sensors

MDPI

Article

A New Trajectory Tracking Algorithm for Autonomous Vehicles Based on Model Predictive Control

Zhejun Huang [1,2,3], Huiyun Li [1,2,3,*], Wenfei Li [1,2,3], Jia Liu [1,2,3], Chao Huang [4], Zhiheng Yang [1,2,3] and Wenqi Fang [5]

[1] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China; zj.huang@siat.ac.cn (Z.H.); wf.li1@siat.ac.cn (W.L.); jia.liu1@siat.ac.cn (J.L.); zh.yang@siat.ac.cn (Z.Y.)
[2] CAS Key Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen Institutes of Advanced Technology, Shenzhen 518055, China
[3] Guangdong-Hong Kong-Macao Joint Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen 518055, China
[4] Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hong Kong 999077, China; hchao.huang@polyu.edu.hk
[5] Research Center of Digital Intelligence Technology, Nanhu Lab, Jiaxing 314033, China; wqfang@nanhulab.ac.cn
* Correspondence: hy.li@siat.ac.cn; Tel.: +86-132-6580-5460

Abstract: Trajectory tracking is a key technology for precisely controlling autonomous vehicles. In this paper, we propose a trajectory-tracking method based on model predictive control. Instead of using the forward Euler integration method, the backward Euler integration method is used to establish the predictive model. To meet the real-time requirement, a constraint is imposed on the control law and the warm-start technique is employed. The MPC-based controller is proved to be stable. The simulation results demonstrate that, at the cost of no or a little increase in computational time, the tracking performance of the controller is much better than that of controllers using the forward Euler method. The maximum lateral errors are reduced by 69.09%, 47.89% and 78.66%. The real-time performance of the MPC controller is good. The calculation time is below 0.0203 s, which is shorter than the control period.

Keywords: autonomous driving; trajectory tracking; real-time control; model predictive control

Citation: Huang, Z.; Li, H.; Li, W.; Liu, J.; Huang, C.; Yang, Z.; Fang, W. A New Trajectory Tracking Algorithm for Autonomous Vehicles Based on Model Predictive Control. *Sensors* **2021**, *21*, 7165. https://doi.org/10.3390/s21217165

Academic Editor: Soufiene Djahel

Received: 3 September 2021
Accepted: 26 October 2021
Published: 28 October 2021

1. Introduction

Research in autonomous driving has aroused increasingly more attention of late [1,2]. The most basic and important goal of an autonomous passenger vehicle is to free people from driving and safely take passengers from an initial state to a final state in a desired interval of time. The architecture of contemporary autonomous driving systems is typically organized into the perception system and the decision-making system [3]. The perception system takes charge of estimating the vehicle states and representing the surrounding environment using data from sensors, including Light Detection and Ranging (LIDAR), Radio Detection and Ranging (RADAR), cameras, a Global Positioning System (GPS), and an Inertial Measurement Unit (IMU). In particular, camera data is of vital importance. Tesla released its fully self-driving version 9 Beta software on 10 July 2021, which relies on camera vision and neural net processing to deliver autopilot. The Lane Support System (LSS) uses cameras to identify the road lines and alert drivers to potential hazards. However, there is still much uncertainty regarding the vision needs of LSS and the results of the experimental tests for LSS are quite limited [4,5]. Cafiso and Pappalardo [4] developed logit models to investigate road characteristics and conditions that affects LSS performance and employed the Firth's penalized maximum-likelihood method to estimate the logistic regression coefficients and standard errors to describe the rareness of the events. They gave threshold values for the luminance coefficient in diffuse lighting conditions and horizontal

curvature radius, and presented remarks on road maintenance and design standards. Pappalardo et al. [5] experimentally tested LSS performance in two lane rural roads with distinct geometric alignments and road marking conditions. They proposed a decision tree method to analyze the cause of the LSS faults and the effects of the variables involved. On the other hand, the decision-making system takes charge of navigating a car from the current position to a goal position safely, feasibly, timely, and comfortably [6]. The decision-making system can be further divided into three subsystems: a decision and planning system, a control system, and an actuation system. Of them, motion planning is a key autonomous driving technique. Li and Shao [7] proposed a motion planner for autonomous parking, and the time-optimal dynamic optimization problem with vehicle kinematics, collision-avoidance conditions and mechanical constraints was solved using a simultaneous approach using the interior-point method. Zhang [8] proposed a hierarchical three-layer trajectory planning framework to realize real-time collision avoidance on highways under complex driving conditions. Therefore, a general framework of an autonomous driving system is shown in Figure 1. Besides a perception and decision-making system, advanced X-by-wire chassis, including drive-by-wire, steer-by-wire, brake-by-wire and active/semi-active suspension subsystems are of vital importance to improving the performance and safety of connected and autonomous vehicles. Zhang et al. [9] proposed a fault-tolerant control method for steer-by-wire systems to mitigate the undesirable influence of front wheel steering angle sensor faults via the use of the Kalman filtering technique. A complete and systematic survey on chassis coordinated control methods for full X-by-wire vehicles can be found in [10]. Here we focus on trajectory tracking, which is a key technology for precisely controlling autonomous vehicles.

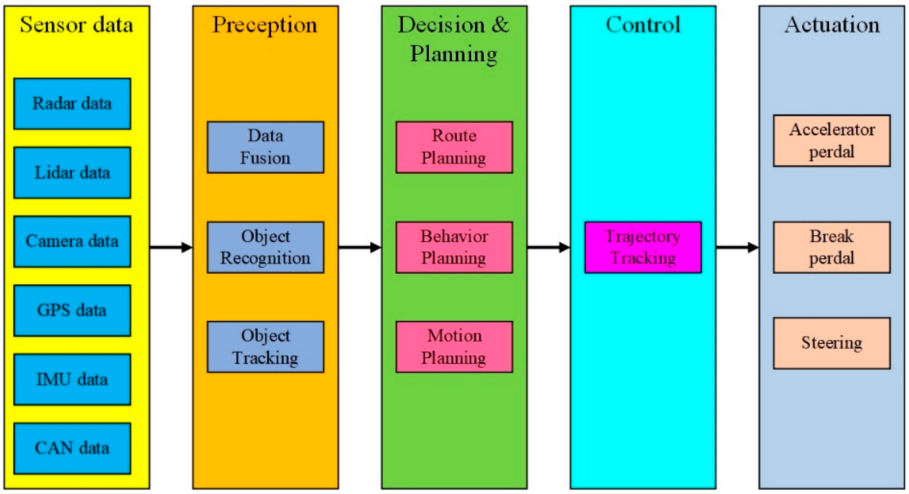

Figure 1. General framework of an autonomous driving system comprising perception, decision and planning, control and actuation.

The trajectory tracking algorithms are designed to ensure that a vehicle follows a predetermined trajectory generated either offline using navigation systems or online using the motion planning module. The performance of trajectory tracking directly determines the performance of autonomous vehicles, which involves driving safety, passenger comfort, travel efficiency, and energy consumption [11]. The trajectory tracking control of autonomous vehicles is a challenging research area because these systems typically are nonlinear systems with non-holonomic constraints.

Pure-pursuit [12] and the Stanley method [13] are two prevalent geometric controllers. The main advantage of these methods is that they use simple geometric models with few parameters, and therefore can give timely feedback on the current state and constraints to meet the real-time requirement of an autonomous vehicle. The pure-pursuit method and its variants are one of the most commonly used methods to solve the path-tracking problem for mobile robots [14]. The Stanley method is the path-tracking approach used by

Stanley, Stanford University's autonomous car; Stanley won the DARPA Grand Challenge in 2005 [13]. However, these methods have their limitations. Pure-pursuit control works as a proportional controller of the steering angle operating on the cross-track error by calculating the curvature from the current position to some goal position. When the look-ahead distance is too large, its performance is poor, and the vehicle may cut corners when changing direction or making a U-turn. The Stanley method considers both the heading and cross-track errors and therefore it is more effective and steady than the pure-pursuit method. But it does not perform well on discontinuous paths. To sum up, geometric-based tracking controllers (pure pursuit, Stanley, etc.) have a simple structure and are easy to implement. However, they are not suitable for applications that need to consider vehicle dynamics (e.g., high-speed trajectory tracking, extreme path curvature, etc.). It is also difficult to achieve a trade-off between stability and tracking performance [15].

