IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 13, 2021, accepted May 16, 2021, date of publication June 14, 2021, date of current version June 22, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3088937

Model Predictive Control Method for Autonomous Vehicles Using Time-Varying and Non-Uniformly Spaced Horizon

MINSUNG KIM[1], DONGGIL LEE[2], (Graduate Student Member, IEEE), JOONWOO AHN[1], MINSOO KIM[1], AND JAEHEUNG PARK[1,3], (Member, IEEE)

[1]Graduate School of Convergence Science and Technology, Seoul National University, Seoul 08826, South Korea
[2]Department of Electrical and Computer Engineering, ASRI, Seoul National University, Seoul 08826, South Korea
[3]Advanced Institute of Convergence Technology (AICT), Suwon 16229, South Korea

Corresponding author: Jaeheung Park (park73@snu.ac.kr)

ABSTRACT This paper proposes an algorithm for path-following and collision avoidance of an autonomous vehicle based on model predictive control (MPC) using time-varying and non-uniformly spaced horizon. The MPC based on non-uniformly spaced horizon approach uses the time intervals that are small for the near future, and time intervals that are large for the distant future, to extend the length of the whole prediction horizon with a fixed number of prediction steps. This MPC has the advantage of being able to detect obstacles in advance because it can see the distant future. However, the presence of longer time interval samples may lead to poor path-following performance, especially for paths with high curvature. The proposed algorithm performs proper adjustment of the prediction interval according to a given situation. For sections with large curvature, it uses the short prediction intervals to increase the path-following performance; further, to consider obstacles over a wider range, it uses the long prediction intervals. This technique allows simultaneous improvement of the path-following performance and the range of obstacle avoidance with fixed computational complexity. The effectiveness of the proposed method is verified through an open-source simulator, CARLA and real-time experiments.

INDEX TERMS Path following, model predictive control, autonomous vehicle, collision avoidance.

I. INTRODUCTION

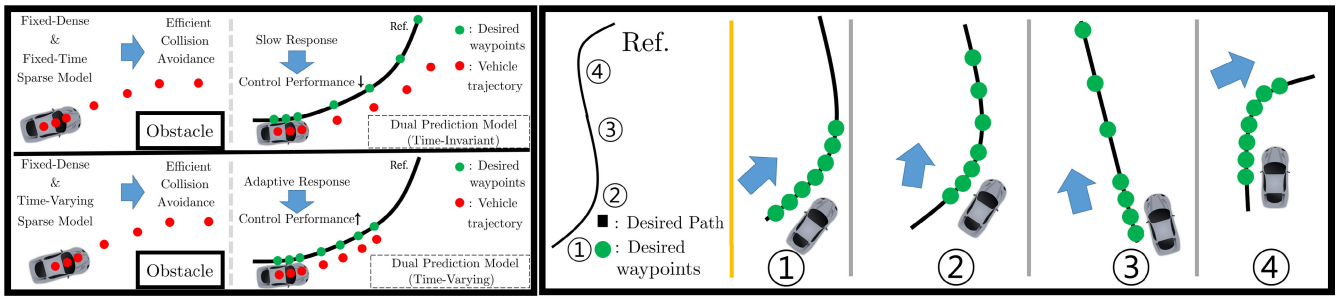
Path-following and collision avoidance for autonomous vehicles have been extensively studied over the past decades considering highway environments and unstructured environments such as parking lots [1], [2].

Most of the proposed controllers utilized an approach of independently designing a path-planning and a path-tracking algorithm [3]–[6] or integrating path-planning and path-tracking algorithms [2], [7], [8]. Unlike the former approach, the latter approach uses nearby sensor information, which is computationally efficient and can be implemented in real-time applications. In addition, it can be used for re-planning to react promptly to environmental changes and/or no a priori known obstacles.

Among these integrated approaches, the method using model predictive control (MPC) has many advantages, such as considering multivariate cost functions and handling multiple constraints directly [9]. Recently, studies using the MPC with a non-uniformly spaced horizon (i.e., this is to discretize the prediction horizon with nodes densely placed near the current time to capture immediate dynamics properly, and with nodes sparsely placed for the rest of horizon to approximate the overall.) were conducted to use limited computational costs efficiently [10]–[12]. These studies attempted to address the limitation of the standard MPC, which requires a large number of prediction steps to incorporate future environmental information using only short time steps. However, when using this method to follow the desired path with high curvature, the control input must be densely obtained over the entire predicted horizon. However, path-following performance can be degraded because of a prediction model with an increased sampling time. Consequently, sampling time of the prediction model has a trade-off relationship between the path-following performance and retaining the prediction horizon length.

The associate editor coordinating the review of this manuscript and approving it for publication was Zheng Chen.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/



(a) Dual prediction model (b) The proposed concept

FIGURE 1. (a)(upper) Although obstacle avoidance is efficiently performed by the MPC with a non-uniformly spaced horizon, the performance of the path-following is degraded in situations with a large path curvature. (lower) Proposed method. (b) The proposed concept: when following the path using the MPC with a non-uniformly spaced horizon, the prediction models need to be variable depending on the situation to prevent the control performance degradation.

Therefore, we proposed a method to efficiently perform path-following and obstacle avoidance while maintaining tracking performance and long prediction horizontal lengths by varying the sampling time of the prediction model. The following settings were considered for the control cycle variations in the prediction model. First, two types of prediction models were constructed, defining the near future as a *Dense* model and the distant future as a *Sparse* model. Varying prediction models over the entire prediction horizon of the MPC may lead to large control input changes. Hence, the prediction models are densely placed near the current time to capture immediate dynamics appropriately, and are sparsely placed for the rest of horizon to approximate the overall trend. Subsequently, the variation is only on the *Sparse* model. Therefore, if the path-following performance is degraded, the sampling time of the *Sparse* model is reduced to improve the control performance of the entire prediction horizon. Likewise, when the performance increases, the sampling time of the *Sparse* model is increased to extend the entire prediction length. This simple idea was used to determine the near-optimal control action at each time interval. Second, the sampling time of the prediction model was set to change by the same or less time as the control period. Third, both the optimal control costs and magnitude of the curvature of the desired path were considered so that model variations were made when needed.

Fig. 1 shows the main concepts of this study. Unlike the approach using two types of fixed dense and sparse models in the MPC (Fig. 1(a), upper), the proposed method (Fig. 1(a), lower) changes the prediction model only in the *Sparse* model. In the case of following a large curvature path, only the *Sparse* model is changed considering both the optimal cost of the MPC problem and the magnitude of the path curvature, as shown in Fig. 1(b) (① → ④). To compare the performance of path-following and collision avoidance, the conventional MPC and DUAL-MPC methods (Fig. 1) using the two types of fixed prediction models were designed.

The main contributions of this study are as follows: A novel MPC method with a time-varying and non-uniformly spaced optimization horizon is proposed to reduce the path-following performance degradation and to avoid obstacles in advance with a reduced computational complexity. Effectiveness of the proposed method was validated using both the simulations and a practical experiment.

A. RELATED STUDIES

In recent years, many studies have been conducted using MPC with a non-uniformly spaced optimization horizon. A predictive framework, including the guidance and control law for soft landing on an asteroid, was proposed in [13]. An efficient and accurate method for controlling an aircraft through model predictive control was introduced in [14]. A method for increasing the efficiency of aircraft vehicle path planners using the direct Receding Horizon Control (RHC) approach was proposed in [15].

Both studies for autonomous vehicles used non-uniform time steps in the prediction horizon of the MPC optimization, thus enabling better integration of stabilization and collision avoidance in a single problem [10], [11]. Combined with advances in computing technology, new algorithms and advanced solvers consistently reduce the computational time required to solve the quadratic programming (QP) problem, enabling MPC to be applied to increasingly fast and higher-dimensional systems. However, there remains a fundamental relationship between the number of prediction steps, the dimensionality of the system, and the resulting QP problem complexity [16]. However, to further mitigate the complexity of the MPC problem, reducing the number of predicted steps can significantly degrade the path-following performance owing to the slow response to changes in the environment. Non-uniformly spaced MPC-based path-following research on autonomous vehicles was demonstrated through a double lane change simulation using dual prediction models [17]. However, this study has only been validated in driving environments with little curvature and is limited to path-following studies.

Meanwhile, in [18], [19], they compared the unconstrained optimal cost of several methods that irregularly divided the prediction horizon using the same length of the prediction horizon. Through these studies, even if a small number of prediction steps are used, effective path following and collision avoidance functions can be performed if the prediction

nodes are well distributed. Therefore, this approach should be further studied to reduce the complexity of the MPC system for efficient real-time application of path-following and obstacle avoidance of autonomous vehicles.

B. ORGANIZATION OF THIS PAPER

The remainder of this paper is organized as follows. Section II presents the basic equations used to model the dynamics of autonomous vehicles and a motivating example. Section III explains the equation of MPC with a non-uniformly spaced optimization horizon, and describes the proposed algorithm. Section IV discusses the simulation results. In Section V, an experimental verification is presented. Section VI draws the conclusions.

