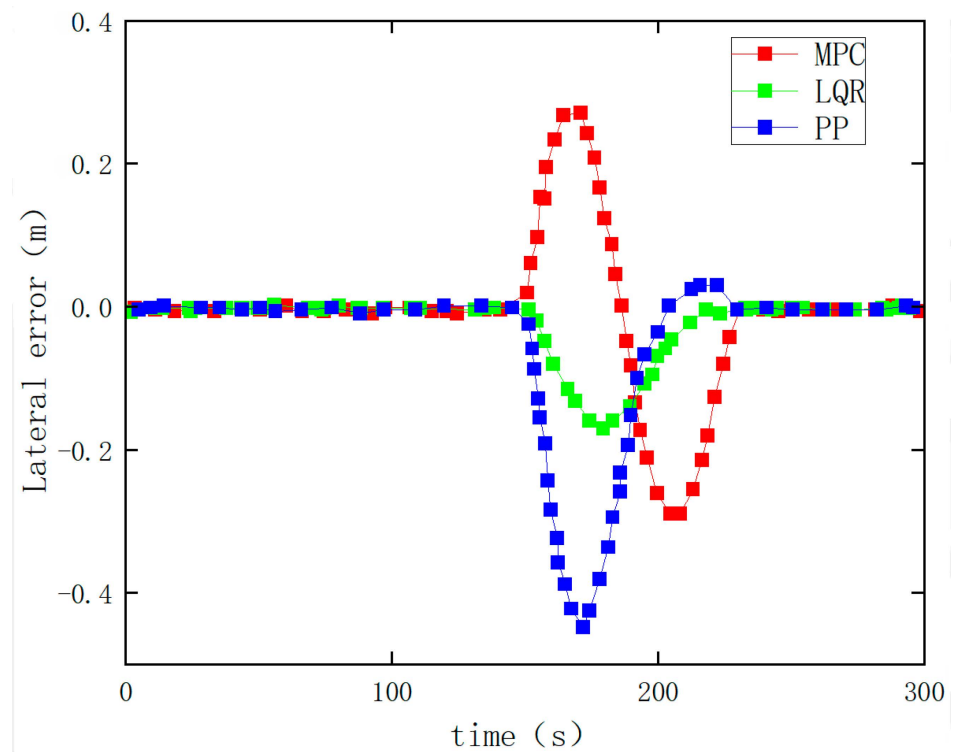


Figure 5. Comparison of lateral errors with vehicle speed of 3 m/s.