Proportional-Integral-Derivative controllers (PID) [16] and sliding model controllers (SMC) [17] are two prevalent classical control algorithms. Although PID controllers have good tracking performance, there is a major challenge in the tuning of the parameters because of the vehicle and tire nonlinearities. SMC is a well-developed nonlinear state-feedback controller and has been used to design vehicle trajectory tracking controllers. Because of the nonlinear control law, SMC shows good tracking accuracy. However, there are several drawbacks: first, its performance is sensitive to the sampling rate of the controller; second, chattering problems exist under certain conditions [18]; third, robustness is only guaranteed on the sliding surface; and lastly, it needs prior knowledge [19]. To sum up, compared with geometric-based tracking controllers, model-based tracking methods are more feasible and reliable in real driving scenarios at the cost of the increase in computational burden and complexity.

Reinforcement learning (RL) has shown an ability to achieve super-human results at turn-based games like Go [20] and chess [21]. Deep RL has been applied to the decision-making system of autonomous driving in several simulated environments [22]. Mohammadi et al. [23] proposed an optimal tracking controller for nonlinear continuous-time systems with time-delay, mismatched external disturbances, and input constraints, using the technique of integral reinforcement learning and a Hamilton-Jacobi-Bellman equation. However, there are two main limitations for RL-based methods. First, they require large amounts of data to build up a feasible model; specifically, data is sometimes expensive and hard to obtain. Second, they require a sufficiently long time to train the model to complete the specific tasks due to significant data manipulation. The performance of the controllers using machine-learning methods relies on the learning capability of the model and the quality of the data.

Model Predictive Control (MPC) has been applied to trajectory planning and tracking of an autonomous vehicle due to its flexibility and ability to compute optimal solutions with hard and soft constraints [24,25]. Shen et al. [24] proposed a unified receding-horizon optimization scheme for the integrated path-planning and tracking control of an autonomous underwater vehicle using nonlinear MPC techniques. Borrelli et al. [25] proposed a novel approach to autonomous steering systems based on an MPC scheme. The general framework of an MPC structure is shown in Figure 2. However, these MPC-based tracking controllers are feasible only in low-speed scenarios.

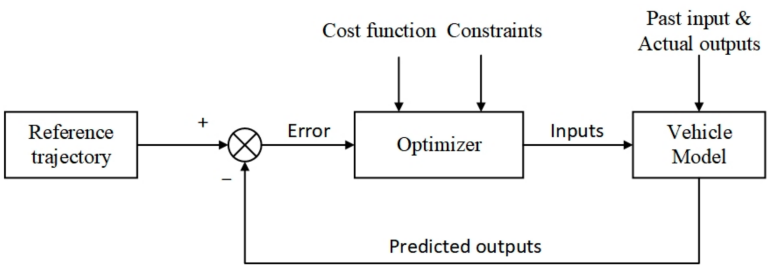

Figure 2. General framework of an MPC structure.

The accuracy of the trajectory tracking control can be greatly improved by improving the accuracy of the predictive model. Most researchers have attempted to improve the accuracy of the kinematic or dynamic model to improve the accuracy of the controller. Few people paid attention to the computation errors during the integration process. With the accumulation of the computation errors, the controller could lose its stability or an accident might even result.

In this paper, we propose a new trajectory-tracking algorithm based on MPC. Instead of using the forward Euler integration method, the backward Euler integration method is used to establish the predictive model.

The contributions here can be summarized as follows:

- A trajectory tracking method is proposed based on MPC. Instead of using the forward Euler method, the backward Euler method is used to establish the predictive model. The proposed method is designed to meet the real-time requirement of autonomous vehicles by structuralizing the control law and employing the warm-start strategy.
- Unlike conventional MPC-based controllers, both the acceleration and steer angle are control inputs. The proposed MPC-based controller can automatically adjust the velocity according to the information of the reference trajectory.
- The dynamic regret of the proposed controller is tightly bounded, and the closed-loop controller is proved to be stable.
- The MPC controller using the backward Euler method has a better tracking accuracy in the lateral error, and it is more robust.

This paper is organized as follows. The MPC-based controller of autonomous vehicles is described in Section 2. After that, the stabilizability of the controller is discussed in Section 3. Simulation results are shown in Section 4. Section 5 concludes this paper by summarizing all of the main results.

2. Control Design

Establishing a prediction model and designing a rolling optimization function are the kernels of designing a path tracking controller. Due to the strongly nonlinearity of vehicle dynamics, it is very hard to establish a model to describe the actual vehicle dynamics. Researchers generally use Ackermann steering geometry and its simplified bicycle models to describe the vehicle kinematics and dynamics. MPC schemes using dynamic vehicle models and various tire models are generally computationally expensive, and tire models may become singular at low speeds [26]. Kong et al. [26] compared a kinematic and a dynamic bicycle model, and showed that both models could correctly predict a vehicle's future states, and combining MPC schemes with a simple kinematic bicycle model is less computationally expensive. Polack et al. [27] compared a 3-DOF kinematic bicycle model with a 9-DOF model, and showed that the 3-DOF model could capture enough of the non-holonomic constraints of the actual vehicle dynamics. When the maximum-allowed lateral acceleration of a vehicle was no greater than 0.5 g m/s^2, where g is the acceleration due to gravity, using a 3-DOF kinematic bicycle model produces acceptable results and could generate a feasible track. Chen et al. [28] implemented an MPC-based controller for path-tracking using three vehicle dynamics models: a bicycle model, an 8-DOF model and a 14-DOF model. They showed that the bicycle controller could successfully navigate a vehicle along the given path and calculate the optimal steering sequences faster than the controllers with the 8-DOF and 14-DOF vehicle. They concluded that the bicycle controller is suitable for a possible physical implementation with real-time requirements. Therefore, in this paper, the kinematic model of autonomous vehicles is used [26,29]

$$\begin{aligned} \dot{x} &= v\cos(\varphi+\beta), \\ \dot{y} &= v\sin(\varphi+\beta), \\ \dot{\varphi} &= \tfrac{v\sin(\beta)}{l_r}, \\ \dot{v} &= a, \\ \beta &= \tan^{-1}(\tfrac{l_r}{l_f+l_r}\tan(\delta)), \end{aligned} \tag{1}$$

where x and y are the coordinates of the center of mass in an inertial frame (X,Y). φ is the inertial heading, and v is the longitudinal speed of the vehicle. The parameters l_f and l_r are the distance from the center to the front and rear axles, and δ is the front steering angle. Two front and two rear wheels of the vehicle are combined into single wheels located at the center of the front and rear axle, respectively, as illustrated in Figure 3. β is the slip angle at the center of gravity.

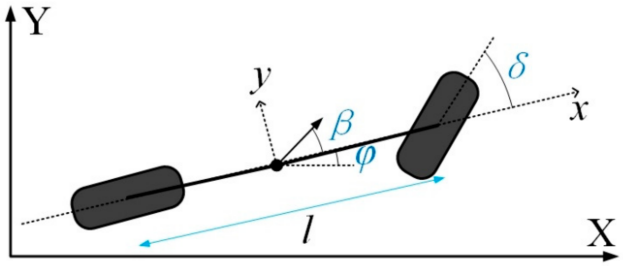

Figure 3. Kinematic rear axle bicycle model of the vehicle.

In our problem, X = [x, y, φ, v] is the vehicle state, U = [a, δ] is the control state. The model is established based on the following assumptions.

- The vehicle is traveling on a flat surface, with the vehicle's movement perpendicular to the road surface ignored.
- Only the front wheel can be steered.
- The wind resistance and ground-side friction that the wheels are subjected to while driving are ignored.
- The wheels always maintain good rolling contact with the ground.
- The impact of the vehicle suspension is not taken into account.
- Load transfer is not considered.

