Master's Thesis

Overtaking in Autonomous Racing with Online Refinement of Opponent Behavior Prediction using Gaussian Process

Jun Sung Kwon

Department of Mechanical Engineering
Mechanical Engineering

Ulsan National Institute of Science and Technology

2023

Overtaking in Autonomous Racing with Online Refinement of Opponent Behavior Prediction using Gaussian Process

Jun Sung Kwon

Department of Mechanical Engineering
Mechanical Engineering

Ulsan National Institute of Science and Technology

Overtaking in Autonomous Racing with Online Refinement of Opponent Behavior Prediction using Gaussian Process

A thesis submitted to
Ulsan National Institute of Science and Technology
in partial fulfillment of the
requirements for the degree of
Master of Science

Junsung Kwon

12.16.2022 of submission
Approved by

Kwon

Advisor
Cheolhyeon Kwon

Overtaking in Autonomous Racing with Online Refinement of Opponent Behavior Prediction using Gaussian Process

Junsung Kwon

This certifies that the thesis of Junsung Kwon is approved.

12.16.2022 of submission

Signature

Advisor: Cheolhyeon Kwon

Signature

Hungsun Son

Signature

Hyondong Oh

Signature

Abstract

This paper addresses an overtaking strategy in autonomous head-to-head racing, by virtue of a learning-based prediction to the opponent vehicle's behavior. The existing prediction approaches either rely on prior model or off-line learning for opponent behavior, whose accuracy diminishes when the opponent in real racing exhibits different driving style. Motivated by this concern, we proposes an online learning-based prediction algorithm that can adapt to the opponents' different driving style and refine the prediction during the race. Resorting to Gaussian Process (GP) regressor as the baseline learning model, we leverage several techniques to reduce the data size and computation cost of GP, making the algorithm suitable for online learning and prediction refinement in real time. The effectiveness of the proposed algorithm is demonstrated with different simulation scenarios and compared with the other algorithms in terms of prediction accuracy, computation efficiency, and success rate of overtaking maneuver.

Contents

List of Figures

Notations

Abbreviations

EV	Ego Vehicle
TV	Target Vehicle
MPC	Model Predictive Control
MPCC	Model Predictive Contouring Control
GP	Gaussian Process
SGP	Sparse Gaussian Process Approximation
RBF	Radial-basis function
SE	The squared-exponential kernel, also known as the radial-basis function(RBF) kernel
MSE	Mean Squared Error
MC	Monte Carlo
RS	Reachable Set
IRL	Inverse Reinforcement Learning
MDP	Markov Decision Process
POMDP	Partially Observable Markov Decision Process
NLP	Nonlinear Programing

Sets

$\mathbb{R}$	Set of Real numbers
$\mathbb{N}$	Set of natural numbers
$\mathbb{R}_+$	Set of positive real numbers
$\mathbb{X}$	State constraints set

$\mathbb{U}$	input constraints set	
D	Training data set	

Indices, sub-, and superscripts

$(\cdot)^{ev}$	Ego Vehicle	
$(\cdot)^{tv}$	Target Vehicle	
$(\cdot)_f$	Front Tire	
$(\cdot)_{center}$	The value of a center track	
$(\cdot)_r$	Rear Tire	
$(\cdot)_k$	Discrete time index	
$\bar{(\cdot)}$	estimated value of a quantity	
$\hat{(\cdot)}$	predicted value of a quantity	
$\|\cdot\|$	l^2-norm	
$p(\cdot)$	probability	
$\in$	left one is element of right set	
$\subset$	left set is subset of the right set	
$Var(\cdot)$	Variance	

Vehicle state, input and parameter

$x(\cdot)$	system state	$\in \mathbb{R}^{n_x}$
$u(\cdot)$	control input	$\in \mathbb{R}^{n_u}$
n_x	System state dimension	$\in \mathbb{N}$
n_u	Control input dimension	$\in \mathbb{N}$
p_x	vehicle CG displacement along the earth-fixed X-axis	$[m]$
p_y	vehicle CG displacement along the earth-fixed y-axis	$[m]$
ψ	Rotation of the vehicle coordinate about the world coordinate z-axis	$[rad]$
$\dot{p}_x, v_x$	linear velocity in longitudinal direction along the body-fixed x-axis	$[m/s]$
$\dot{p}_y, v_y$	linear velocity in lateral direction along the body-fixed y-axis	$[m/s]$