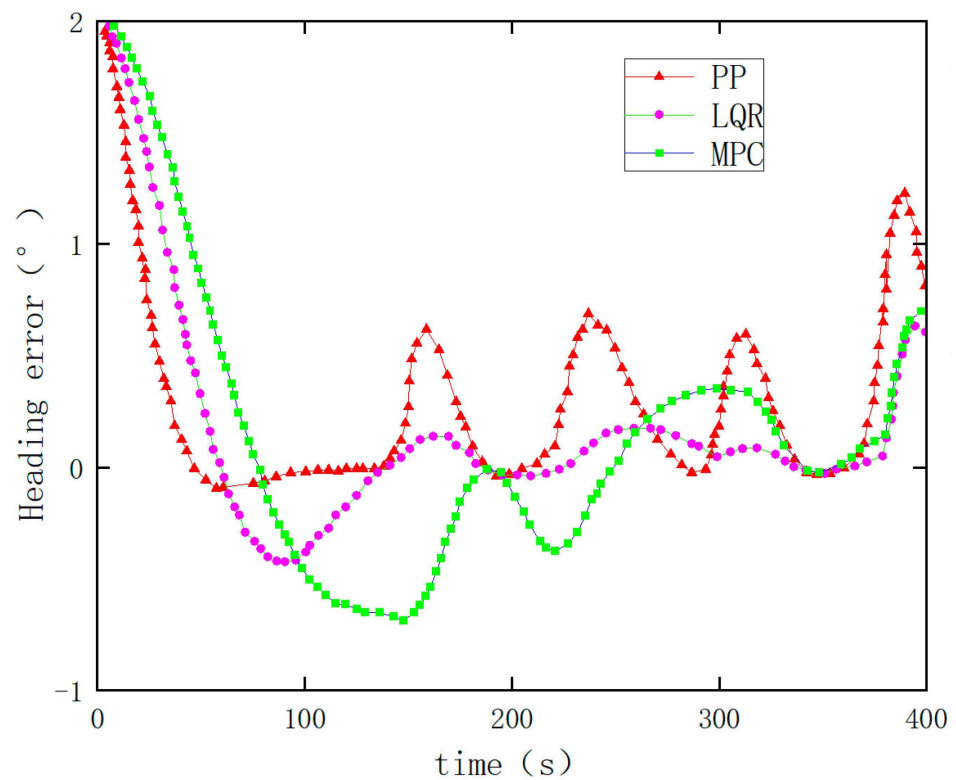


Figure 6. Comparison of heading errors with vehicle speed of 3 m/s.

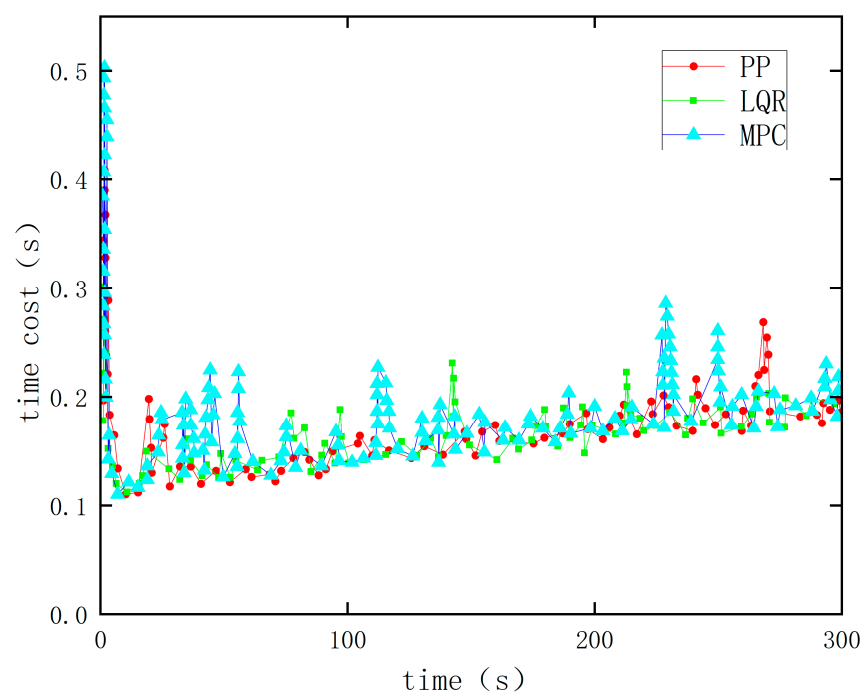


Figure 7. Comparison of the algorithm with a vehicle speed of 3 m/s.

Table 2. Comparison of MPC, PP, and LQR data with vehicle speed of 7 m/s.

Algorithm	Lateral Error	Heading Error	Time-Consuming
MPC	−0.29~0.27	−0.687~1.98	0.11~0.559
PP	−0.45~0.033	−0.038~1.958	0.12~0.379
LQR	−0.176~0.009	−0.42~1.98	0.11~0.371