The state-space equations of the vehicle system (1) are continuous in time and cannot be used for the design of the MPC algorithm directly. Therefore, the model of the system was converted to discrete state-space equations by discretizing the state-space equations. We assume that the model can be rewritten as

$$\dot{\mathrm{X}} = f(\mathrm{X}, \mathrm{U}). \tag{2}$$

Generally, the state at $k+1$ instant at time t is computed using the forward Euler integration method

$$\mathrm{X}(k+1|t) = \mathrm{X}(k|t) + T_s\dot{\mathrm{X}}(k|t) = \mathrm{X}(k|t) + T_s f(\mathrm{X}(k|t), \mathrm{U}(k|t)), \tag{3}$$

where T_s is the sampling time.

In this paper, instead of the forward Euler method, the backward Euler method is used to establish the predictive model. Although it requires an extra computation at each iteration, the backward Euler method has great stability properties and its local truncation error is of order $O(T_s^3)$, which is much smaller than $O(T_s^2)$ using the forward Euler method. Hence, the backward Euler method's error generally decreases faster as $T_s \to 0$.

The state at $k+1$ instant at time t is computed using the backward Euler method

$$\begin{aligned} \widetilde{\mathrm{X}}(k+1|t) &= \mathrm{X}(k|t) + T_s f(\mathrm{X}(k|t), \mathrm{U}(k|t)), \\ \mathrm{X}(k+1|t) &= \mathrm{X}(k|t) + T_s f(\widetilde{\mathrm{X}}(k+1|t), \mathrm{U}(k|t)), \end{aligned} \tag{4}$$

Equation system (8) can be rewritten as

$$\begin{aligned} \mathrm{X}(k+1|t) &= \mathrm{X}(k|t) + T_s\widetilde{f}(\mathrm{X}(k|t), \mathrm{U}(k|t)), \\ \widetilde{f}(\mathrm{X}(k|t), \mathrm{U}(k|t)) &= f(\mathrm{X}(k|t) + T_s f(\mathrm{X}(k|t), \mathrm{U}(k|t)), \mathrm{U}(k|t)). \end{aligned} \tag{5}$$

Therefore, the state information of vehicles in the prediction horizon N_P can be obtained

$$\begin{aligned}
\mathrm{X}(k+1|t) &= \mathrm{X}(k|t) + T_s\widetilde{f}(\mathrm{X}(k|t), \mathrm{U}(k|t)),\\
&\vdots\\
\mathrm{X}(k+i|t) &= \mathrm{X}(k+i-1|t) + T_s\widetilde{f}(\mathrm{X}(k+i-1|t), \mathrm{U}(k+i-1|t)),\\
&\vdots\\
\mathrm{X}(k+N_c+1|t) &= \mathrm{X}(k+N_c|t) + T_s\widetilde{f}(\mathrm{X}(k+N_c|t), \mathrm{U}(k+N_c|t)),\\
&\vdots\\
\mathrm{X}(k+N_P|t) &= \mathrm{X}(k+N_P-1|t) + T_s\widetilde{f}(\mathrm{X}(k+N_P-1|t), \mathrm{U}(k+N_c|t)),
\end{aligned} \tag{6}$$

where N_c is the control horizon and $1 \leq N_c \leq N_P$, which denotes component-wise inequality.

The differences between the predictive states and the reference trajectory X_{ref} are defined as follows

$$\begin{aligned}
e(k+1|t) &= \mathrm{X}(k+1|t) - \mathrm{X}_{ref}(k+1|t),\\
&\vdots\\
e(k+N_P|t) &= \mathrm{X}(k+N_P|t) - \mathrm{X}_{ref}(k+N_P|t).
\end{aligned} \tag{7}$$

To ensure the passenger comfort and feasibility of the vehicle, the output control should be varied as smoothly as possible. Therefore, the optimization objective function is defined as

$$J(e(t), \mathrm{U}(t)) = \sum_{i=1}^{N_P} \|e(k+i|t)\|_Q^2 + \sum_{i=1}^{N_c} \|\mathrm{U}(k+i|t) - \mathrm{U}(k+i-1|t)\|_R^2, \tag{8}$$

where Q and R are the weight matrices for the vehicle states and control states, respectively. Consequently, the rolling optimization can be obtained by solving the constrained optimization problem in every sampling period

$$\begin{aligned}
&\min_{\mathrm{U}(t)} J(e(t), \mathrm{U}(t))\\
&\text{s.t.}\\
&a_{\min} \leq a(k+i|t) \leq a_{\max}, \quad i = 1, 2, \cdots, N_c,\\
&\delta_{\min} \leq \delta(k+i|t) \leq \delta_{\max}, \quad i = 1, 2, \cdots, N_c,\\
&e_{\min} \leq e(t) \leq e_{\max}, \qquad\quad t = k+T_s, \cdots, k+N_P T_s,
\end{aligned} \tag{9}$$

where ($a_{\min}$, $a_{\max}$) and ($\delta_{\min}$, $\delta_{\max}$) are the hard constraints of the vehicle. The last constraints are added to ensure safety driving.

The control inputs are obtained by solving the optimization problem (9). The first element in the control inputs is taken as the optimal control at the current time. After the prediction and control of the current time step are completed, the states are updated with the actual ones, which are then used as the initial states for the optimization problem in the next predictive horizon. The process is repeated until the vehicle reaches the final state.

The problem (9) is a quadratic programming (QP) one which is a traditional optimization problem for trajectory tracking. The first term in the cost function requires that the actual trajectory be as close as possible to the reference trajectory to ensure the safety and feasibility of the trajectory. The second term requires that the control input be varied smoothly to ensure the feasibility of the vehicle and the comfort of passengers. The difference between the reference and the actual trajectory must be sufficiently small. Otherwise, it may lead to a crash, and the trajectory is no longer feasible.

To meet the real-time requirement, instead of directly calculating a control sequence by solving (9), we solve an approximate optimization problem by imposing the constraints $u_{k+1} = u_{k+2} = \ldots = u_{k+Nc}$ on the control law. Therefore, we only need to calculate a 'mediocre' control to follow the given trajectory. This significantly reduces the complexity

of the primal problem as it dramatically reduces the number of the variables. It is worth noting that imposing the constraint conditions $u_{k+1} = \ldots = u_{k+Nc}$ on (9) is equivalent to setting N_c to 1. Besides greatly reducing the computational burden, one of the most telling advantages of structuralizing the control is to produce an improvement in the robustness and in the general behavior of the system, because allowing the free evolution of the manipulated variables could lead to undesirable high-frequency control signals and even to instability as noted in [30]. We note that, if the coefficient matrices Q and R are positive semi-definite, the primal problem is tightly bounded and the approximate problem is also tightly bounded. Since Q and R are positive semi-definite, ${x_1}^T Q x_1 \geq 0$, ${x_2}^T R x_2 \geq 0$. Hence, the cost function in (9) is convex. The feasible region subjected to the constraint conditions (linear equations and inequalities) in (9) is also convex. Thus, the optimal solution of (9) is located in either the interior or the boundary of the feasible region. Therefore, the value of the cost function does not go to infinity, and the primal problem is tightly bounded. When imposing the constraint condition $u_{k+1} = \ldots = u_{k+Nc}$, the corresponding feasible region is still convex since the intersection of convex sets is still a convex set. Similarly, the approximate problem is tightly bounded. In the next section, we prove that the proposed close-loop MPC controller is stable if $N_c = N_1 = 1$, $\lambda = 0$ and N_P is large.