ψ, ω	angular velocity of vehicle, about the z-axis	$[rad/s]$
s	distance vehicle drives along the track from start position	$[m$
e_y	lateral distance from the reference line	$[m]$
a_x	control acceleration in longitudinal direction along the body-fixed x-axis	$[m/s^2]$
δ	control steering angle of the front wheel	$[rad]$
N	Prediction horizon length	$\in \mathbb{N}$
k	Discrete time	$\in \mathbb{N}$
q_c	lateral distance penalizing cost	$\in \mathbb{R}$
q_s	progress cost	$\in \mathbb{R}$
R_u	control effort penalizing cost	$\in \mathbb{R}^{n_u \times n_u}$
R_u	change of the control effort penalizing cost	$\in \mathbb{R}^{n_u \times n_u}$

I Introduction

Autonomous racing is one of the most notable subtopics of autonomous driving that has recently attracted significant interest [1]. Focusing on the autonomous algorithm aspect, many competitions like Roborace, Indy Autonomous Challenge, and F1TENTH have emerged that allow researchers to implement and test their S/W stacks within archetypal H/W platforms. Albeit the competition rules are almost the same as manual racing, adhering to them fully autonomously necessitates several key algorithms, including sensing, planning, and control. And there are some notable features to consider in the racing to generate algorithms.

First, it is highly dynamic. The algorithm must account for the vehicle's limits, such as fast speeds and quick reaction times. Second, the race is competitive and adversarial. In competition, race cars attempt to drive as quickly as possible on a specific track. In addition, it must be able to identify unstructured environments, such as track conditions that are dry, wet, snowy, or sticky. Most importantly, racecars must always prioritize safety while making decisions. Even if they are racing at high speeds to overtake other vehicles, they must be aware of the other's movements and prevent situations that could result in a collision.

Formal studies have expanded the classical motion planning or control techniques to account for collision avoidance while considering opponents as static obstacles [2, 3]. However, such methods are limited to executing reliable maneuvers at high speeds. Since approximating opponents as static obstacles are no longer applicable. Furthermore, making the opponent's vehicle stationary makes it impossible to reflect one's different driving style. Therefore, to be accurately aware of the other's movements, it is necessary to accurately predict the opponent's future behavior for overtaking the opponent's vehicle without collision, which is the main interest of this paper.

1.1 Related Work

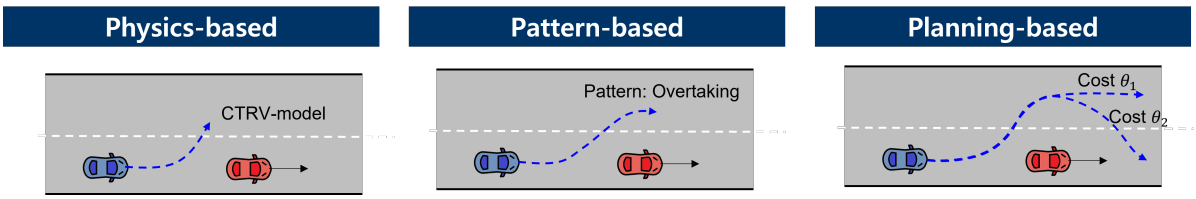


Figure 1: A visual representation of the three types of prediction methods.

We present three methodologies for motion prediction, categorized according to how they describe the object's motion; planning-based, physics-based, and pattern-based. Figure 1 demonstrates the distinction between each method in the scenario of two vehicles on a two-lane highway. We introduce each method and its application to autonomous racing.