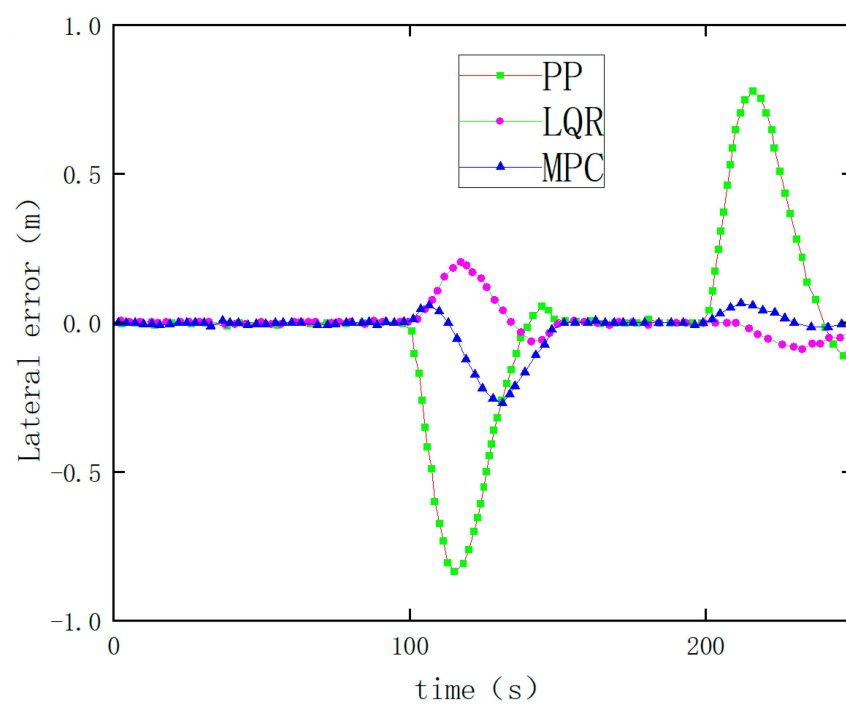


Figure 8. Comparison of lateral errors at a velocity of 7 m/s.

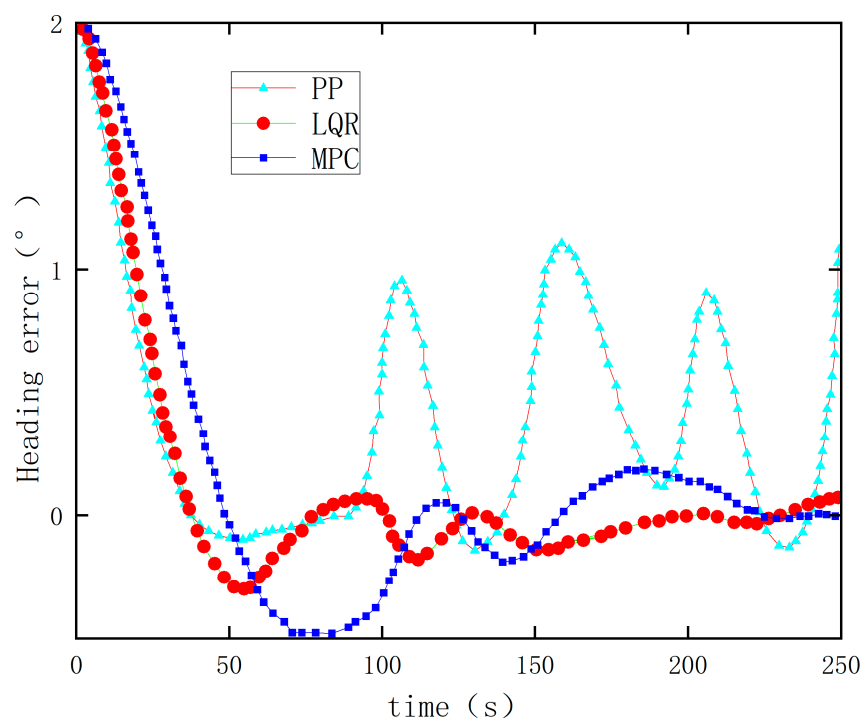


Figure 9. Comparison of heading errors at a velocity of 7 m/s.