3. Stabilizability of Controller

Combining (1) and (5) leads to

$$\begin{aligned} x_{k+1} &= x_k + T_s(v_k + aT_s)\cos(\varphi_k + T_s v_k \sin(\beta)/l_r + \beta), \\ y_{k+1} &= y_k + T_s(v_k + aT_s)\sin(\varphi_k + T_s v_k \sin(\beta)/l_r + \beta), \\ \varphi_{k+1} &= \varphi_k + T_s(v_k + aT_s)\sin(\beta)/l_r, \\ v_{k+1} &= v_k + aT_s. \end{aligned} \tag{10}$$

Equation system (10) can be rewritten as

$$\begin{aligned} X_{k+1} &= (I + A_k T_s)X_k + B_k T_s U_k, \\ A_k &= \begin{bmatrix} 0 & 0 & -(v_k + aT_s)\sin(\gamma) & \cos(\gamma) - (v_k + aT_s)T_s \sin(\gamma)\sin(\beta)/l_r \\ 0 & 0 & (v_k + aT_s)\cos(\gamma) & \sin(\gamma) + (v_k + aT_s)T_s \cos(\gamma)\sin(\beta)/l_r \\ 0 & 0 & 0 & \sin(\beta)/l_r \\ 0 & 0 & 0 & 0 \end{bmatrix}, \\ B_k &= \begin{bmatrix} T_s\cos(\gamma) & -(v_k + aT_s)\sin(\gamma)(1 + v_k T_s \cos(\beta)/l_r)\beta_\delta \\ T_s\sin(\gamma) & (v_k + aT_s)\cos(\gamma)(1 + v_k T_s \cos(\beta)/l_r)\beta_\delta \\ T_s\sin(\beta)/l_r & (v_k + aT_s)\cos(\beta)\beta_\delta/l_r \\ 1 & 0 \end{bmatrix}, \\ \gamma &= \varphi_k + T_s v_k \sin(\beta)/l_r + \beta, \quad \beta_\delta = \frac{l_r l}{l^2\cos^2(\delta) + l_r^2 \sin^2(\delta)}, \quad l = l_f + l_r. \end{aligned} \tag{11}$$

For the sake of convenience, we omit the subscript k in the remainder of this paper.

Theorem 1. *System (11) is controllable.*

Proof of Theorem 1. First, we seek the eigenvalues λ of A. By solving the characteristic polynomial $\det(\lambda I\text{-}A) = 0$, we have

$$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0. \tag{12}$$

According to the definition of controllability proposed by Hautus [31], system (4) is controllable if and only if, for all λ_i, i = 1, 2, 3, 4 Rank([$\lambda_i I$-A,B]) = 4. Here we only need to consider Rank([$\lambda_1 I$-A,B]) due to (12)

$$[\lambda_1 I - A, B] = \begin{bmatrix} 0 & 0 & -A_{13} & -A_{14} & B_{11} & B_{12} \\ 0 & 0 & -A_{23} & -A_{24} & B_{21} & B_{22} \\ 0 & 0 & -A_{33} & -A_{34} & B_{31} & B_{32} \\ 0 & 0 & -A_{43} & -A_{44} & B_{41} & B_{42} \end{bmatrix} = [0_{4\times 2}, \Omega_{4\times 4}]. \tag{13}$$

The determinant of Ω is

$$\det(\Omega) = \frac{(v_k + aT_s)^2 l \cos(\beta)}{l^2 \cos^2(\delta) + l_r^2 \sin^2(\delta)} \neq 0. \tag{14}$$

Hence we have Rank(Ω) = 4 and 4 $\geq$ Rank([$\lambda_1 I$-A,B]) $\geq$ Rank(Ω) = 4. □

Theorem 2. *System (11) is observable.*

Proof of Theorem 2. According to the definition of observability proposed by Hautus [31], system (4) is observable if and only if, for all λ_i, i = 1, 2, 3, 4 Rank([$\lambda_i I$-A;C]) = 4.

In our problem, the output function is $Y = X = CX$, and thus $C = I_{4\times 4}$ and Rank(C) = 4. Therefore, 4 $\geq$ Rank([$\lambda_1 I$-A;C]) $\geq$ Rank(C) = 4. □

Theorem 3. *System (11) is stabilizable.*

Proof of Theorem 3. According to the definition of stabilizability proposed by Hautus [31], system (11) is stabilizable if and only if $\lambda_i \geq 0$, i = 1, 2, 3, 4, and the system is controllable. Combining Theorem 1 and (12) proves that Theorem 3 holds. □

Theorem 4. *The closed-loop MPC controller is stable for N_c = 1, λ = 0 and large N_P.*

Proof of Theorem 4. This proof is similar as that for Theorem 4 in [32] for generalized predictive control. When N_P is sufficiently large, we have

$$G^T G > 0, \tag{15}$$

where

$$G = \left[B; AB; \cdots; A^{N_P-1}B\right]_{N_P\times N_c}. \tag{16}$$

Therefore, $G^T G$ is a positive scalar, which is always invertible. Therefore, the matrix $G^T G + \lambda I$ is invertible, and a feasible control can be obtained using the expression in [32]

$$u_{opt} = \left(G^T G + \lambda I\right)^{-1} G^T (X_r - X). \tag{17}$$

Since our optimization problem is convex, there is only one optimal solution and thus our controller will asymptotically converge to (17). □

4. Simulation

The simulation environment is MATLAB/Simulink R2020a, and (9) is solved using 'fmincon', a built-in function in MATLAB. Sequential quadratic programming is used as the nonlinear solver. The warm-start technique is employed by using the result of the previous optimization problem as a guess for the current optimization problem to further speed up the efficiency of the nonlinear solver. The accuracy of 'fmincon' is set to 10^{-6}. The processor used in the simulation is Intel(R) Core(TM) i7-4510U @ 2.00 GHz 2.6 GHz.

Real-Time Synchronization is enabled to test the real-time performance of the controllers. The simulation system consists of a kinematic model of autonomous vehicles and the trajectory tracking controller proposed in this paper. The parameters of the vehicle model and the controller are shown in Table 1 and can be found in [33]. The road conditions are assumed to be dry and clean and they can support the forces required for braking, accelerating and steering.

Table 1. Parameters of the vehicle and the controller.

Parameter	Value
l_f	1.232 m
l_r	1.468 m
Range of a	$[-1\ m/s^2, 1\ m/s^2]$
Range of δ	[−0.44 rad, 0.44 rad]
Range of lateral error	[−0.5 m, 0.5 m]
N_P	15
Nc	1
Q	$100I$ [1]
R	I

[1] I is the unit matrix.

4.1. Sinusodial Path Following

First, we present the tracking results of a sinusoidal trajectory with an amplitude of 4 m and a wavelength of 100 m in [20]. The reference speed along x-axis $Vref$ is set to be a constant. The open-loop reference trajectory is given by

$$Yref = 4\sin(2\pi Xref/100). \tag{18}$$

The tracking result of the sinusoidal trajectory is shown in Figure 4. The sampling time is set to T_s = 0.05 s. The reference trajectory was indicated by the black solid line. The obtained trajectories using the forward and backward Euler method were represented by a blue dotted line and a red dashed line, respectively. When the reference velocity is set to $Vref$ = 40 km/h, the maximum lateral error using the backward Euler method was 0.0767 m, in contrast to 0.2481 m using the forward Euler method. The maximum longitudinal errors using the forward and backward Euler method were 0.07 m and 0.0703 m, respectively. The maximum calculation time using the backward Euler method was 0.0203 s, and the average calculation time was 0.0081 s, in contrast to 0.0197 s and 0.01 s using the forward Euler method. The maximum heading errors were 0.0277 rad using the backward Euler method and 0.019 rad using the forward Euler method. When the reference velocity is set to $Vref$ = 60 km/h, the maximum lateral error using the Euler method was 0.4191 m; whereas, it was 0.2184 m using the backward Euler method. The maximum calculation times using the forward and backward Euler method were 0.0183 s and 0.0143 s, respectively; the average computation times were 0.0086 s and 0.0084 s; the maximum heading errors were 0.0293 rad and 0.0355 rad. The maximum longitudinal errors are 0.1059 m and 0.1085 m. To sum up, the lateral error using the backward Euler method was much smaller than that using the forward Euler method. However, the longitudinal error and heading error using the backward Euler method were slightly larger than that using the forward Euler method. Besides that, the backward Euler method required a little more calculation time. The state errors, including the lateral, longitudinal and heading errors, increased with the reference velocity.