II. VEHICLE DYNAMICS AND MOTIVATION

A. VEHICLE LATERAL DYNAMIC MODEL

This subsection introduces the lateral dynamics and error dynamics model of a vehicle. A 2-DOF bicycle model was used as the vehicle model, as shown in Fig. 2. This bicycle model consists of the lateral and yaw dynamics of the body coordinate system. This model is simple, effective, and widely used in vehicle control [20]. Based on the definition of forces applied to tires, the vehicle model can be represented as a nonlinear and linear model. If we use a nonlinear model such as the *Pacejka* model (also known as the *Magic Formula*) [21], the MPC controllers will have a considerably complex burden, which makes it difficult to implement them in real-time. Therefore, a linear model of vehicle dynamics was used in this study.

The force applied to the tire can be expressed linearly based on small slip angles, and the following assumptions (i) Wheels on the same axle are lumped into one wheel located in the center of the front or rear axle (single-track model). (ii) The body weight is evenly distributed on the wheels. (iii) The vehicle drives at a constant speed. (iv) The suspension movement, slippage, and aerodynamic effects were ignored. With these assumptions, the force applied to each tire is written as follows:

$$\begin{aligned} F_{yf} &= 2C_f\alpha_f = 2C_f(\delta - \theta_{vf}) \\ F_{yr} &= 2C_r\alpha_r = 2C_r(-\theta_{vr}) \end{aligned} \tag{1}$$

where F_{yf} and F_{yr} are the lateral tire forces of the front and rear axles, respectively, and C_f and C_r are the cornering stiffness of the front and rear tires, respectively. α_f and α_r are the slip angles of the front and rear tires, respectively, and δ is the steering angle. θ_{vf} and θ_{vr} are the velocity angles of the front and rear tires, respectively. According to *Newton's second law*, the 2-DOF bicycle model is expressed as follows:

$$\begin{aligned} m(\dot{V}_y + \dot{\psi}V_x) &= F_{yf} + F_{yr} \\ I_z\ddot{\psi} &= l_fF_{yf} - l_rF_{yr} \end{aligned} \tag{2}$$

where m denotes the vehicle mass. V_y is the lateral velocity of the vehicle, and $\dot{\psi}$ is the yaw rate. V_x denotes the longitudinal velocity of the vehicle. I_z is the moment of inertia about the z-axis. l_f and l_r are the distances from the center of gravity of the vehicle to the front and rear axles, respectively. Using Eqs. (1) and (2), the combined vehicle lateral dynamics models are expressed, as follows:

$$\begin{aligned} \dot{V}_y &= -\left(\frac{2C_f + 2C_r}{mV_x}\right)V_y - \left(V_x + \frac{2C_fl_f + 2C_rl_r}{mV_x}\right)\dot{\psi} \\ \ddot{\psi} &= -\left(\frac{2C_fl_f - 2C_rl_r}{I_zV_x}\right)V_y \\ &\quad - \left(\frac{2C_fl_f^2 + 2C_rl_r^2}{I_zV_x}\right)\dot{\psi} + \frac{2C_fl_f}{I_z}\delta \end{aligned} \tag{3}$$

To follow the path, the lateral offset error (e_1) is defined based on the shortest distance of the mass center from the desired path, and the orientation (e_2) error is defined as the difference between the angle of the desired path and the vehicle's orientation, as follows:

$$\begin{aligned} \ddot{e}_1 &= \dot{V}_y + V_x(\dot{\psi} - \dot{\psi}_r) \\ e_2 &= \psi - \psi_r \end{aligned} \tag{4}$$

where $\ddot{e}_1$ is the second derivative of e_1. Eqs. (3) and (4) are summarized into one state-space equation, and the

$$\dot{E}(t) = \mathbf{A}E(t) + \mathbf{B}U(t) + \mathbf{B}_r\dot{\psi}_r(t) \tag{5}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\dfrac{2C_f + 2C_r}{mV_x} & \dfrac{2C_f + 2C_r}{m} & -\dfrac{2l_fC_f - 2l_rC_r}{mV_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\dfrac{2l_fC_f - 2l_rC_r}{I_zV_x} & \dfrac{2l_fC_f - 2l_rC_r}{I_z} & -\dfrac{2l_f^2C_f + 2l_r^2C_r}{I_zV_x} \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ \dfrac{2C_f}{m} \\ 0 \\ \dfrac{2l_fC_f}{I_z} \end{bmatrix} \quad \mathbf{B}_r = \begin{bmatrix} 0 \\ -\dfrac{2l_fC_f - 2l_rC_r}{mV_x} - V_x \\ 0 \\ -\dfrac{2l_f^2C_f - 2l_r^2C_r}{I_zV_x} \end{bmatrix} \tag{6}$$

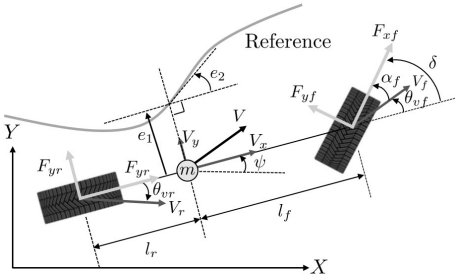


FIGURE 2. Vehicle dynamic model.

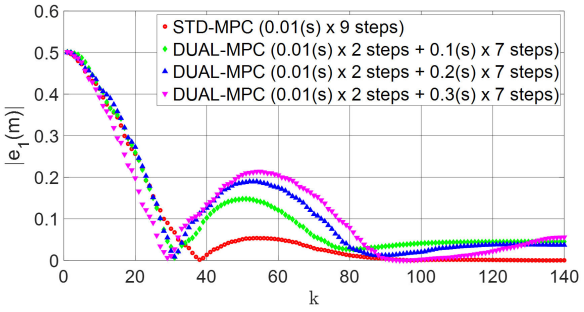


FIGURE 3. The motivating example: path stabilization simulations using four MPC-based controllers. Each controller used the same number of prediction steps (9). STD-MPC used a single model. DUAL-MPC used dual-models (i.e., 0.01(s) × 2 steps + 0.1(s) × 7 steps means two prediction models with the sampling time of 0.01 s, and seven prediction models with the sampling time of 0.1 s).

continuous-time state-space model is written as (5) and (6) shown at the bottom of the previous page. U is the control input (steering input, δ), and $\dot{\psi}_r$ is the desired yaw rate, which is defined using the desired path. And full state feedback is being considered.

where $E = [e_1\ \dot{e}_1\ e_2\ \dot{e}_2]^T$ is the state vector, and $\dot{e}_1$ is the derivative of e_1, and $\dot{e}_2$ is the derivative of the orientation error, e_2.

The physical system is expressed in the continuous-time domain; however, because the controller is implemented in the discrete-time domain, the system must be discretized. The continuous-time model was discretized using the *Tustin* approximation in this study [22]. The resulting discretized vehicle model is as follows:

$$\begin{aligned} E(k+i) &= A_d E(k+(i-1)) \\ &\quad + B_d U(k+(i-1)) + C_d(k+(i-1)) \\ &\quad i = 1, 2, \ldots, N_d \\ E(k+N_d+N_n i) &= A_s E(k+N_d+N_n(i-1)) \\ &\quad + B_s U(k+N_d+N_n(i-1)) \\ &\quad + C_s(k+N_d+N_n(i-1)) \\ &\quad i = 1, 2, \ldots, N_s \end{aligned} \tag{7}$$

where N_d is the number of prediction steps for the *Dense* model, N_s is the number of prediction steps for the *Sparse* model, and N_p denotes the total number of predicted steps. The subscripts d and s indicate the *Dense* and *Sparse* models at each discrete time step k with sampling period Δt, respectively. $(k+i)$ denotes the predicted trajectory at time $k+i$ based on the information available at time k. The *Dense* model represents a discrete model with a sampling time of Δt equal to the control cycle. The *Sparse* model is a model with a longer sampling time of $N_n \Delta t$ (N_n is a constant). $C_d(k)$ and $C_s(k)$ are vectors, each containing the desired yaw rate. The basic idea of the proposed method is to vary the sampling time N_n of the *Sparse* model. The switching criterion for N_n is described in the following sections. The standard MPC refers to an approach based on uniformly spaced optimization, when the value of N_n is 1.

B. MOTIVATING EXAMPLE: CONTROL PERFORMANCE DEGRADATION

The objective of this subsection is to compare the control performance according to methods with different sampling times of the prediction model as described in [18]. Thus, through an example, the control performance degradation is figured out when using dual-fixed prediction models in the MPC.