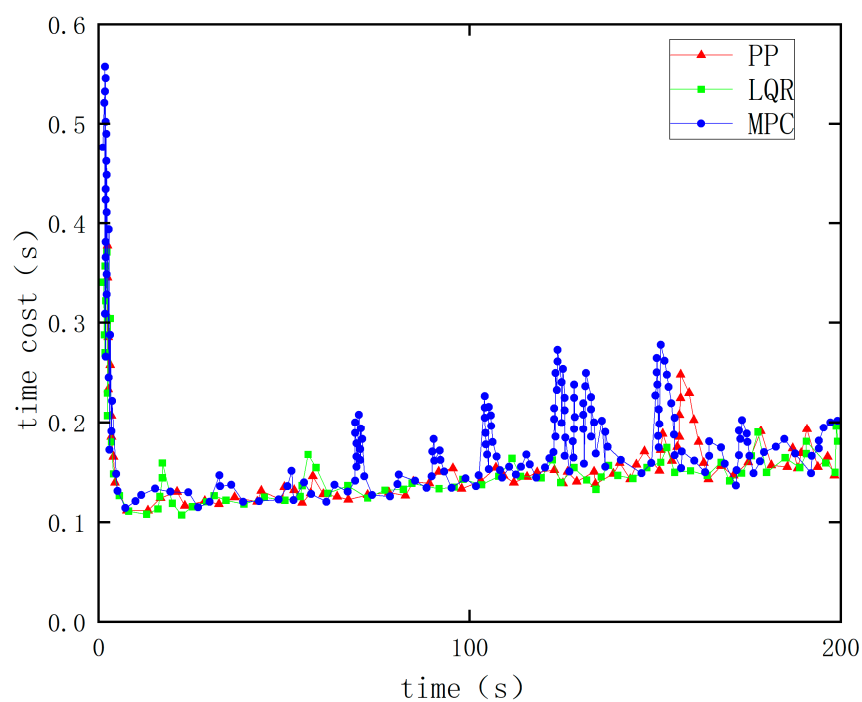


Figure 10. Comparison of the algorithm with a speed of 7 m/s.

Table 3. Comparison of MPC, PP, and LQR data with vehicle speed of 10 m/s.

Algorithm	Lateral Error	Heading Error	Time-Consuming
MPC	−0.07~0.26	−0.51~1.99	0.12~0.50
PP	−0.78~0.056	−0.098~1.981	0.084~0.39
LQR	−0.073~0.266	−0.586~1.985	0.11~0.44

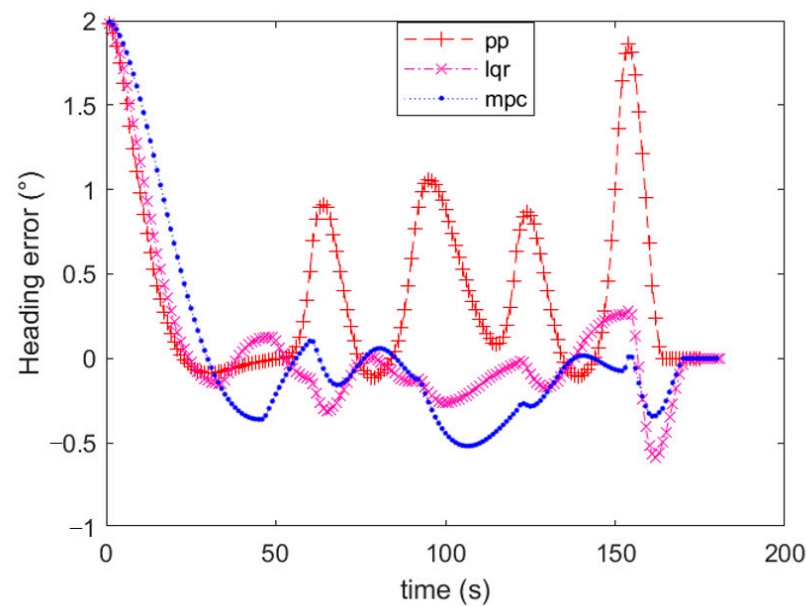


Figure 11. Comparison of heading errors with vehicle speed of 10 m/s.