Planning-based prediction

The planning-based prediction approach generates the trajectory of a vehicle by taking into account the actions of surrounding vehicles to achieve their goal under their hidden policies. The most important is the modeling goal and hidden policies, usually approximated by the Markov Decision Process (MDP). In [4], they construct a cost function with a linear combination of static and dynamic variables that parameterize each state. They model the trajectory of a vehicle as a series of states. In [5], The cost function parameters are learned from examples using Inverse Reinforcement Learning (IRL), which considers risk-averse vehicle encounters. These methods require more comprehensive data to infer expert behavior, particularly IL, correctly.

In addition, accounting for the opponent's driving style, the game-theoretic approaches have been considered to solve the rational driving action of the opponent for the given cost function. With these techniques, continuous motion planning is converted into a sequential game in which players can make "one move" at a time. The trajectory of each vehicle is repeatedly planned in sequence until convergence [6]. And the resulting control commands generate trajectories of rich game strategies such as blocking, faking, and opportunistic overtaking. In [7], a game-theoretic MPC is presented using a data-driven method for model identification. It plans behaviors that can respond to various race environments based on their game-based opponents' trajectory predictor and high-level race strategy planner.

However, the induced driving action could be too optimistic as the real opponent in practice may not necessarily take the rational action as assumed. Further, one can also apply IRL to identify the cost of the game-theoretic formulation, thereby directly tackling the overtaking problem [8]. Although the game-theoretic technique produces the optimal control commands, the computational complexity it demands occasionally makes the controller unsuitable for racing which needs high frequency.

Physics-based prediction

Physics-based prediction models predict future motion utilizing motion equations from classical mechanics. Dynamic or kinematic model equations form a basis for modeling the future motion of the target object to create its physical description. Numerous works which use simple kinematics have been presented to develop high-level decision-making processes to account for dynamic opponents. Constant acceleration and constant velocity model are presented under the straight-line motion assumption. The author of [9] has applied the Kalman filter to perform prediction accounting for the uncertainty in the vehicle model.

A line of these works has been developed for the constant turn rate and velocity (CTRV) and constant turn rate and acceleration(CTRA) model adding the constant vehicle state. To improve expressiveness in heterogeneous scenarios, the author of [10] proposes a method that combines multiple KF-based kinematic models in an Interacting Multiple Model (IMM) based on heuristics. Despite its simple computation, which is advantageous for real-time prediction, this method lacks an understanding of the opponent's driving style due to its simplified input assumption.

These challenges of physics-based methods in real-world racing applications require a combina-

tion with other methods to provide more accurate long-term forecasts. Reachable set (RS) predictions with a physics model are presented to represent all physically possible behaviors [11]. Kinematic simplifications can be applied to convex hulls in conjunction with an over-approximation to account for simplification errors to reduce the size of a set [12]. However, this process of approximating the sets is too conservative for long-term prediction and too costly to calculate RS without approximations in the highly dynamic autonomous racing field.

Pattern-based prediction

Pattern-based prediction models predict future motion by comparing an observation to a human-made or learned pattern. The motion of objects is classified into one of the predefined maneuver classes by human-made patterns. [13] proposed a method to recognize maneuver. Based on kinematic measurements and road geometry detection, the most similar maneuver is selected from a predefined set, and trajectory is generated and propagated with selected maneuvers. [14] applied spectral clustering to cluster vehicle trajectory with k-means based on a trajectory similarity measure, which uses modified Hausdorff(MH) distance. Then, the trajectory is represented by ordered collections of points.

Different machine learning methods are also studied to learn the pattern from data, such as hidden Markov models (HMMs) [15, 16], kernel regression [17], bayesian network [18, 19], and encoder-decoder [20]. In [16], the combination of prior and posterior probabilities for behavior prediction is first presented to learn the driving behavior of traffic scenarios. [17] generates discrete lane-changing start/endpoints while taking into account the interaction of surrounding vehicles as a non-parametric regression. To handle a large amount of data, it was extended to the Recurrent Meta Induction Neural Network (RMIN) framework [18]. [19] uses an RNN with Long Short Term Memory (LSTM) cells to predict possible lane-change intentions based on accumulated information about the surrounding vehicle and its trajectory history. Using an LSTM with an encoder and a decoder, [20] predict the likelihood of occupancy in a grid map and then use a beam search to select the k trajectories with the highest probability.