There are two types of path-following methods based on the MPC. Standard method (STD-MPC) uses a single-fixed model, whereas dual methods (DUAL-MPC) use dual-fixed models, as shown in Fig. 3. Each of the three methods of DUAL-MPC has a different sampling time for the prediction model in the *Sparse* part (0.1, 0.2, and 0.3 s). Fig. 3 represents four simulations of the path following with $Q = (10, 0.01, 0.01, 0.01)$, $R = 0.1$ from initial state $x_0 = [0.5\ 0\ 0\ 0]^T$ with input and input rate constraints. In addition, all methods use identical number of prediction steps, 9. The details of the MPC formulation in this example are described in the following section. The prediction model configuration is shown in each method parentheses in Fig. 3, i.e., 0.01(s) × 2 steps + 0.1(s) × 7 steps means two prediction models with the sampling time of 0.01 s, and seven prediction models with the sampling time of 0.1 s.

As previously described, the method using two prediction models has the advantage of obtaining relatively long prediction lengths to avoid obstacles compared to STD-MPC. However, DUAL-MPC methods decrease control performance compared to methods using short sampling times (0.01 s) as a single-fixed model. Furthermore, among the three methods using DUAL-MPC, path-following performance differs depending on the sampling time of the prediction model of the *Sparse* part (0.1(s)∼0.3(s)), as highlighted in [18].

In terms of path-following and obstacle avoidance with the MPC, long prediction lengths are inevitably required; however, path-following performance degradation is accompanied by the sampling time of the prediction models. In addition, the limitations of using DUAL-MPC are obvious because this phenomenon will appear prominently when the curvature of the desired path is large and varying. Therefore, this simple example highlights the basic reason why the varying prediction model approach needs to be considered while

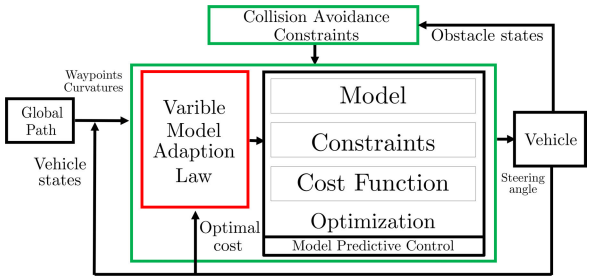


FIGURE 4. Overall architecture for the proposed method: The prediction model is changed by considering the optimal cost of MPC and the magnitude of curvature of the desired path (Variable Model Adaption Law block).

performing path-following and collision avoidance maneuvers with reduced computational complexity.

III. THE PROPOSED METHOD

In this section, time-varying and non-uniformly spaced optimization via an MPC scheme for the vehicle lateral motion control method is presented. The overall structure of the proposed study is illustrated in Fig. 4.

A. MPC FORMULATION

Assuming that all states variables are measurable, the output and input prediction matrices are as follows:

$$\tilde{E}(k) = \begin{bmatrix} E(k+1) \\ \vdots \\ E(k+N_d) \\ E(k+N_d+N_n) \\ \vdots \\ E(k+N_d+N_nN_s) \end{bmatrix}$$

$$\tilde{U}(k) = \begin{bmatrix} U(k) \\ \vdots \\ U(k+N_d-1) \\ U(k+N_d) \\ \vdots \\ U(k+N_d+(N_s-1)N_n) \end{bmatrix} \tag{8}$$

where N_s denotes the number of prediction models that are *Sparse* and variable at the same time. We assumed that the prediction horizon and control horizon were identical to simplify the expression. N_p is the sum of N_d and N_s. Subsequently, the total prediction time is calculated as $(N_d + N_sN_n)\Delta t$. The basic concept of the proposed MPC is to change the value of N_n of the sampling time over the prediction interval. The problem is to obtain an optimal control input sequence $U^*(k)$ for each instance k to minimize the following cost function J:

$$\begin{aligned} J &= \tilde{E}(k)^T\tilde{Q}\tilde{E}(k) + \tilde{U}(k)^T\tilde{R}\tilde{U}(k) \\ &= \sum_{i=1}^{N_d} E(k+i)^T QE(k+i) \\ &\quad + \sum_{i=0}^{N_d-1} U(k+i)^T RU(k+i) \\ &\quad + \sum_{i=1}^{N_s} E(k+N_d+N_ni)^T QE(k+N_d+N_ni) \\ &\quad + \sum_{i=0}^{N_s-1} U(k+N_d+N_ni)^T RU(k+N_d+N_ni) \end{aligned}$$

subject to

$$\begin{aligned} U_{\min} &\le U(k) \le U_{\max} \\ \Delta U_{\min} &\le \Delta U(k) \le \Delta U_{\max} \\ E_{\min} &\le E(k) \le E_{\max} \end{aligned} \tag{9}$$

where $\tilde{Q}$ is diag(Q,...,Q) and $\tilde{R}$ is diag(R,...,R). $U_{\min}$ and $U_{\max}$ represent the minimum and maximum input values, respectively. $E_{\min}$ and $E_{\max}$ denote the minimum and maximum values of the state, respectively, which are used for obstacle avoidance constraints. By expressing the current and future predicted state matrix $\tilde{E}(k)$ in terms of the initial state $E(k)$, the following relationship can be obtained:

$$\tilde{E}(k) = M(k)E(k) + \Phi(k)\tilde{U}(k) + \Lambda(k) \tag{10}$$

where M is a matrix of dimension $n_xN_p \times n_x$ (n_x is the number of states), represented as M_i. Φ is a matrix of dimension $n_xN_p \times n_uN_p$ (n_u is the number of inputs), represented as Φ_{ij}. $\Lambda(k)$ is a vector of dimensions $n_xN_p \times 1$, represented as Λ_{i1}.

$$M_i = \begin{cases} A_d, & for\ i = 1 \\ A_dM_{i-1}, & for\ 1 < i \le N_d \\ A_sM_{i-1}, & for\ N_d < i \le N_d + N_s \end{cases}$$

$$\Phi_{ij} = \begin{cases} 0, & for\ i < j \\ B_i, & for\ i = j \\ A_i, \times\Phi_{i-1,j} & for\ i > j \end{cases} \tag{11}$$

where A_i and B_i are defined as follows:

$$A_i, B_i = \begin{cases} A_d, B_d, & if\ i \le N_d \\ A_s, B_s, & if\ N_d < i \le N_s \end{cases} \tag{12}$$

$$\Lambda_{i1} = \begin{cases} C_d, & for\ i = 1 \\ A_d\Lambda_{i-1,1} + C_d, & for\ 1 < i \le N_d \\ A_s\Lambda_{i-1,1} + C_s, & for\ N_d < i \le N_d + N_s \end{cases} \tag{13}$$

Eq. (9) being rearranged can be expressed as follows:

$$\begin{aligned} J &= (M(k)E(k) + \Phi(k)\tilde{U}(k) + \Lambda(k))^T\tilde{Q}(M(k)E(k) \\ &\quad + \Phi(k)\tilde{U}(k) + \Lambda(k)) \\ &\quad + \tilde{U}(k)^T\tilde{R}\tilde{U}(k) \\ &\approx \tilde{U}(k)^T(\Phi(k)^T\tilde{Q}\Phi(k) + \tilde{R})\tilde{U}(k) \\ &\quad + (\Phi(k)\tilde{U}(k))^T\tilde{Q}(M(k)E(k) + \Lambda(k)) \end{aligned} \tag{14}$$

$$H(k) = \Phi(k)^T\tilde{Q}\Phi(k) + \tilde{R}f(k) = \Phi(k)^T\tilde{Q}(M(k)E(k) + \Lambda(k)) \tag{15}$$

where H is a *Hessian* matrix of the quadratic part of the cost function, which should be positive definite, and f is the

linear part. The input and state constraints (9) in the N_p-step prediction form can be written as:

$$
\begin{aligned}
G\tilde{U}(k) &\leq W(k) \\
&\text{where} \\
G &= \begin{bmatrix} G_e\Phi(k) \\ G_u \end{bmatrix} G_e = \begin{bmatrix} I_{(n_x,N_p)} \\ -I_{(n_x,N_p)} \end{bmatrix} \\
G_u &= \begin{bmatrix} I_{(n_u,N_p)} \\ -I_{(n_u,N_p)} \end{bmatrix} W = \begin{bmatrix} W_e \\ W_u \end{bmatrix} \\
W_e &= \begin{bmatrix} \tilde{E}_{\max} - G_e(M(k)+\Lambda(k)) \\ -\tilde{E}_{\min} - G_e(M(k)+\Lambda(k)) \end{bmatrix} \\
W_u &= \left[\tilde{U}_{\max}; -\tilde{U}_{min}; \Delta\tilde{U}_{\max}; -\Delta\tilde{U}_{\min}\right] \\
&\text{where} \\
\tilde{E}_{\max} &= [E_{\max}, \cdots, E_{\max}]^T, \tilde{E}_{\min} = [E_{\min}, \cdots, E_{\min}]^T \\
\tilde{U}_{\max} &= [U_{\max}, \cdots, U_{\max}]^T, \tilde{U}_{\min} = [U_{\min}, \cdots, U_{\min}]^T \\
\Delta\tilde{U}_{\max} &= [\Delta U_{\max}, \cdots, \Delta U_{\max}]^T \\
\Delta\tilde{U}_{\min} &= [\Delta U_{\min}, \cdots, \Delta U_{\min}]^T
\end{aligned} \tag{16}
$$