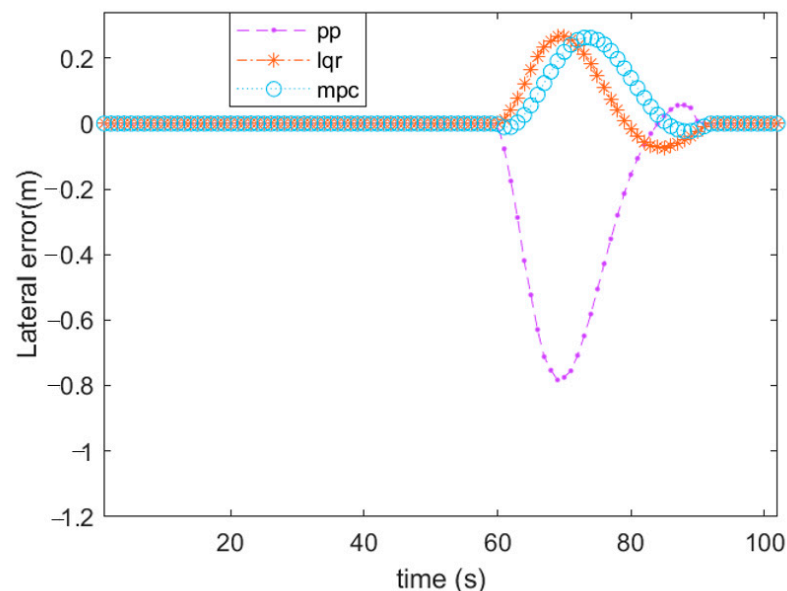


Figure 12. Comparison of transverse errors with vehicle speed of 10 m/s.

(1) Lateral Error

At all speeds, the lateral error range of MPC (Model Predictive Control) is relatively stable, maintaining between -0.29 and 0.27 m. The PP (Pure Pursuit) algorithm has a large lateral error at low speeds, especially when the vehicle speed is 3 m/s; the error reaches $-0.45 \sim 0.033$ m. However, as the speed increases, the error range does not change significantly. The LQR (Linear Quadratic Regulator) algorithm performs relatively stable in lateral errors, especially at high speeds (10 m/s), and the error range is similar to that of MPC.

(2) Heading Error

At different speeds, the heading error range of the MPC algorithm is wide ($-0.687 \sim 1.98^\circ$), indicating that it has certain flexibility in heading control, but this flexibility may also bring about instability in heading control, especially at high speeds. The heading error range of the PP algorithm varies greatly at different speeds, and overall, it is more stable than

MPC (such as $-0.038\sim 1.958^\circ$ at 3 m/s), which implies that the PP algorithm may focus more on stability in heading control than on flexibility. The heading error range of the LQR algorithm is similar to that of MPC, but overall, it is more concentrated (such as $-0.42\sim 1.98^\circ$ at 3 m/s). This shows that the LQR algorithm has a certain degree of flexibility while maintaining relative stability in heading control.

(3) Algorithm Execution Time

As the speed increases, the time consumption of the MPC algorithm increases slightly (from 0.108~0.5 at 3 m/s to 0.12~0.5 at 10 m/s). The time consumption of the PP algorithm fluctuates greatly at different speeds, especially as the time-consuming process decreases at 10 m/s (0.084~0.39 s). The time-consuming process of the LQR algorithm is relatively stable and shorter at different speeds (0.11~0.302 s at 3 m/s, and 0.11~0.371 s at 7 m/s, 0.11~0.44 s at 10 m/s), indicating that it has certain advantages in real time.