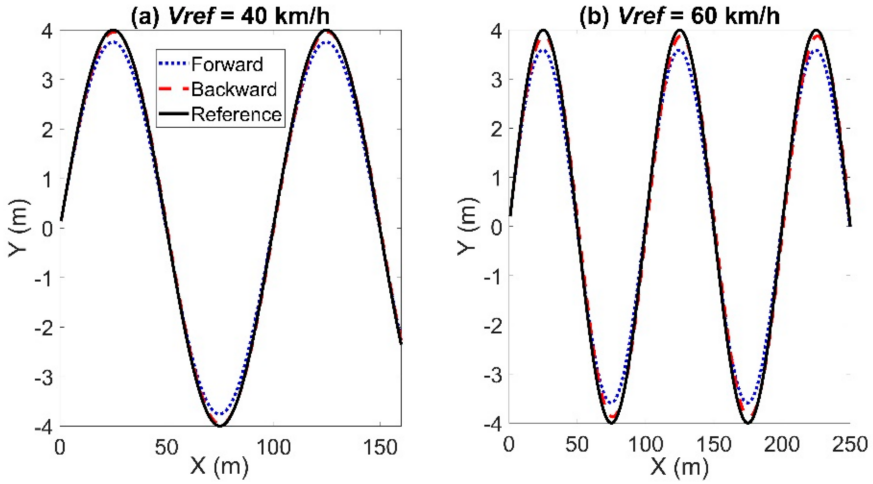

Figure 4. Results for tracking the sinusoidal trajectory with T_s = 0.05 s: left for Case (**a**) with *Vref* = 40 km/h and right for Case (**b**) with *Vref* = 60 km/h.

The comparison of the calculation time between the two controllers is shown in Figure 5. The calculation time of the MPC controller using the backward Euler method at each control period was almost the same or slightly larger than that of the MPC-based controller using the forward Euler method.

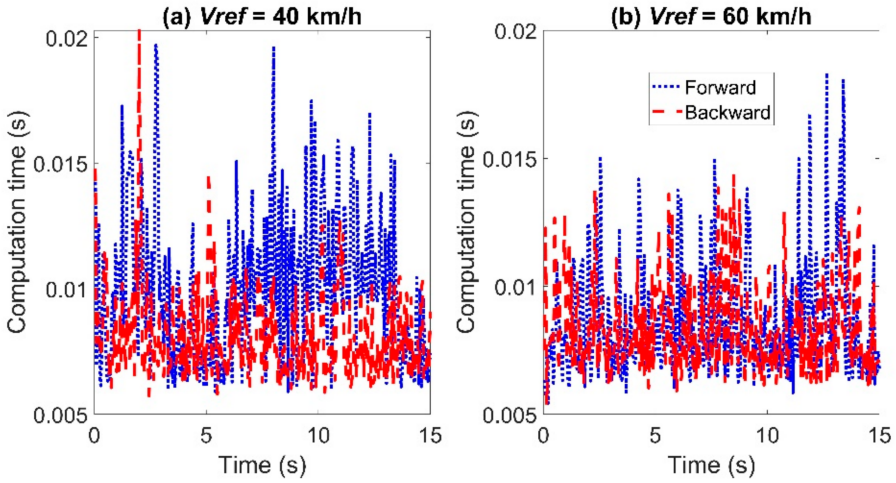

Figure 5. Comparison of the computation time for tracking the sinusoidal trajectory with T_s = 0.05 s: left for Case (**a**) with *Vref* = 40 km/h and right for Case (**b**) with *Vref* = 60 km/h.

Figures 6 and 7 show the articulated acceleration and steer angle, respectively. The Forward Euler method was more sensitive to the longitudinal velocity, whereas the backward Euler method was more sensitive to the steer angle.

We noted that, as mentioned before, the differences between the reference and actual trajectories increase with the vehicle velocity. There exists a threshold value of velocity to determine the existence of the solution of the optimization problem for trajectory tracking. In other words, when the reference velocity is greater than some value, no feasible solution exists. When T_s = 0.05 s, the threshold value of the reference velocity using the forward Euler method was 67.7 km/h (when *Vref* = 67.8 km/h, the maximum lateral error was 0.5009 m), whereas it was 83 km/h using the backward Euler method (when *Vref* = 83.1 km/h, the maximum lateral error was 0.5002 m). Hence, the MPC using the backward Euler method was more robust than that using the forward Euler method.

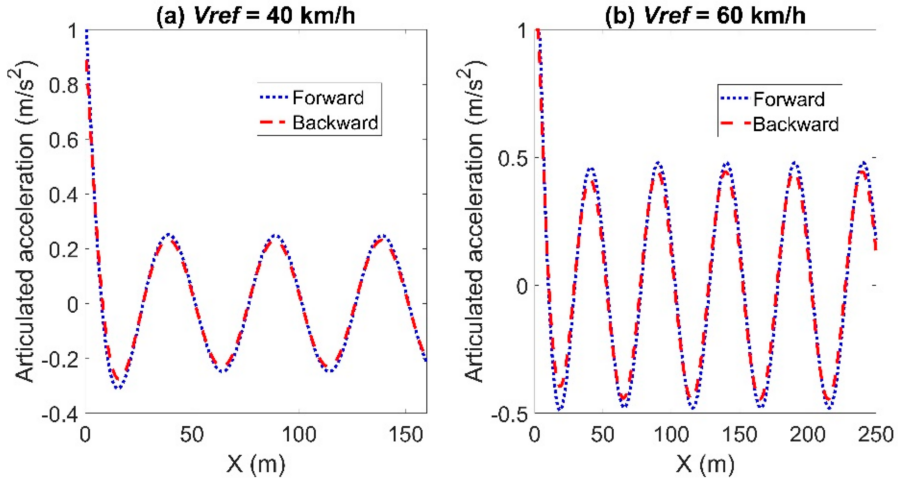

Figure 6. Articulated acceleration for the comparison between the forward and backward Euler method: left for Case (**a**) with *Vref* = 40 km/h and right for Case (**b**) with *Vref* = 60 km/h.

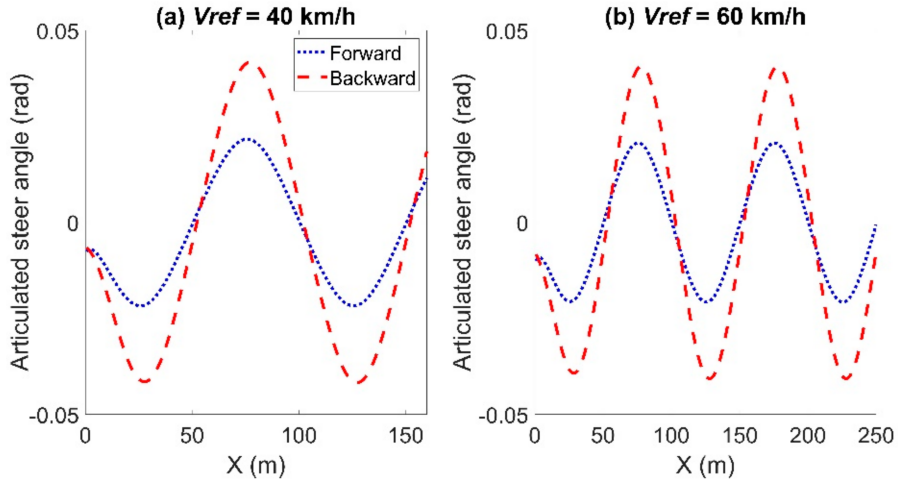

Figure 7. Articulated steer angle for the comparison between the forward and backward Euler method: left for Case (**a**) with *Vref* = 40 km/h and right for Case (**b**) with *Vref* = 60 km/h.