The infeasibility problem of the optimization is a severe problem faced by the MPC algorithm owing to the unexpected large disturbance or mismatch of the predicted and actual behaviors of the dynamics. For online implementation, the soft constraint method was employed to guarantee the feasibility. By introducing a slack variable ϵ, the final optimization problem takes the following form:

$$
\begin{aligned}
&\min_{U_k,\epsilon} \frac{1}{2}\tilde{U}_k^T H\tilde{U}_k + f^T\tilde{U}_k + \rho\epsilon \\
&\text{s.t. Equality constraints (7)} \\
&\quad G\tilde{U} \leq W + \epsilon \\
&\quad 0 \leq \epsilon
\end{aligned} \tag{17}
$$

where $\rho > 0$ denotes the penalty parameter. Subsequently, equation (15) can be used to reformulate the cost function expressed in equation (9) to the form of the cost function in the QP minimization problem (17). The first optimal control input sequence from solving this QP problem was used as the control input. An approach that varies the sampling of these prediction models can increase the physical prediction horizon without increasing the number of prediction steps. We aimed to reduce the computational complexity by identifying the advantages of further increasing the prediction interval, and consequently reducing the number of prediction steps.

B. A CRITERION FOR VARIABLE SAMPLING TIME OF THE SPARSE MODEL

The sampling time of the prediction model varied with two variables. One variable was the ratio of the previous optimal cost and the current optimal cost, and the other one was the magnitude of the desired path curvature. The optimization problem is a minimization problem. Thus, controllers should be designed such that the optimal costs are decreased. Therefore, it is necessary to design a criterion that changes the model to reduce the optimal cost considering the optimal cost ratio. However, using only the optimal cost ratio can unnecessarily alter the model even in small disturbances during path following. Because a sharp curvature of the desired path can usually increase the path-following error, a reasonable approach would be varying the prediction model by considering both the optimal cost ratio and curvature of the desired path. When the MPC problem is solved, the optimal cost can be calculated as follows:.

$$
J_k^* = \sum_{i=1}^{N_p} \tilde{E}(i)^T\tilde{Q}\tilde{E}(i) + \tilde{U}^*(i)^T\tilde{R}\tilde{U}^*(i) \tag{18}
$$

where $*$ is the optimal value of the problem (9). The curvature value was calculated using the target points. If the target path curvature is large, because the path-following performance is degraded significantly, it needs to be supplemented by reducing the sampling time of the prediction model. Similarly, when the path-following error is reduced significantly compared to the previous control period, and if the target path curvature is less than a threshold C_{th}, the sampling time of the prediction model must be increased because the length of the prediction horizon must be increased to obtain information about obstacles from a distant future. Thus, we defined our simple switching algorithm as shown in Algorithm 1.

Algorithm 1 MPC With the Variable Sampling Model on the *Sparse* Part

Input: $C_c, C_p, C_n, N_{np}, N_{nc}$
Output: N_n
while $T < sim_{total}$ **do**
  solving the MPC problem, Eq. (9)
  $C_t = (C_p - C_n)/C_p$
  if $N_{np} == N_{nc}$ **then**
    if $C_t \geq \gamma$ **then**
      if $C_c \leq C_{th}$ *and* $N_n < N_{n,\max}$ **then**
        $N_n \leftarrow N_n + \Delta t$
      end if
    end if
    if $C_t \leq -\gamma$ **then**
      if $C_c \geq C_{th}$ *and* $N_n > N_{n,\min}$ **then**
        $N_n \leftarrow N_n - \Delta t$
      end if
    end if
  end if
end while

In Algorithm 1, C_p is the constrained optimal cost in the previous instant, and C_n is the present cost. C_{th} denotes the threshold value of the curvature. C_c is the desired curvature along the path computed over the entire predicted horizon. C_t is a contraction factor, which is the rate of change in the optimal costs. C_t indicates the extent to which the present optimal cost is reduced with respect to the previous cost. Both the contraction factor and curvature along the path are exploited to choose an appropriate sampling time for the *Sparse* prediction model.

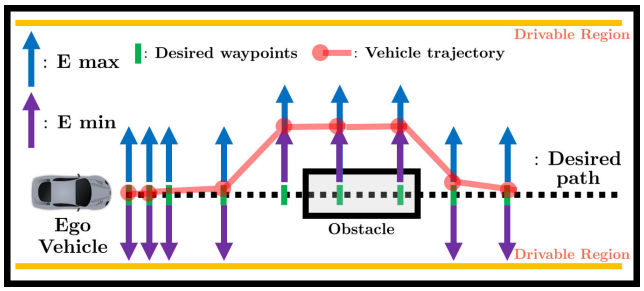


FIGURE 5. Collision avoidance constraints: The sign of E_{min} and E_{max} determine the direction of obstacle avoidance (Left or Right).

To compare the optimal cost of the same model, changing the model has a delay at every control instant (Line 6 in *Algorithm* 1: $N_{np} == N_{nc}$). N_{np} is the N_n value of the previous instant, and N_{nc} is the N_n value of the current instant. For example, if the value of N_n is changed from 9 to 10 at instant k, then the value of N_n in instant $k+1$ holds at 10, and then C_t is calculated at the instant $k+2$. If the current optimal cost is greater than the previous cost, then C_t is negative, and if the value of C_t becomes less than the threshold -γ, then a smaller sampling time is required. Furthermore, if the magnitude of curvature of the desired path, C_c is more than a threshold value, then the sampling time needs to be decremented by one to further consider the near future. Similarly, if the value of C_t is greater than γ, then the path-following performance is improved. In this case, if the magnitude of curvature along the desired path C_c is less than the threshold value C_{th}, the sampling time of the *Sparse* part is incremented by one to extend the length of the prediction horizon.

If the value of γ and C_{th} are extremely low, the model can be easily changed even with small cost changes, and the threshold of these values is obtained empirically. Consequently, not only this optimal control ratio but also the curvature values of the desired path are used to account for situations in which model variations are essential. This approach can decrease the computational complexity while retaining the long prediction horizon length to address the collision avoidance problem efficiently.

C. TIME-VARYING OBSTACLE AVOIDANCE CONSTRAINTS

In this study, the collision avoidance method uses a time-varying constraint based on a physical error bound (E_{min}, E_{max}). Moreover, this method causes the ego vehicle to avoid obstacles deviating from the desired path using the minimum and maximum values of the state variable. This method guarantees the global optima and has the advantage of solving convex optimization problems compared to other obstacle avoidance methods, such as potential fields [2]. Fig. 5 shows the value of lateral deviation used for collision avoidance when following the path.

The collision avoidance maneuver presented here relies on virtual, predefined bounds of the drivable area, and obstacles. The obstacle avoidance function assumes a situation in which obstacles are recognized within the prediction horizon length. Thus, additional local planner methods or heuristic obstacle recognition assumptions were not made, as in the existing studies [10], [11]. Sampling time of the prediction model did not change during the obstacle avoidance process to guarantee the safety of vehicle driving. The experimental environments can provide multiple traversal paths, such as the left or right traversal of obstacles, and need to be selected before determining the boundaries. The desired path does not need to be obstacle-free, and these bounds can be used to deviate the vehicle from the desired path. At each time step k, error bounds are defined to confirm collision avoidance, and it can be compactly written as a linear inequality:

$$\begin{aligned} Y_c E(k) &\leq b_c(k) \\ &\text{where} \\ Y_c &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \\ b_c(k) &= \begin{bmatrix} E^k_{\max} \\ -E^k_{\min} \end{bmatrix} \end{aligned} \tag{19}$$

$E^k_{\min}$ and $E^k_{\max}$ indicate the lateral deviation bounds ensuring safety from collisions with obstacles for the time step k. Subsequently, (19) is included in the constraints of (16).

Remark 1: Investigating the stability of MPCs with a non-uniformly spaced optimization horizon is a nonstandard problem; thus, it is difficult to address this problem using existing methods, such as a Lyapunov function [23]. To confirm that this function is decreasing, terms related to different points on the optimization horizon are used to cancel each other. Unfortunately, this approach cannot be used to guarantee the stability of MPCs with a non-uniformly spaced optimization horizon, as the above 'cancelation' does not occur due to the varying time intervals. A stability condition for MPC with a non-uniformly spaced optimization horizon was developed in [24] based on the dissipativity (an input-output property) in defining plant-to-controller relationships. However, the stability issue in situations where prediction models vary is complex and requires further investigation.

TABLE 1. The main parameters of the vehicle.