This method may be effective for standard road scenarios but not for high-speed racing conditions. Recently, an approach including physics-based prediction methodologies has been implemented to address this issue. While prior work did not explicitly encode vehicle kinematics and instead relied on data-driven learning constraints, [21] introduces kinematic constraints to deep neural networks to provide kinematically feasible motion prediction. [22] integrates dynamic constraints and diverse data into a graph-structured recurrent model for trajectory prediction. And, using uncertainty propagation of dynamic equation, it generates dynamically constrained predictions accompanied by their uncertainty. Notwithstanding the success of the learning-based prediction, they usually require a large amount of training data over a long time, and thus the learning must be processed offline. Furthermore, it doesn't give any failure explanations, such as uncertainty bounds.

To plan the overtaking maneuver of the ego vehicle in high-speed racing, it is crucial to generate a prediction model with limited data and quantify the confidence of the predicted opponent's behavior. In

that regards, Gaussian Process model is deemed to be adequate as it accompanies the predicted behavior with its covariance statistics. The Gaussian process (GP) is used in [23] to create a state transition model for a linearized kinematic vehicle model. They use the predicted trajectory to construct the free space for a stochastic MPC constraining the half space which car most likely to go. In [24], a path-following model is used to forecast the cut-in behavior of surrounding vehicles using an estimated behavior parameter obtained by GP. [25] is developed to generate the prediction of nominal trajectory and uncertainty for nonlinear dynamics via sampling. Although these GP-based methods can be used in a racing setting, the learned prediction model cannot accommodate the opponent in real racing who exhibits a different driving style from the training data [26]. This, in turn, degrades the prediction accuracy and correspondingly limits the overtaking maneuver.

1.2 Contribution

In this paper, we propose an online learning framework for predicting the opponent's maneuver while adapting with different driving styles including those who are not trained *a priori*. To the best of our knowledge, the online prediction and refinement framework for high-speed overtaking scenarios has yet to be addressed before. By learning the opponent driving style during the racing, we can refine the prediction more accurately facilitating more overtaking maneuver. Respecting the earlier work, we resort to GP model as online learning baseline. The major technical barrier is the computation overhead, as GP model is computationally prohibitive from real-time operation through the on-board unit of the vehicle. To this end, we carry out several techniques to make the online GP-based prediction viable for autonomous racing.

First, we decompose the opponent behavior model into offline GP and online GP parts, on top of the prior knowledge on the racing environment. Considering the fact that many autonomous racing competitions standardize uniform vehicle specification, the vehicle dynamics model is less sensitive to the opponent's driving style. Thus, the vehicle dynamics part can be learned through the offline GP part, while the online GP part exclusively learn the opponent's driving action model that varies with driving style. The integrated offline-online GP model takes advantage in computation aspect, i.e., significantly reducing online learning part, and also in data aspect, i.e., using both offline and online data together.

Second, we further reduce the training data size for the online GP based on the racing track information. Inspired by [27, 28] where the control policy of the vehicle depends on the spatial-temporal states over the racing track, we establish new feature space that better represents the input of the opponent's driving action model. Having smaller input dimension, once can accommodate more data points with the limited kernel matrix size of the online GP model. Meanwhile, we additionally employ the sparse GP technique to reduce the total data point [29]. To preserve the prediction accuracy, we need to appropriately update the inducing points, which is again contingent upon the racing track information. The new feature space and sparse GP technique can reduce the data dimension and point size for the online GP, which in turn reduce the computation burden.

Finally, the two on/off-line trained modules are combined together to predict the opponent vehicle

trajectory prediction, and the predicted trajectory is fed into MPC solver such that more reliable, flexible and accurate overtaking.

The main contributions are summarized as follows:

- In the midst of racing, a new prediction model is trained based on the collected data online and combined with a nominal prediction model learned offline. The combined prediction model generated the predicted action of the target vehicle considering the opponents' driving style.
- Leveraging the known model information in conjunction with the learning framework, the implemented prediction model generates a more accurate and physically feasible trajectory.
- The accuracy of the opponent's predicted trajectory is also quantified in terms of the corresponding covariance measure.This prediction information is then used as the constraints of the Model Predictive Control (MPC) for the ego vehicle overtaking maneuver.