4.2. Circular Path

In the second scenario, the vehicle was required to track a circle with a radius of 40 m. The parametric equations for the circle were

$$\begin{cases} X(t) = Rd\cos(\varphi(t) - \pi/2), \\ Y(t) = Rd + Rd\sin(\varphi(t) - \pi/2), \\ \varphi(t) = \ t\ Vref/Rd, \end{cases} \tag{19}$$

where Rd is the radius of the reference circle. The initial configuration and constraint conditions were chosen to be same as previously to be consistent. The sampling time and the reference velocity are set to T_s = 0.05 s and *Vref* = 10 m/s, respectively.

The tracking result of the circular path is shown in Figure 8. The maximum lateral and longitudinal errors using the backward Euler method were 0.0596 m and 0.0091 m, in contrast to 0.3664 m and 0.3664 m using the forward Euler method. The maximum calculation time using the backward Euler method was 0.016 s, and the average calculation time was 0.0084 s, in contrast to 0.0199 s and 0.0085 s using the forward Euler method. The maximum heading errors were 0.0411 rad using the backward Euler method and 0.0194 rad using the forward Euler method. In sum, the MPC controller using the backward Euler

method had a better tracking accuracy in the circular path than that using the forward Euler method.

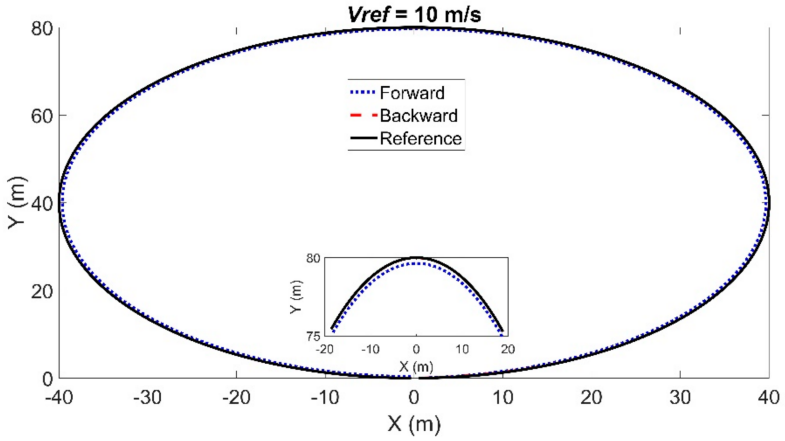

Figure 8. Simulation results for the circular path with radius Rd = 40 m. Notations the same as in Figure 4.

Figures 9 and 10 show the articulated acceleration and steer angle, respectively. The articulated acceleration and steer angle using the backward Euler method were quite different from those using the forward Euler method. As can be seen from Figure 8, the backward Euler method was more accurate than the forward Euler method, and the calculation times were almost the same as shown in Figure 11.

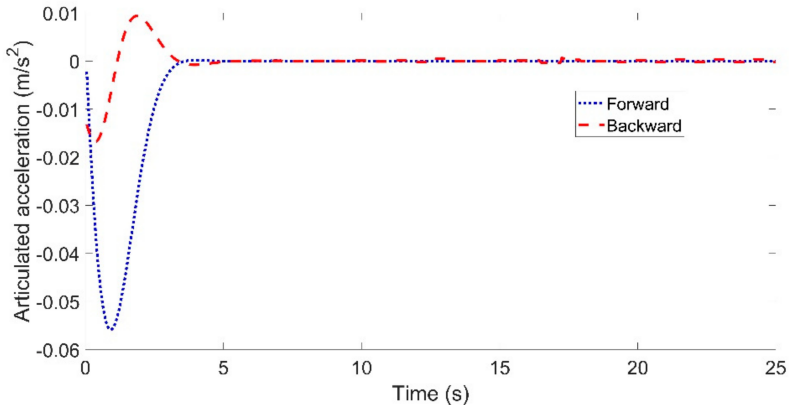

Figure 9. Articulated acceleration for the comparison between the forward and backward Euler method for the circular trajectory.

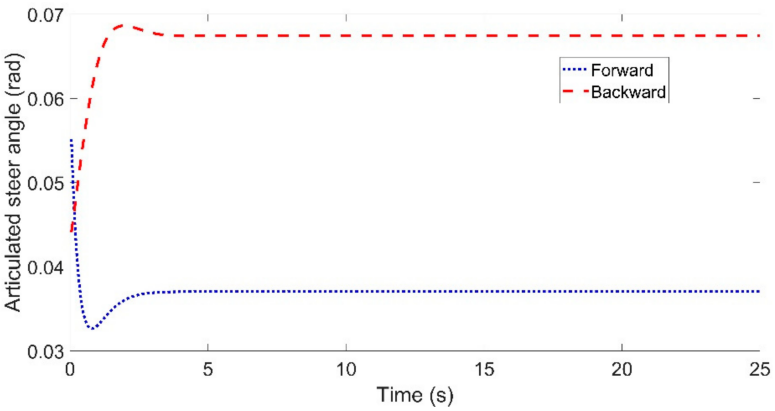

Figure 10. Articulated steer angle for the comparison between the forward and backward Euler method for the circular trajectory.

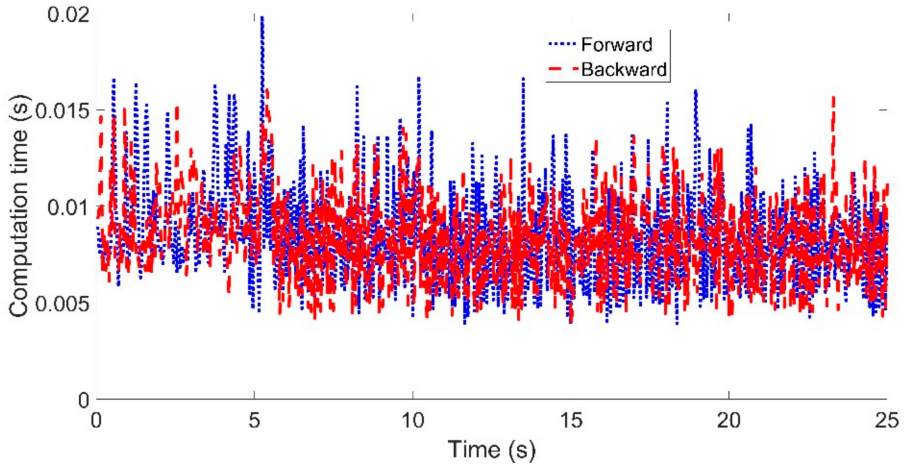

Figure 11. Comparison of the computation time for the circular path. Red dashed curve: MPC-based controller using the backward Euler method; blue dotted curve: MPC-based controller using the forward Euler method.

4.3. Double Line Change Path

In this scenario, the vehicle was required to track a double line change path. The reference trajectory of the double line change path can be found in [33]. The tracking result of the double line change path is shown in Figure 12. When the reference velocity is set to *Vref* = 40 km/h, the maximum lateral and longitudinal errors using the backward Euler method were 0.3034 m and 0.0203 m, in contrast to 0.3827 m and 0.0412 m using the forward Euler method. The maximum heading errors were 0.0673 rad using the backward Euler method and 0.0648 rad using the forward Euler method. When the reference velocity is set to *Vref* = 60 km/h, the maximum lateral and longitudinal errors using the backward Euler method were 0.587 m and 0.0504 m, in contrast to 0.6187 m and 0.0311 m using the forward Euler method. The maximum heading errors were 0.1035 rad using the backward Euler method and 0.0967 rad using the forward Euler method. In sum, the MPC controller using the backward Euler method had a better tracking accuracy in the circular path than that using the forward Euler method. Figures 13 and 14 show the articulated acceleration and steer angle, respectively.