Parameter	Symbol	Value	Units
Vehicle mass	m	1,650	kg
Yaw inertia	I_z	2,650	kg·m^2
Front axle to CG	I_f	1.1	m
Rear axle to CG	I_r	1.7	m
Cornering stiffness of front-axle	C_f	55,494	N/rad
Cornering stiffness of rear-axle	C_r	55,494	N/rad
Sampling time	Δt	0.01	s

IV. THE SIMULATION TEST

A. THE SIMULATION SETUP

To verify the effectiveness of the proposed method, simulation results are presented, obtained using *CARLA* [25] simulator and a robotic operating system (ROS) [26]. *CARLA* is an open-source simulator developed for autonomous driving

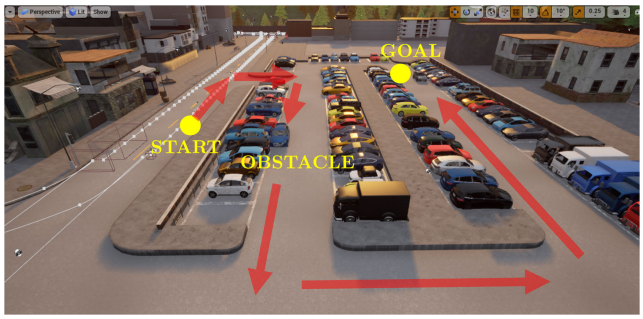


FIGURE 6. Screen shot of CARLA simulator: Red arrows represent roughly the direction of the desired path.

studies. A parking lot environment was constructed by measuring the parking lot of the *Advanced Institute of Convergence Technology* (*AICT*), South Korea.

The driving speed was kept constant at 5 km/h. Simple paths with monotonous curvatures were used in previous studies on the path-following algorithm [27], [28], to demonstrate the performance of their algorithms. In this study, we obtained the desired path by moving the vehicle at various steering angles and speeds in the simulator. Hence, the path used had a scenario in which the tracking performance could be tested in various path environments with straight lines and varying curvature sections. Obstacles were set virtual to provide consistent and repeatable tests to verify the control method. Table 1 lists the parameters of the vehicle used in the simulations. *CVXGEN* [29] generated a code to solve the convex optimization problem. All experiments were carried out in a C++ implementation on an Intel (R) Core (TM) i7-7700K CPU @ 4.20 GHz CPU, 16G GB memory, and Ubuntu 16.04 LTS operating system.

Four types of MPC-based controllers were used to perform the path following and obstacle avoidance tests, as shown in Fig. 7. The blue dots denote the proposed algorithm. The red dots indicate the STD-MPC, and the green dots indicate the DUAL-MPC [10]. The values in parentheses refer to the number of prediction steps. The parameters of the simulation were determined experimentally, as listed in Table 2. $Q1$ through $Q4$ are the diagonal component values of the weight matrix Q. The sampling time of the prediction model of the *Sparse* part varies between 0.01 and 0.3 s. The reason for choosing the 30-prediction steps for the STD-MPC is to compare the STD-MPC and the proposed method with roughly equal prediction horizon length (i.e., total prediction length $\rightarrow$ the proposed (9): $0.01(\text{s}) \times 2 + 0.3(\text{s}) \times 7 = 2.12(\text{s})$, STD-MPC (30): $0.07(\text{s}) \times 30 = 2.1(\text{s})$). Additionally, we compared the proposed method to results with those of the DUAL-MPC which is simply increasing the sampling time of the prediction model to identify the differences in path-following performance. We also compared the proposed method results with those of the STD-MPC (30) to verify the obstacle avoidance capability of the proposed method with similar path-following performance and low computational cost at the same time.

TABLE 2. Parameters of MPC methods for simulation tests.

Prediction Models		
Methods	# of Dense model, sampling time (s) = N_d	# of Sparse model, sampling time (s) = N_s
STD-MPC (9)	9, 0.01	0
DUAL-MPC (9) [10]	2, 0.01	7 (fixed), 0.3
Proposed-MPC (9)	2, 0.01	7 (varying), 0.01∼0.3
STD-MPC (30)	30, 0.07	0
Algorithm Parameters		
Description	Symbol	value
Weightings	Q1,Q2,Q3,Q4,R	500,0.1,0.2,0.1,5
Input constraints	$\delta, \Delta\delta$	0.52 (rad), 0.01 (rad)
Max expected lateral error	E_{max}	3.0 (m)
Threshold of the ratio (C_t)	γ	0.01
Threshold of the curvature (absolute value)	C_{th}	0.01 (m^{-1})
Min, Max sampling time (N_n)	$N_{n(min,max)}$	1, 30

B. THE SIMULATION RESULTS

Fig. 7-12 shows a comparison among the four MPC-based methods in path-following performance and collision avoidance. As shown in Fig. 7, the desired path includes four driving scenarios: (i) from 'Start' to 'Closed-up (1)' is the process of accurate path-following as a curved section, (ii) from 'Closed-up (1)' to 'Closed-up (2)' is the stabilization section, (iii) 'Closed-up (2)' is the process of collision avoidance, (iv) from 'Closed-up (2)' to 'Goal' is a path-following section with various curvatures of the desired path. The whole trajectories and the lateral distance error using the four MPC-based control methods are shown in Fig. 7.

In Fig. 7(a), the area marked by a red box 'Closed-up (1)' is the section with a high curvature (Fig. 7(b)) and the area 'Closed-up (2)' is the collision avoidance section as shown in Fig. 7(c). In Fig. 7(a), the proposed and DUAL-MPC methods start the drive with a sampling of the spare model part '30', as shown in Fig. 11. The '30' on the vertical axis in Fig. 11 represents a prediction model with a 0.3-second sampling time based on the control cycle of 0.01 s. Trajectories of the four methods on the curved section are shown in Fig. 7(b). The DUAL-MPC method exhibits the largest distance error, as shown in Fig. 7(d) ($250<k<700$). On the contrary, the proposed method used a 0.3-second sampling time model, as the *Sparse* part of DUAL-MPC; however, as shown in Fig. 9 and Fig. 11, according to the C_t and curvature values, the distance error is almost similar to the STD-MPC method because it can decrease the sampling time of the prediction model in advance, as shown in Fig. 11 ($250<k<700$). The sampling time of the *Sparse* part using the proposed method is expressed in the gradation, as shown

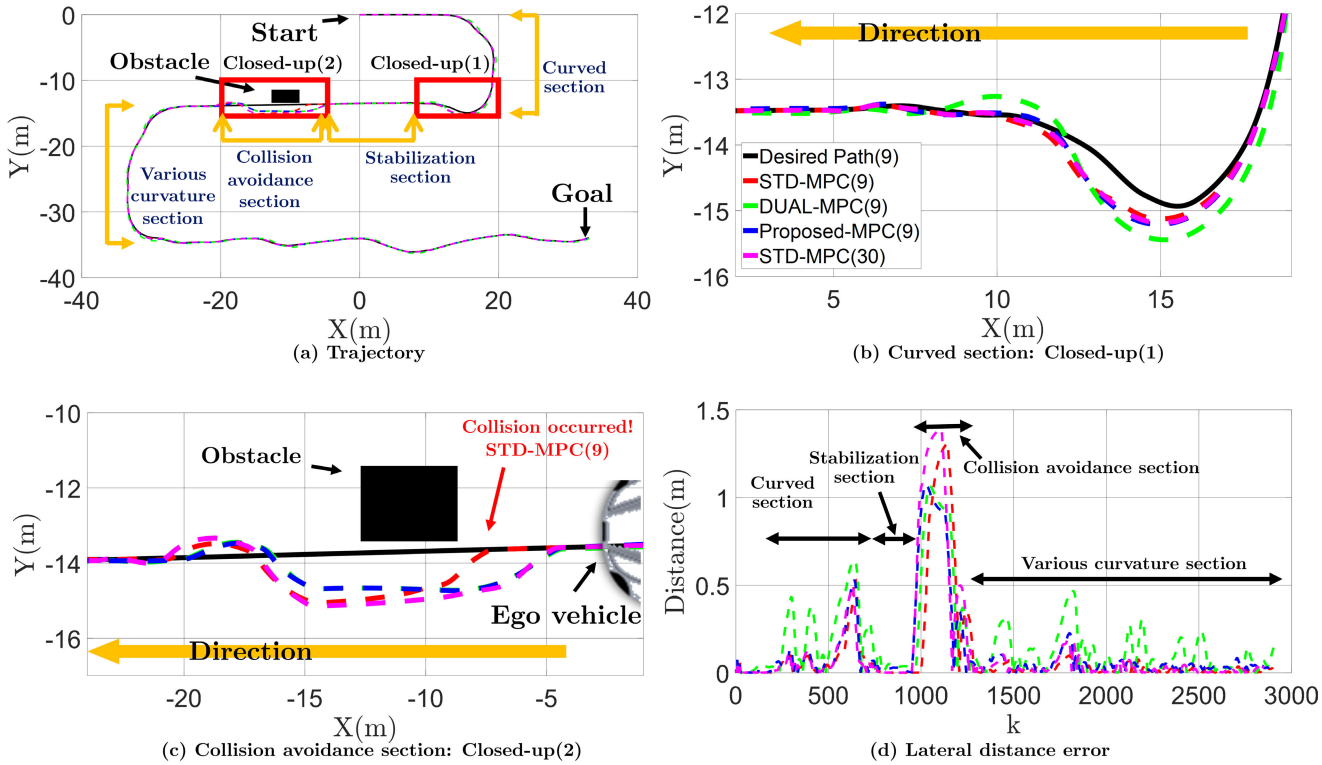


FIGURE 7. (a) Path following and collision avoidance trajectories. (b) Closed-up (1) - Curved section. (c) Closed-up (2) - Collision avoidance section. (d) Lateral distance error.