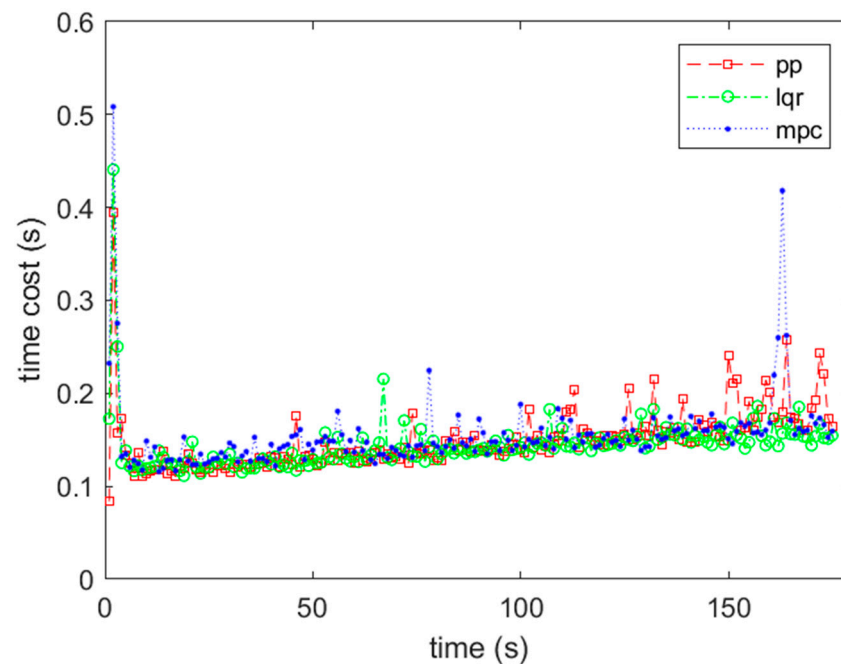


Figure 13. Comparison of algorithm time-consuming.

5. Conclusions

Based on the kinematic model and geometric model, this paper introduces the lateral control algorithms of MPC, PP, and LQR, respectively, and uses MATLAB software to verify the performance of various algorithms when making cars walk according to the path.

In terms of lateral error, the MPC and LQR algorithms are $-0.29\sim 0.27$ and $-0.176\sim 0.009$ at low speeds, respectively, and their performance is relatively stable. As the speed increases, the performance of the MPC algorithm will exceed that of LQR, while the PP algorithm is 3 m/s and 7 m/s is $-0.45\sim 0.033$, indicating that the PP algorithm has a large error at low speed. In terms of heading error, all algorithms have a large error range, but the high-speed MPC algorithm ($-0.51\sim 1.99$) and the LQR algorithm ($-0.586\sim 1.985$) perform slightly better than the PP algorithm ($-0.098\sim 1.981$). In terms of algorithm time consumption, the LQR algorithm consumes the shortest and most stable time (0.11~0.302 at 3 m/s, 0.11~0.371 at 7 m/s, 0.11~0.44 at 10 m/s), and the MPC algorithm consumes relatively long time (3 m/s) (3 m/s). 0.108~0.50, 0.11~0.559 at 7 m/s, 0.12~0.5 at 10 m/s), while the PP algorithm takes time to change at different speeds (0.119~0.39 at 3 m/s, 0.12~0.379 at 7 m/s, 0.084~0.39 at 10 m/s).

Therefore, when selecting an algorithm, trade-offs need to be made based on specific application scenarios and requirements. If you pursue higher control accuracy and stability, you can consider using the MPC or LQR algorithm; if you pay more attention to real-time and computing efficiency, you can consider using the PP algorithm. At the same time, it is also necessary to consider comprehensively based on factors such as the actual operating speed of the vehicle and road conditions.

This article jointly analyzes the lateral error, heading error, and time consumption of MPC, PP, and LQR algorithms in 3 m/s, 7 m/s, and 10 m/s scenarios, revealing the impact of speed changes on algorithm performance. A quantitative benchmark for algorithm performance is established through indicators such as error range (e.g., PP lateral error reaches $-0.78\text{ m}\sim 0.056\text{ m}$ at 10 m/s), standard deviation (MPC lateral error standard deviation is reduced by 68% compared to PP), and time consumption growth rate (PP time increases by 95% with speed). Through multi-speed simulation experiments, we reveal the performance differences and speed dependence of MPC, PP, and LQR algorithms in lateral control. We quantify the range of algorithm errors, computation time, and their correlation with speed, providing a data-driven decision-making basis. Based on experimental results, we construct algorithm matching rules for different autonomous driving scenarios (such as urban roads and highways), balancing accuracy, real-time performance, and system resource constraints.