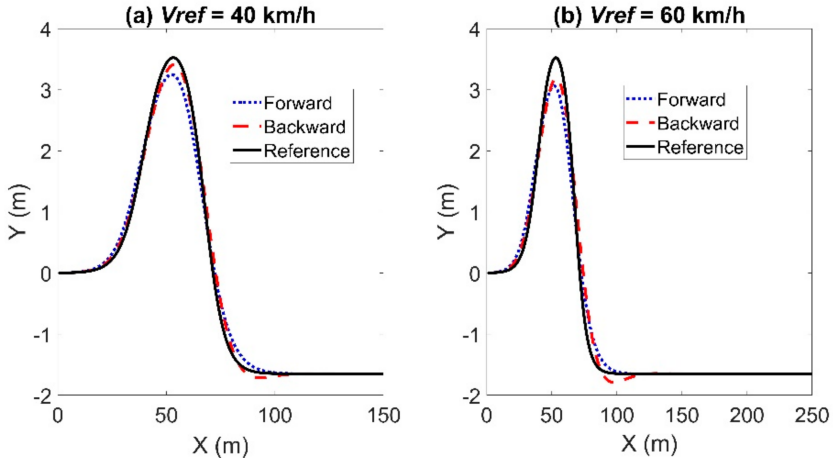

Figure 12. Simulation results for the double line change path: left for Case (**a**) with *Vref* = 40 km/h and right for Case (**b**) with *Vref* = 60 km/h.

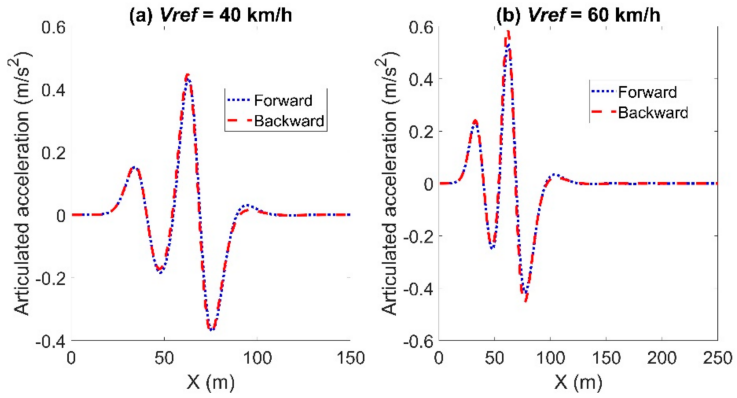

Figure 13. Articulated acceleration for the comparison between the forward and backward Euler method for the circular trajectory: left for Case (**a**) with *Vref* = 40 km/h and right for Case (**b**) with *Vref* = 60 km/h.

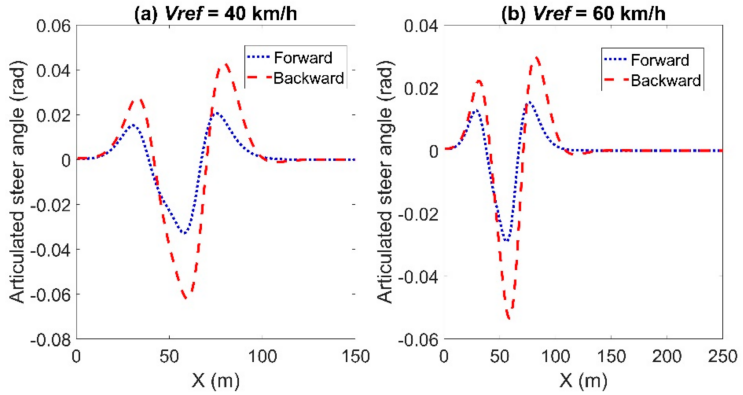

Figure 14. Articulated steer angle for the comparison between the forward and backward Euler method for the circular trajectory: left for Case (**a**) with *Vref* = 40 km/h and right for Case (**b**) with *Vref* = 60 km/h.

The comparison of the calculation time between the two controllers is shown in Figure 15. When the reference velocity is set to *Vref* = 40 km/h, the maximum calculation time using the backward Euler method was 0.0178 s, and the average calculation time was 0.0084 s, in contrast to 0.0157 s and 0.0084 s using the forward Euler method. When the reference velocity is set to *Vref* = 60 km/h, the maximum calculation time using the backward Euler method was 0.0152 s and the average calculation time was 0.008 s, in contrast to 0.0169 s and 0.0083 s using the forward Euler method.

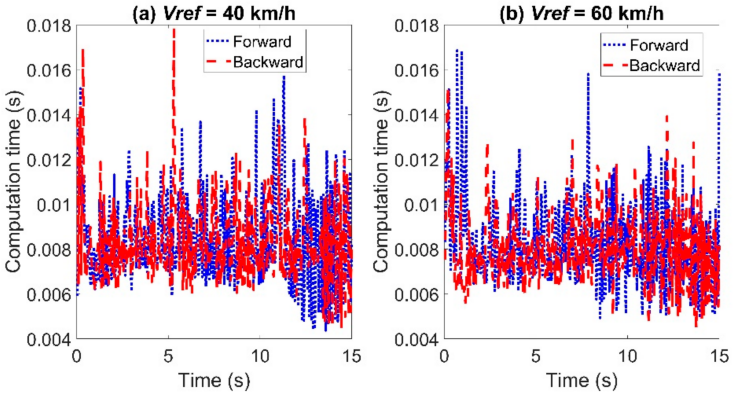

Figure 15. Comparison of the computation time for tracking the double line change path with T_s = 0.05 s: left for Case (**a**) with *Vref* = 40 km/h and right for Case (**b**) with *Vref* = 60 km/h.

Through three sets of comparisons, we can draw some conclusions. First, the lateral tracking errors using the backward Euler method were much smaller than those when using the forward Euler method. Second, the lateral tracking errors using either the forward or backward Euler method increased with the reference velocity. Third, the calculation times using the backward Euler method were almost the same with that using the forward Euler method. Lastly, compared with the articulated acceleration, there was a clear discrepancy in the articulated steer angle.

5. Conclusions

An effective and efficient method for generating a feasible trajectory is of vital importance to meet the requirement of instantaneous control for autonomous driving. In this paper, we have proposed a trajectory tracking controller based on MPC. Most MPC-based and other methods either set the velocity to a constant or cannot actively adjust the longitudinal velocity according to the information of the reference trajectory. To solve this problem, both the acceleration and steer angle are set to control inputs. Hence, the proposed controller can automatically adjust the velocity according to the information of the reference trajectory. Moreover, instead of the forward Euler integration method, the backward Euler integration method is used to establish the predictive model. To meet the real-time requirement, we impose the constraints $u_{k+1} = \ldots = u_{k+Nc}$ on the control law. This significantly reduced the problem complexity. The warm-start technique was used to further accelerate the convergence of the optimization solver of the controller by using the previous results as a guess for the current optimization problem.

The proposed closed-loop MPC controller was stable and validated by simulation experiments. Compared with the MPC controller using the forward Euler method, the MPC controller using the backward Euler method had a much better accuracy in the lateral error, which is an important indicator to ensure driving safety. The lateral error could be reduced by up to 78%. There is little difference in the longitudinal error between the two controllers. However, the heading error of the MPC controller using the backward Euler method was larger than that of the MPC controller using the forward Euler method. The maximum and average computation times using the backward Euler method were almost the same or slightly larger than those using the forward Euler method. Moreover, the MPC controller using backward Euler method was more robust than that using the forward Euler method. The threshold value of the velocity for the MPC controller using the backward Euler method was larger than that using the forward Euler method (83 km/h versus 67.7 km/h for the sinusoidal trajectory). Overall, the MPC controller using the backward Euler method had a better tracking accuracy at the cost of no or little computation time.

The existence of the discrepancy between the actual trajectory and the reference trajectory is mainly due to the modelling errors, computation errors and disturbance errors. Recent studies on MPC-based controllers mainly focus on the modelling errors and disturbance errors. Few authors investigated the computation errors during the discretization for nonlinear systems. We hope that this paper is instructive and allows researchers new insight into creating MPC-based controllers.

From the perspective of science, our contribution is to give a new way to establish a predictive model, which is a cornerstone of designing a path tracking controller. Besides the forward and backward Euler methods, there are several integration methods, such as the midpoint method and Runge-Kutta methods. Improving the accuracy of the prediction model using other integration methods could be a promising way to get an effective control to maintain a good tracking accuracy.