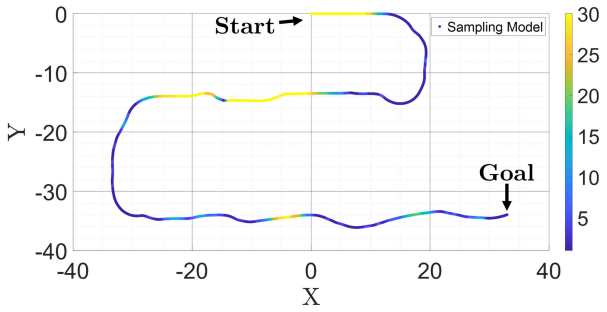


FIGURE 8. The prediction model sampling time of the *Sparse* part is expressed in gradation. The 5 to 30 indicated on the right axis of the graph means a multiple of the sampling time (ex: 30 = Δ t × 30 = 0.3 (s)).

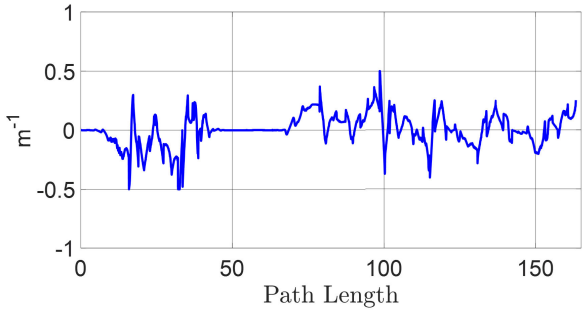


FIGURE 9. Curvature (length of path: 165 (m), maximum of curvature (abs): 0.5 (m^{-1}), average of curvature: 0.098 (m^{-1})).

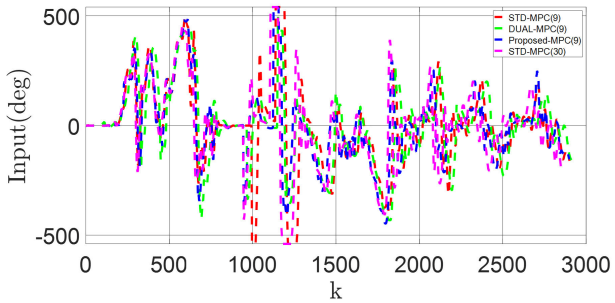


FIGURE 10. Steering input.

in Fig. 8. It is clear that the sampling time of the *Sparse* part, where the curvature is small, is relatively long in the desired path. Fig. 10 shows the steer angle.

Next, 'stabilization section' of Fig. 7(d), the four methods can perform path stabilization properly. In the obstacle avoidance section, shown in Fig. 7(c), three methods succeeded in obstacle avoidance except for the STD-MPC (9), which has a relatively short prediction length. As shown in Fig. 11 (700<k<800), the proposed method increased sampling time of the *Sparse* part during this section according to the optimal control cost and curvature value. Therefore, the proposed method can effectively avoid obstacles.

As shown in the 'Various curved section' in Fig. 7(d), the proposed method allows us to vary the sampling time of the prediction model, reducing the following error compared to the DUAL-MPC method. The maximum and average values of the distance error excluding the obstacle

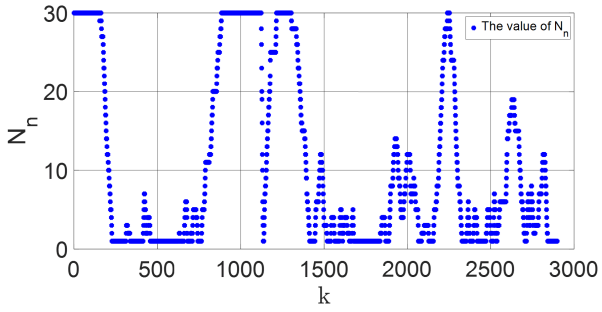


FIGURE 11. The value of N_n: only proposed-MPC (9).

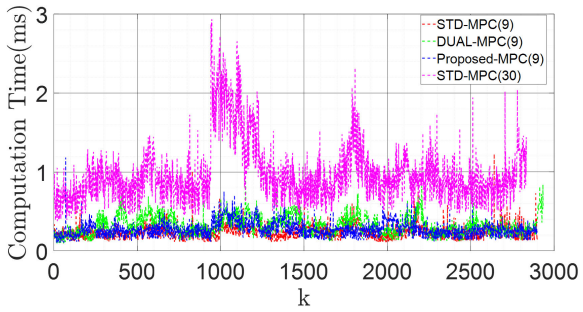


FIGURE 12. Computation time (ms) in the simulation.

TABLE 3. Simulation results.

Methods (# of Horizons)	Max. Distance Error (m)	Avg. Distance Error (m)	Collision Avoidance
STD-MPC (9)	0.53	0.06	X
DUAL MPC (9) [10]	0.64	0.18	O
Proposed MPC (9)	0.52	0.07	O
STD-MPC (30)	0.49	0.07	O

TABLE 4. Computation time (ms) in the simulation.

Methods	Average Computation Time (ms)	Computational Complexity [30]
STD-MPC (9)	0.24	$\mathcal{O}(9^4)$
DUAL-MPC (9) [10]	0.32	$\mathcal{O}(9^4)$
Proposed-MPC (9)	0.27	$\mathcal{O}(9^4)$
STD-MPC (30)	0.97	$\mathcal{O}(30^4)$

avoidance section are listed in Table 3. The proposed method can reduce the maximum distance and average distance errors by 18% and 61%, respectively, compared to the DUAL-MPC method, which simply increases sampling time of the prediction model. In addition, the average computation time can be reduced by 72% compared with the standard method, STD-MPC (30), while avoiding the obstacle (Table 4). This is because the varying sampling model approach offers a significant reduction in computational complexity to solve the optimization problem, as shown in Fig. 12.

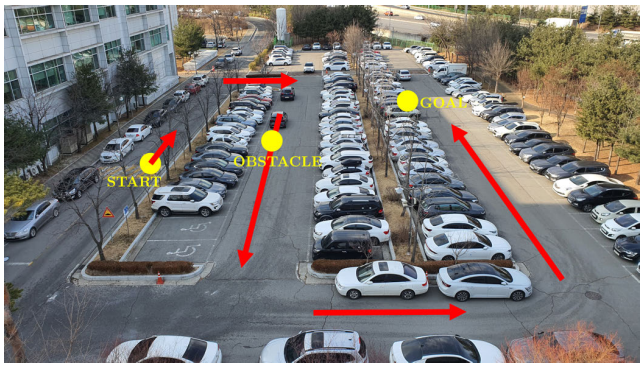


FIGURE 13. The parking lot of advanced institute of convergence technology. Red arrows represent roughly the direction of the desired path.

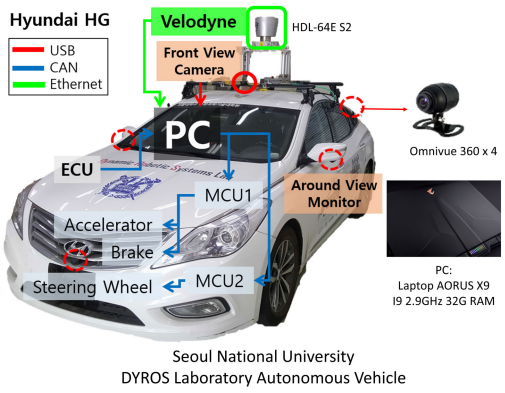


FIGURE 14. Hardware configuration of autonomous vehicle. The autonomous vehicle of seoul national university dynamic robotic systems (DYROS) laboratory is used.

Consequently, path-following performance degradation can be reduced using the proposed method, and obstacle avoidance can be performed efficiently with reduced computational complexity.

V. THE EXPERIMENTAL TEST

A. THE EXPERIMENTAL SETUP

The experiment were carried out to evaluate both the path-following and collision avoidance performances using the proposed method, which was implemented on an autonomous vehicle, as shown in Fig. 14. The experiments were conducted in an outdoor parking lot of the *AICT* located in Gwanggyo-dong, Suwon city, South Korea (Fig. 13). The autonomous vehicle platform had a 3D-LiDAR (Velodyne-64) sensor. Because GPS reception is challenging in outdoor areas immediately next to the building, the ego vehicle position and orientation angle were estimated by simultaneous localization and mapping (SLAM) and LeGO-LOAM based on the 3D lidar sensor. LeGO-LOAM was proposed in [31], making it possible to achieve real-time pose estimation on a low-power embedded system (driving a translation error of approximately, 3km-0.2m) using the KITTI benchmark dataset [32].