While this study offers systematic comparisons of lateral control algorithms during speed variations, it has several limitations. The simulations are based on idealized road conditions and overlook real-world disturbances like crosswind effects and road curvature variations. However, in future research, we plan to create multi-physics road scenarios with improved environmental modeling and design a layered control architecture coupling. In the research of lateral control algorithms for autonomous driving, MPC, PP, and LQR each have their own advantages but still have significant limitations. Although MPC can handle multi-constrained problems through rolling optimization, its high computational complexity (average time of 12.5 ms) and strong model dependency limit its application on resource-constrained platforms. In the future, it can be improved in real-time and robustness through lightweight frameworks and hybrid data model driving (combined with deep learning to predict dynamic characteristics). PP is efficient in low-speed scenarios, but the lateral error significantly increases at high speeds (such as $-0.78\text{ m}\sim 0.056\text{ m}$ at 10 m/s), and it is difficult to cope with dynamic disturbances such as low adhesion road surfaces. Therefore, it is necessary to enhance adaptability through dynamic forward distance adjustment and sliding mode control compensation mechanisms. Although LQR has strong real-time performance (stable time between 0.11~0.44 ms), its linear model dependence leads to an increase in heading error in high-speed and high curvature scenarios (such as 1.985 rad). In the future, it can be extended to nonlinear iterative LQR and integrated with constraint processing to improve the performance of complex working conditions. These improvements will drive collaborative breakthroughs in the accuracy, efficiency, and robustness of lateral control algorithms, providing key technical support for the large-scale implementation of autonomous driving.