1.3 Outline

The first part of this thesis introduces the motivation of the work and state-of-the-art trajectory prediction work used in the autonomous driving field. Chapter 2 presents the vehicle model described by the bicycle model with nonlinear tire forces. And basic theories about model predictive control(MPC) and the gaussian process (GP) are introduced. Chapter 3 introduces the details of the problem concerned with MPC Formulation and assumption. Chapter 4 describes the algorithms to learn the residual vehicle and control action model described in Figure 2. Chapter 5 describes the details of the implementation and simulation results. A final chapter, Chapter 6, is dedicated to conclusions.

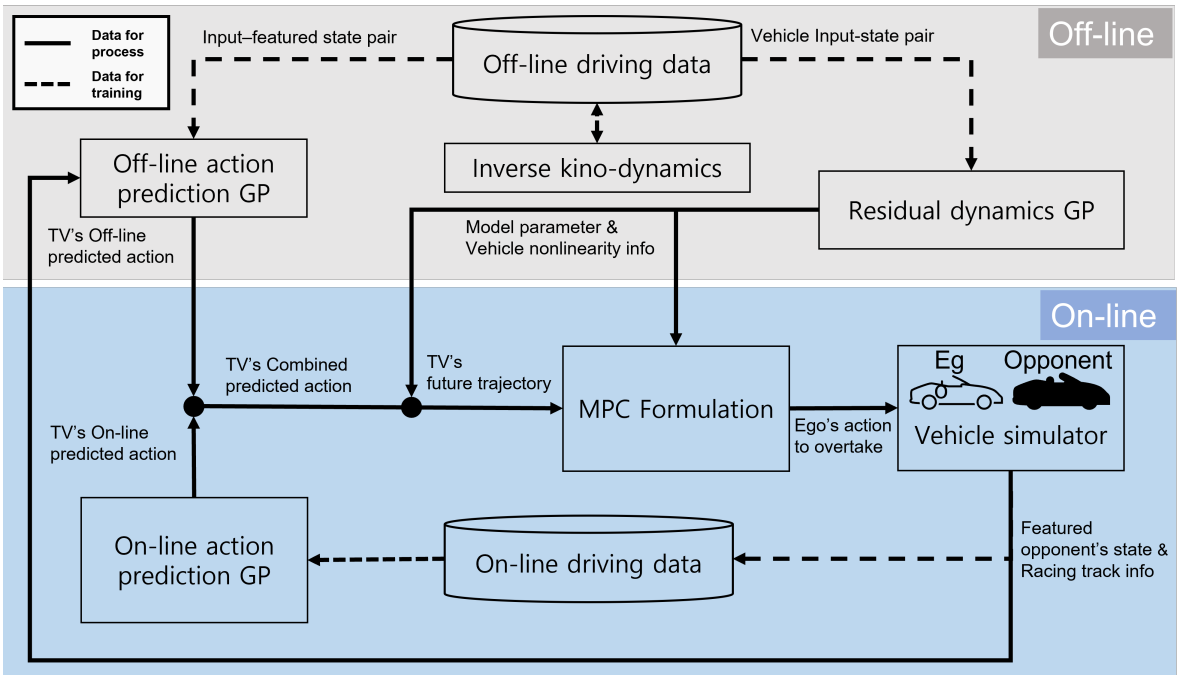


Figure 2: The overall architecture for offline and online GP-based trajectory prediction is represented here. The offline process is colored by gray color. The online process is colored blue color. The dotted arrow line represents data training, and data processing is represented by the regular arrow line with its processing data.