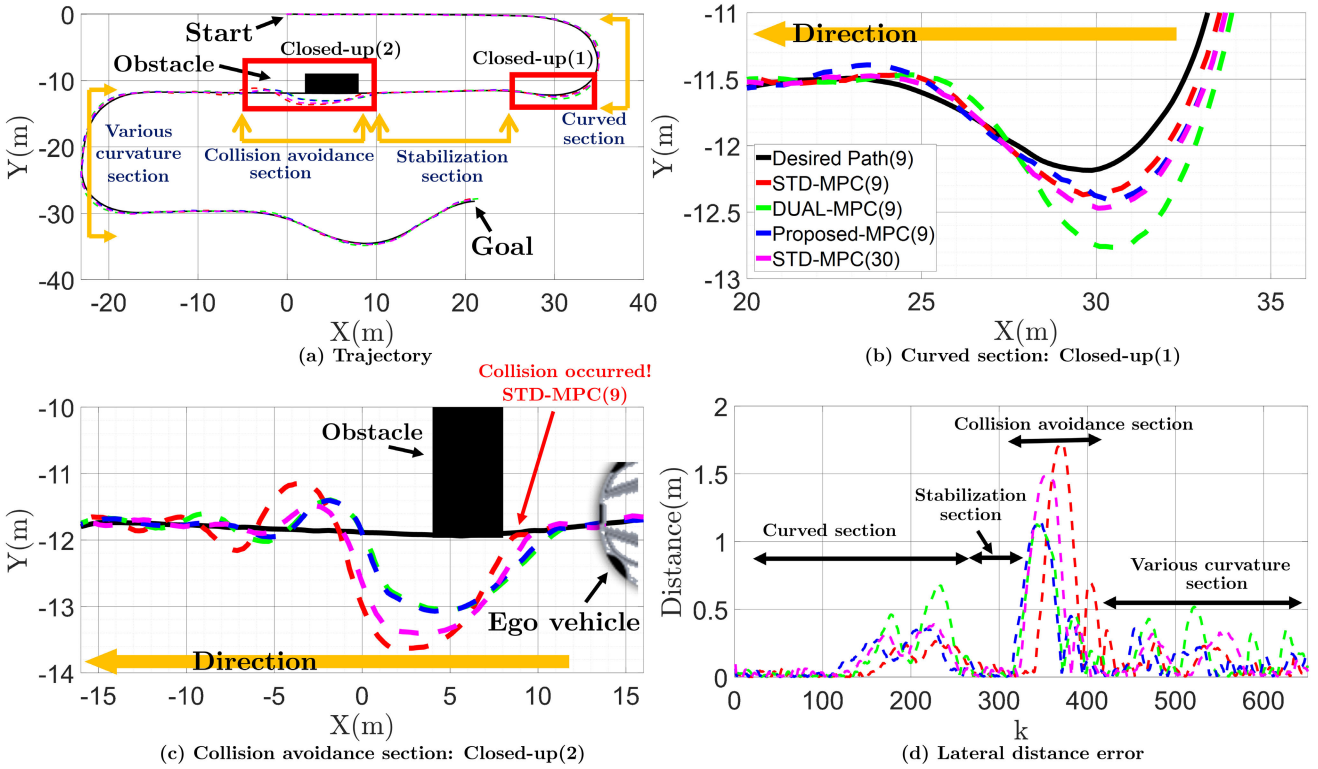


FIGURE 15. (a) Path following and collision avoidance trajectories. (b) Closed-up(1) - Curved section. (c) Closed-up(2) - Collision avoidance section. (d) Lateral distance error.

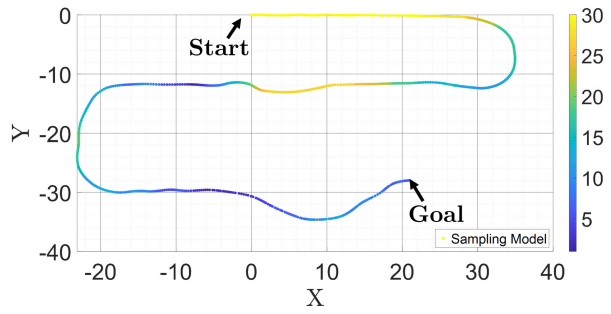


FIGURE 16. The prediction model sampling time of the 'Sparse' part is expressed in gradation.

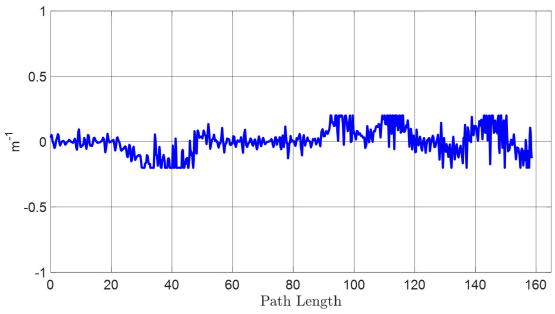


FIGURE 17. Curvature (length of path: 159 (m), maximum of curvature (abs): 0.2 (m^{-1}), average of curvature: 0.077 (m^{-1})).

The controller was implemented in C++ programming environment under Ubuntu ROS with a PC (Intel i9, 32 GB of RAM), and operated at 10 Hz. The vehicle moved at an average speed of 5 km/h in the parking lot. The algorithm commands the steering angle via a motor-driven power steering (MDPS) module. The desired path was obtained by driving the vehicle directly and then interpolating it.

B. THE EXPERIMENTAL RESULTS

All driving environments were identical to the simulation environments, and the results were comparable. In the experiment, the maximum wheel angle was limited to 0.48 rad to avoid overloading the handle column.

Fig. 15(d) shows that it exhibited the largest distance error using the DUAL-MPC in the 'Curved section'. On the other hand, the proposed method could reduce the path-following error by reducing sampling time of the prediction model in advance, as shown in Fig. 18. In addition, all methods could reduce the error to less than 0.1 m in the 'Stabilization section'. The other three methods, except for the STD-MPC (9), succeeded in avoiding the obstacle in the 'Collision avoidance section'. After avoiding the obstacle, the controllers exhibited oscillations (Fig. 15(c)) to converge on the desired path, and the STD-MPC (30), which used the highest number of prediction steps, damped the oscillation quickly. The experimental results are listed in Table 5.

TABLE 5. Experimental results.

Methods(# of Horizons)	Max. Distance Error (m)	Avg. Distance Error (m)	Collision Avoid-ance	Avg. Com-putation Time (ms)
STD-MPC (9)	0.26	0.14	X	0.17
DUAL MPC (9) [10]	0.68	0.17	O	0.19
Proposed MPC (9)	0.36	0.13	O	0.16
STD-MPC (30)	0.38	0.13	O	0.75

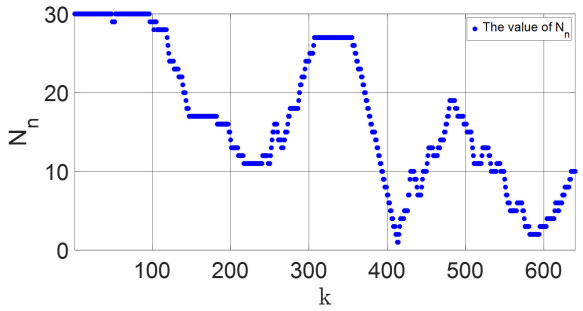


FIGURE 18. The value of N_n: only proposed-MPC (9).

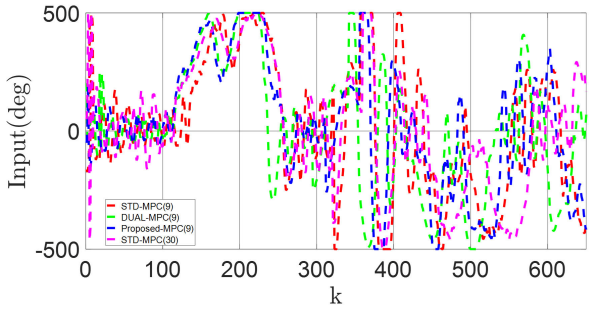


FIGURE 19. Steering input.

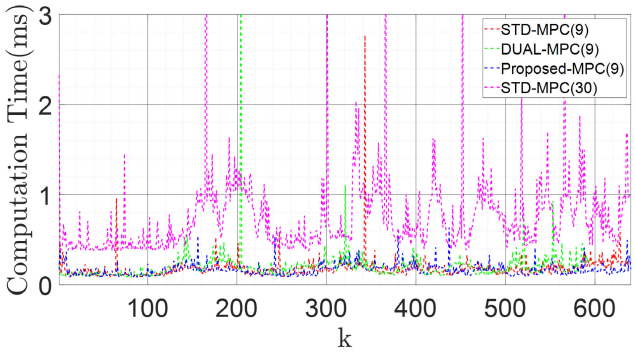


FIGURE 20. Computation time (ms) in the experimental test.