Author Contributions: Concept: W.Z.; Methodology: W.Z.; Software: W.Z. and J.W.; Verified: W.Z., J.W. and T.P.; Formal analysis: W.Z. and J.W.; Survey: W.Z. and T.P.; Source: W.Z.; Data Management: W.Z.; Writing—First draft writing: W.Z.; Writing—Review and Editor: J.W. and T.P.; Visualization: J.W.; Supervision: J.W.; Project Management: J.W.; Funding Acquisition: J.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Enterprise Technology Development Committee's Simulation Test System Project for the Development of New Materials and Technologies for Automotive Products, Grant No. 60000102279.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article; further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhu, X. Research on Trajectory Tracking Control for Autonomous Vehicles Based on Fuzzy Adaptive MPC. Master's Thesis, Xihua University, Chengdu, China, 2022.
2. You, Z. *Research on Trajectory Tracking Control for Autonomous Vehicles Based on MPC Algorithm*; Jilin University: Changchun, China, 2018.
3. Sun, Y. Research on Trajectory Tracking Control Algorithm for Autonomous Vehicles Based on Model Predictive Control. Master's Thesis, Beijing Institute of Technology, Beijing, China, 2015.
4. Changoski, V.; Vasileva, A.; Jakimovska, K.; Danev, D. Implementing Model Predictive Control (MPC) in Steer-by-Wire Systems for Future Automated Vehicles. *IOP Conf. Ser. Mater. Sci. Eng.* **2024**, *1311*, 012024. [CrossRef]
5. Zhu, Z.W.; Jiang, Z.T.; Wang, W.Z.; Jiang, W.H.; Li, S.H. Multi-shooting nonlinear MPC trajectory tracking control for autonomous vehicles. *Control. Theory Appl.* **2025**, 1–9. Available online: <http://kns.cnki.net/kcms/detail/44.1240.TP.20250311.1350.016.html> (accessed on 27 March 2025).
6. Xie, X.Y.; Wang, Y.H.; Jin, L.S.; Guo, B.; Li, K. Trajectory Tracking Control of Intelligent Vehicles Based on Changing the Time Step of the Control Horizon. *J. Jilin Univ. (Eng. Technol. Ed.)* **2024**, *54*, 620–630. [CrossRef]
7. Kayacan, E.; Ramon, H.; Saeys, W. Robust trajectory tracking error model-based predictive control for unmanned ground vehicles. *IEEE/ASME Trans. Mechatron.* **2015**, *21*, 806–814. [CrossRef]
8. Liu, Y.; Shi, G.; Chen, Y.; Shi, L. A Fuzzy Path Tracking Control Method Based on the Pure Pursuit Model. *Mach. Des. Res.* **2022**, *38*, 136–140+157.
9. Liu, W.; Zhang, C.; Jiang, H.; Guan, L.; Cao, T.; Zhang, Z.; Wu, X. Path Tracking Control for Low-Speed Intelligent Vehicles Based on Improved Pure Pursuit. *Sci. Technol. Eng.* **2024**, *24*, 10983–10992.
10. Wang, M.; Lv, X.; Chen, J.; Su, X. Improved Pure Pursuit Algorithm Based Path Tracking Method for Autonomous Vehicle: Regular Papers. *J. Adv. Comput. Intell. Inform.* **2024**, *28*, 1034–1042. [CrossRef]
11. Chen, R.D. Research on Trajectory Tracking Control of Unmanned Racing Cars Based on the PP Algorithm. Master's Thesis, Guangdong University of Technology, Guangzhou, China, 2022. [CrossRef]
12. Ni, J.; Wang, Y.; Li, H.; Du, H. Path Tracking Motion Control Method of Tracked Robot Based on Improved LQR Control. In Proceedings of the 2022 41st Chinese Control Conference (CCC), Hefei, China, 25–27 July 2022.
13. Trapanese, S.; Naddeo, A.; Cappetti, N. A preventive evaluation of perceived postural comfort in car-cockpit design: Differences between the postural approach and the accurate muscular simulation under different load conditions in the case of steering-wheel usage. In Proceedings of the SAE 2016 World Congress and Exhibition, Detroit, MI, USA, 12–14 April 2016.
14. Gao, L.; Tang, F.; Guo, P.; He, J. Research on Improved LQR Method for Lateral Motion Control of Autonomous Driving. *Mech. Sci. Technol. Aerosp. Eng.* **2021**, *40*, 435–441.
15. Chen, L.; Qin, Z.; Kong, W.; Chen, X. LQR Lateral Control of Intelligent Vehicles Based on Optimal Front Wheel Side Slip Force. *J. Tsinghua Univ. (Sci. Technol.)* **2021**, *61*, 906–912.
16. Lu, Y.; Shen, Y.F.; Gao, L.Z. Optimal Design of Lateral LQR Controller for Autonomous Driving Based on Improved Genetic Algorithm. *Mach. Des. Res.* **2024**, *40*, 202–208. [CrossRef]
17. Zhang, Y.J.; Feng, L.X.; Li, G.B. An Adaptive LQR Path Tracking Control Method for Intelligent Vehicles Considering Time-Varying Parameters. *Comput. Sci.* **2025**, 1–11. Available online: <http://kns.cnki.net/kcms/dtail/50.1075.tp.20241126.1821.011.html> (accessed on 27 March 2025).
18. Gong, J.; Liu, K.; Qi, J. *Model Predictive Control for Driverless Vehicles: A Major Tool for National Development*, 2nd ed.; Posts & Telecom Press: Beijing, China, 2020.
19. Gan, H.; Liao, X. Lateral Control of Intelligent Electric Vehicles Based on Model Predictive Control Algorithm. *West. China Commun. Sci. Technol.* **2023**, *11*, 194–199. [CrossRef]
20. Zheng, C.; Du, Y.; Liu, Z. A Review of Lateral Control Methods for Autonomous Vehicles. *Automob. Eng.* **2024**, *5*, 1–10. [CrossRef]
21. SAE International. *J3016B: Taxonomy and Definitions for Terms Related to Driving Automation Systems*; SAE: Warrendale, PA, USA, 2021.
22. ISO 11270:2020; Intelligent Transport Systems—Lane Keeping Assistance Systems (LKAS)—Performance Requirements and Test Procedures. ISO: Geneva, Switzerland, 2020.

23. GB/T 40429-2021; Evaluation Method of Road Test Capability of Autonomous Vehicle. ISO: Geneva, Switzerland, 2021.
24. Li, X.; Zhang, Y.; Wang, C.; Chen, D.; Liu, E.; Zhao, F.; Sun, G.; Huang, H.; Wu, J.; Zhou, K.; et al. Low-Speed Path Tracking for Logistics AGVs Using Adaptive Pure Pursuit. *IEEE Trans. Ind. Electron.* **2022**, *69*, 4321–4330. [[CrossRef](#)]
25. Zhang, Y.; Liao, M.; Zhang, R.; Huang, W.; Shao, M. Research on Longitudinal Control of Vehicles Based on Model Predictive Control. *Model. Simul.* **2023**, *12*, 290–301. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.