II Preliminary

2.1 Vehicle Dynamics

Vehicle dynamic model

We consider the vehicle to be a single rigid-body bicycle for simplicity, with a mass of m and an inertial moment of I_z. The distances l_f and l_r represent the front and back tires to the CoG, respectively. The vehicle's nominal dynamics are modeled using the dynamic bicycle in [30]. The model is represented by the following equations and described in Figure 3:

$$\begin{aligned}
\dot{p}_x &= v_x cos(\psi) - v_y sin(\psi) \\
\dot{p}_y &= v_x sin(\psi) + v_y cos(\psi) \\
\dot{\psi} &= \omega \\
\dot{v}_x &= \frac{1}{m}(F_{r,x} - F_{f,y} sin(\delta) + m v_y \omega \\
\dot{v}_y &= \frac{1}{m}(F_{r,y} + F_{f,y} cos(\delta)) - m v_x \omega \\
\dot{\omega} &= \frac{1}{I_z}(F_{f,y} l_f cos(\delta) - F_{r,y} l_r),
\end{aligned} \tag{1}$$

where p_x and p_y are the positions of the vehicle's center of gravity (CoG), ψ is the vehicle's heading, v_x and v_y are the longitudinal and lateral velocity of the CoG respectively (w.r.t. a body-fixed frame), and ω is the vehicle yaw rate. The vehsicle is controlled by the longitudinal acceleration a and steering angle δ. The subscripts f and r represent the front and rear tires concerning the vehicle's body-fixed frames, whereas the subscripts x and y indicate longitudinal and lateral direction.

Tire force model

The tire forces F_x, F_y is the part that represents the interaction between the car and the road. We ignore other terms and parameters for longitudinal tire forces, such as aerodynamic force, and describe the longitudinal force F_x as longitudinal acceleration a, which is the control action. The lateral tire forces represented by F_x, F_y are modeled using a simplified pacejka tire model defined by the following equations:

$$\begin{aligned}
F_y^f &= D^f sin(C^f + arctan(B^f \alpha_f)) \\
F_y^r &= D^r sin(C^r + arctan(B^r \alpha_r)).
\end{aligned} \tag{2}$$

The parameters B, C, D are obtained from the semi-empirical curve. The slip angles for front α_f and rear α_r are by the following equations:

$$\begin{aligned}
\alpha_f &= -arctan(\frac{\omega l_f + v_y}{v_x}) + \delta, \\
\alpha_r &= arctan(\frac{\omega l_r - v_y}{v_x}).
\end{aligned} \tag{3}$$