The proposed method can reduce the maximum distance and average distance errors by 47% and 23%, respectively, compared to the DUAL-MPC method. The prediction model sampling time of the *Sparse* part using the proposed method is expressed in the gradation shown in Fig. 16. Fig. 19 shows the steer angle.

The computational time of the MPC, as shown in Fig. 20, is closely related to the number of prediction steps, and the proposed method exhibited an average of 78% reduction in computational time compared to the STD-MPC (30), which increased the number of predictions while using a fixed prediction model. This can be an evidence to confirm that path-following and avoiding obstacles can be realized efficiently and safely by varying the prediction models.

VI. CONCLUSION

In this study, a path-following and obstacle-avoidance algorithm was proposed for an autonomous vehicle that varied sampling time of the MPC prediction model. The path-following and collision avoidance performances were evaluated using both the CARLA simulator and a real vehicle. The simulation results demonstrated that the proposed method was successful in mitigating path-following performance degradation compared to the DUAL-MPC method by changing sampling time of the prediction models. The proposed method could reduce the maximum distance error and average distance error by 18% and 61%, respectively, compared to the DUAL-MPC method, which simply increased the prediction length. In addition, the proposed method could avoid obstacles by reducing the number of prediction steps of the MPC. Compared with the standard MPC, the computational time could be reduced by 72% while avoiding obstacles. Furthermore, the proposed method efficiently performed path following and collision avoidance on paths with large curvatures in real vehicle tests. The proposed method reduced the maximum and average distance errors by 47% and 23%, respectively, compared with the DUAL-MPC method. Furthermore, the proposed method could reduce the calculation time by 78% compared with the standard MPC while avoiding obstacles. Consequently, the proposed method can be expected to have a greater effect on reducing the computational complexity of the nonlinear MPC.

Our future study will integrate the proposed method and recognition method of the camera to park autonomously at a lower cost in an actual parking environment.

REFERENCES

[1] J. M. Snider, "Automatic steering methods for autonomous automobile path tracking," Robot. Inst., Pittsburgh, PA, USA, Tech. Rep. CMU-RITR-09-08, 2009.
[2] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 952–964, Feb. 2017.
[3] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.
[4] H. Choset and J. Burdick, "Sensor based motion planning: The hierarchical generalized Voronoi graph," in *Algorithms for Robot Motion Manipulation*. Wellesley, MA, USA: AK Peters, 1996, pp. 47–61.
[5] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.
[6] V. Boor, M. H. Overmars, and A. F. van der Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1999, pp. 1018–1023.
[7] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, Mar. 1986.

[8] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.
[9] F. Kuhne, W. F. Lages, and J. G. da Silva, Jr., "Model predictive control of a mobile robot using linearization," in *Proc. Mechatronics Robot.*, 2004, pp. 525–530.
[10] J. Funke, M. Brown, S. M. Erlien, and J. C. Gerdes, "Collision avoidance and stabilization for autonomous vehicles in emergency scenarios," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 4, pp. 1204–1216, Jul. 2017.
[11] K. Liu, J. Gong, A. Kurt, H. Chen, and U. Ozguner, "Dynamic modeling and control of high-speed automated vehicles for lane change maneuver," *IEEE Trans. Intell. Vehicles*, vol. 3, no. 3, pp. 329–339, Sep. 2018.
[12] R. Hajiloo, M. Abroshan, A. Khajepour, A. Kasaiezadeh, and S.-K. Chen, "Integrated steering and differential braking for emergency collision avoidance in autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 5, pp. 3167–3178, May 2021.
[13] M. AlandiHallaj and N. Assadian, "Soft landing on an irregular shape asteroid using multiple-horizon multiple-model predictive control," *Acta Astronautica*, vol. 140, pp. 225–234, Nov. 2017.
[14] P. W. Gibbens and E. D. B. Medagoda, "Efficient model predictive control algorithm for aircraft," *J. Guid., Control, Dyn.*, vol. 34, no. 6, pp. 1909–1915, Nov. 2011.
[15] P. Ogren and J. Robinson, "Receding horizon control of uavs using gradual dense-sparse discretizations," in *Proc. AIAA Guid., Navi. Control Conf.*, 2010, pp. 8084–8094.
[16] V. D. Blondel and J. N. Tsitsiklis, "A survey of computational complexity results in systems and control," *Automatica*, vol. 36, no. 9, pp. 1249–1274, Sep. 2000.
[17] B.-A. Kim, Y. S. Son, S.-H. Lee, and C. C. Chung, "Model predictive control using dual prediction horizons for lateral control," *IFAC Proc. Volumes*, vol. 46, no. 10, pp. 280–285, Jun. 2013.
[18] R. Gondhalekar and J. Imura, "Non-linear prediction horizon time-discretization for model predictive control of linear sampled-data systems," in *Proc. IEEE Int. Conf. Comput.-Aided Control Syst. Design*, Oct. 2006, pp. 597–602.
[19] R. Gondhalekar and J.-I. Imura, "Performance measures in model predictive control with non-linear prediction horizon time-discretization," in *Proc. Eur. Control Conf. (ECC)*, Jul. 2007, pp. 467–474.
[20] R. Rajamani, *Vehicle Dynamics and Control*. New York, NY, USA: Springer-Verlag, 2011.
[21] H. Pacejka, *Tire and Vehicle Dynamics*. Amsterdam, The Netherlands: Elsevier, 2002.
[22] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*. Chelmsford, MA, USA: Courier Corporation, 2013.
[23] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, Jun. 2000.
[24] C. K. Tan, M. J. Tippett, and J. Bao, "Model predictive control with non-uniformly spaced optimization horizon for multi-timescale processes," *Comput. Chem. Eng.*, vol. 84, pp. 162–170, Jan. 2016.
[25] *Carla*. Accessed May 10, 2020. [Online]. Available: https://carla.org/
[26] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *Proc. Robot. Automat. Workshop Open Source Softw.*, vol. 3, 2009, pp. 1–6.
[27] Z. Sun, R. Wang, Q. Ye, Z. Wei, and B. Yan, "Investigation of intelligent vehicle path tracking based on longitudinal and lateral coordinated control," *IEEE Access*, vol. 8, pp. 105031–105046, 2020.
[28] H. Wang, B. Liu, X. Ping, and Q. An, "Path tracking control for autonomous vehicles based on an improved MPC," *IEEE Access*, vol. 7, pp. 161064–161073, 2019.
[29] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optim. Eng.*, vol. 13, no. 1, pp. 1–27, Mar. 2012.
[30] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 2, pp. 267–278, Mar. 2010.
[31] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized LiDAR odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 4758–4765.
[32] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

MINSUNG KIM received the B.S. degree from the School of Electrical and Electronics Engineering, Chung-Ang University, Seoul, South Korea, in 2010. He is currently pursuing the Ph.D. degree in intelligent systems with the Department of Intelligence and Information, Seoul National University, Seoul. His research interests include local planning, path-following of an autonomous vehicle, optimal control theory, and robot-applied deep learning.

DONGGIL LEE (Graduate Student Member, IEEE) received the B.S. degree in electrical engineering and physics education from Seoul National University, Seoul, South Korea, in 2015, where he is currently pursuing the Ph.D. degree in electrical engineering. His current research interests include vehicle control and distributed control of multi-agent systems.

JOONWOO AHN received the B.S. degree in robotics from Kwangwoon University, Seoul, Republic of Korea, in 2016. He is currently pursuing the Ph.D. degree in intelligent systems with the Department of Intelligence and Information, Seoul National University, Seoul. His research interests include autonomous valet parking, vision-based navigation, robot-applied deep learning, and path tracking.

MINSOO KIM received the B.S. degree in mechanical engineering from Hanyang University, Seoul, South Korea, in 2018. He is currently pursuing the Ph.D. degree in intelligent systems with the Department of Intelligence and Information, Seoul National University, Seoul. His research interests include autonomous vehicle, autonomous valet parking, and motion planning.

JAEHEUNG PARK (Member, IEEE) received the B.S. and M.S. degrees in aerospace engineering from Seoul National University, South Korea, in 1995 and 1999, respectively, and the Ph.D. degree in aeronautics and astronautics from Stanford University, CA, USA, in 2006. From 2006 to 2009, he was a Postdoctoral Researcher and later a Research Associate with the Stanford Artificial Intelligence Laboratory. From 2007 to 2008, he worked part-time with Hansen Medical Inc., USA, a medical robotics company. Since 2009, he has been a Professor with the Graduate School of Convergence Science and Technology, Seoul National University. His research interests include robot-environment interaction, contact force control, robust haptic teleoperation, multicontact control, whole-body dynamic control, biomechanics, and medical robotics.