Author Contributions: Conceptualization, Z.H. and H.L.; methodology, Z.H.; software, Z.Y.; validation, W.L., J.L., C.H. and W.F.; formal analysis, Z.H., W.L., C.H., J.L. and W.F.; investigation, Z.H.; writing—original draft preparation, Z.H.; writing—review and editing, H.L.; supervision, H.L.; project administration, H.L.; funding acquisition, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, Grant number No. 62003328 and the China Postdoctoral Science Foundation, Grant No. 2020M682985.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank for support from National Natural Science Foundation of China and the China Postdoctoral Science Foundation.

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

References

1. Li, L.; Song, J.; Wang, F.Y.; Niehsen, W.; Zheng, N.N. IVS 05: New developments and research trends for intelligent vehicles. *IEEE Intell. Syst.* **2005**, *20*, 10–14. [CrossRef]
2. Ma, Y.; Wang, Z.; Yang, H.; Yang, L. Artificial intelligence applications in the development of autonomous vehicles: A survey. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 315–329. [CrossRef]
3. Paden, B.; Cap, M.; Yong, S.Z.; Yershov, D.; Frazzoli, E. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans. Intell. Veh.* **2016**, *1*, 33–55. [CrossRef]
4. Cafiso, S.; Pappalardo, G. Safety effectiveness and performance of lane support systems for driving assistance and automation—Experimental test and logistic regression for rare events. *Accid. Anal. Prev.* **2020**, *148*, 105791. [CrossRef]
5. Pappalardo, G.; Cafiso, S.; Di Graziano, A.; Severino, A. Decision Tree Method to Analyze the Performance of Lane Support Systems. *Sustainability* **2021**, *13*, 846. [CrossRef]
6. Badue, C.; Guidolini, R.; Carneiro, R.V.; Azevedo, P.; Cardoso, V.B.; Forechi, A.; Jesus, L.; Berriel, R.; Paixão, T.; Mutz, F.; et al. Self-driving cars: A survey. *Expert Syst. Appl.* **2021**, *165*, 113816. [CrossRef]
7. Li, B.; Shao, Z. A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles. *Knowl.-Based Syst.* **2015**, *86*, 11–20. [CrossRef]
8. Zhang, Z.; Zhang, L.; Deng, J.; Wang, M.; Wang, Z.; Cao, D. An Enabling Trajectory Planning Scheme for Lane Change Collision Avoidance on Highways. *IEEE Trans. Intell. Veh.* **2021**. [CrossRef]
9. Zhang, L.; Wang, Z.; Ding, X.; Li, S.; Wang, Z. Fault-Tolerant Control for Intelligent Electrified Vehicles Against Front Wheel Steering Angle Sensor Faults During Trajectory Tracking. *IEEE Access* **2021**, *9*, 65174–65186. [CrossRef]
10. Zhang, L.; Zhang, Z.; Wang, Z.; Deng, J.; Dorrel, D.G. Chassis Coordinated Control for Full X-by-Wire Vehicles—A Review. *Chin. J. Mech. Eng.* **2021**, *34*, 42. [CrossRef]
11. Fabiani, F.; Grammatico, S. Multi-vehicle automated driving as a generalized mixed-integer potential game. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 1064–1073. [CrossRef]
12. Coulter, R.C. *Implementation of the Pure Pursuit Path Tracking Algorithm*; Technical Report; Carnegie-Mellon UNIV Pittsburgh PA Robotics INST: Pittsburgh, PA, USA, 1992.
13. Thrun, S.; Montemerlo, M.; Dahlkamp, H.; Stavens, D.; Aron, A.; Diebel, J.; Fong, P.; Gale, J.; Halpenny, M.; Hoffmann, G.; et al. Stanley: The Robot That Won the DARPA Grand Challenge. *J. Field Robot.* **2006**, *23*, 661–692. [CrossRef]
14. Amidi, O.; Thorpe, C.E. Integrated mobile robot control. *Mob. Robot. V* **1991**, *1388*, 504–523.
15. Dixit, S.; Fallah, S.; Montanaro, U.; Dianati, M.; Stevens, A.; Mccullough, F.; Mouzakitis, A. Trajectory planning and tracking for autonomous overtaking: State-of-the-art and future prospects. *Annu. Rev. Control* **2018**, *45*, 76–86. [CrossRef]
16. Araki, M. Control Systems, Robotics, and Automation—Vol. II—PID Control. In *Encyclopedia of Life Support Systems*; EOLSS Publishers Ltd.: London, UK, 2009.
17. Young, K.D.; Utkin, V.I.; Ozguner, U. A control engineer's guide to sliding mode control. *IEEE Trans. Control Syst. Technol.* **1999**, *7*, 328–342. [CrossRef]
18. Utkin, V.; Lee, H. Chattering problem in sliding model control systems. In Proceedings of the 2nd IFAC Conference on Analysis and Design of Hybrid Systems, Alghero, Italy, 7–9 June 2006.
19. Amer, N.; Zamzuri, H.; Hudha, K.; Hudha, K.; Kadir, Z. Modelling and control strategies in path tracking control for autonomous ground vehicles: A review of state of the art and challenges. *J. Intell. Robot. Syst.* **2017**, *86*, 225–254. [CrossRef]
20. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef]
21. Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Matthew, L.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *Science* **2018**, *362*, 1140–1144. [CrossRef]
22. Kendall, A.; Hawke, J.; Janz, D.; Mazur, P.; Reda, D.; Allen, J.M.; Lam, V.D.; Bewley, A.; Shah, A. Learning to drive in a day. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 2–6 May 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 8248–8254.

23. Mohammadi, M.; Arefi, M.M.; Setoodeh, P.; Kaynak, O. Optimal tracking control based on reinforcement learning value iteration algorithm for time-delayed nonlinear systems with external disturbances and input constraints. *Inf. Sci.* **2021**, *554*, 84–98. [CrossRef]
24. Shen, C.; Shi, Y.; Buckham, B. Integrated path planning and tracking control of an AUV: A unified receding horizon optimization approach. *IEEE/ASME Trans. Mechatron.* **2016**, *22*, 1163–1173. [CrossRef]
25. Borrelli, F.; Falcone, P.; Keviczky, T.; Asgari, J.; Hrovat, D. MPC-based approach to active steering for autonomous vehicle systems. *Int. J. Veh. Auton. Syst.* **2005**, *3*, 265–291. [CrossRef]
26. Kong, J.; Pfeiffer, M.; Schildbach, G.; Borrelli, F. Kinematic and dynamic vehicle models for autonomous driving control design. In Proceedings of the 2015 IEEE Intelligent Vehicles Symposium (IV), Seoul, Korea, 28 June–1 July 2015; IEEE: Piscataway, NJ, USA, 2016; pp. 1094–1099.
27. Polack, P.; Altché, F.; d'Andréa-Novel, B.; de La Fortelle, A. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In Proceedings of the 2017 IEEE intelligent vehicles symposium (IV), Redondo Beach, CA, USA, 11–14 June 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 812–818.
28. Chen, S.; Chen, H.; Negrut, D. Implementation of MPC-Based Path Tracking for Autonomous Vehicles Considering Three Vehicle Dynamics Models with Different Fidelities. *Automot. Innov.* **2020**, *3*, 386–399. [CrossRef]
29. Rajamani, R. *Vehicle Dynamics and Control*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2011.
30. Camacho, E.F.; Brodons, C. *Model Predictive Control*; Springer: London, UK, 1999.
31. Hautus, M. Stabilization controllability and observability of linear autonomous systems. *Indag. Math. (Proc.)* **1970**, *73*, 448–455. [CrossRef]
32. Clarke, D.W.; Mohtadi, C. Properties of generalized predictive control. *Automatica* **1989**, *25*, 859–875. [CrossRef]
33. Gong, J.W.; Jiang, Y.; Xu, W. *Model Predictive Control for Self-Driving Vehicles*; Beijing Institute of Technology Press: Beijing, China, 2014.