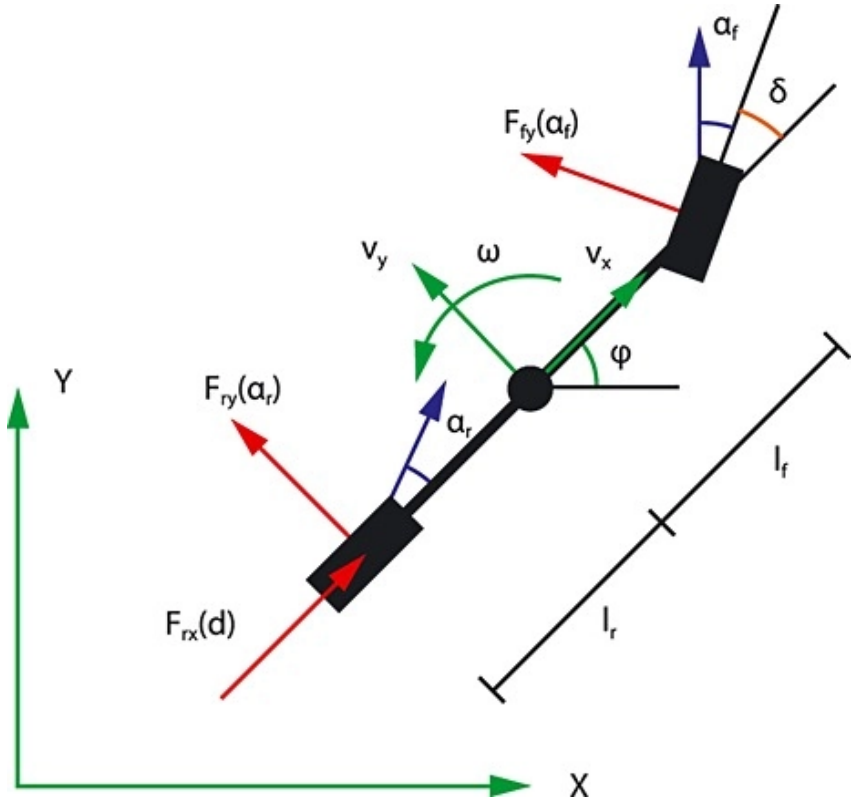


Figure 3: This figure shows the bicycle model of the vehicle dynamics. The position of the vehicle is represented by global(inertial) coordinates. The velocity of the vehicle is represented by body-fixed frames. Only the front steering angle can be controlled by the steering angle (δ). The longitudinal and lateral tire force is represented through the simplified pacejka tire model with each tire's slip angle.

Frenet Coordinate

Frenet coordinates are an effective method to represent the position of a car on the road more intuitively in Figure 4. The centerline is parameterized by the arc length s using twice continuously differentiable functions, such as third-order spline polynomials. The parameter s means the path arc length from the track's start position, where L is the entire length.We can get any vehicle position by evaluating the function $\tau(\cdot)$ for its parameter s. With the help of this parameterization, the centerline's known points can be interpolated with accuracy. Also, transforming the frenet Coordinate makes it simple to represent the track's state constraints, making it a simple box that makes it easy to formulate the MPC problem formulation in Chapter 3.2.

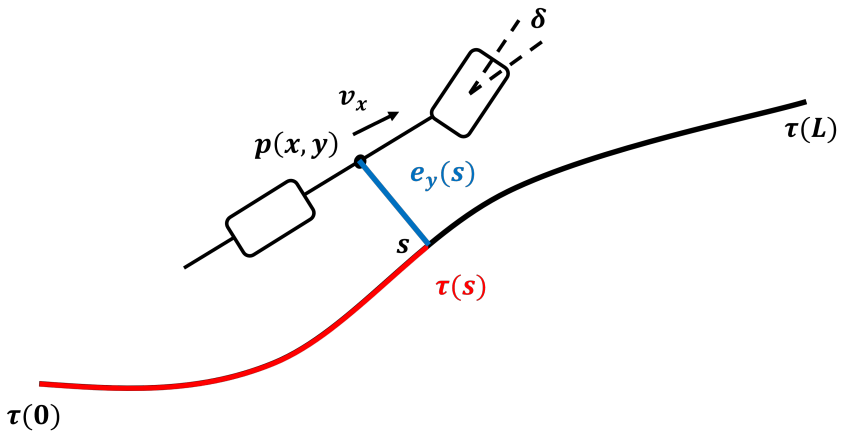


Figure 4: The representation of vehicle state in frenet coordinate

The frenet coordinates define the vehicle position using the e_y, e_ψ, the lateral distance from the center track, and the heading deviation from the centerline tangent angle ψ_{center}, respectively. The position of the vehicle in frenet coordinate is calculated from the global position $p = [p_x, p_y]$ and heading ψ as follows.

$$s(p) = \arg\min_s \|\tau(s) - p\|,$$
$$e_y(p) = \min_s \|\tau(s) - p\|,$$
$$e_\psi(p, \psi) = \psi - \arctan(\tau'(s(p))).$$

Also, the track information, curvature κ, can be calculated as follows.

$$|(s)| = \|\tau''(s)|.$$

Furthermore, the motion of the vehicle in vehicle global coordinates can be transformed to frenet coordinates using the transform matrix. The transformation matrix is the rotation matrix around the axis orthogonal to the plane by angle θ and represented as follows:

$$T = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4)$$

The rotation velocity is given by $\dot{R} = [0,0,\dot{\theta}]^T = [0,0,\kappa(s)\dot{s}]^T$. Now, using the transformation matrix and rotation velocity, we can get the equation as follows.

$$\begin{bmatrix} \dot{s} \\ \dot{e_y} \\ 0 \end{bmatrix} = T \cdot \begin{bmatrix} \dot{p_x} \\ \dot{p_y} \\ 0 \end{bmatrix} - \dot{R} \times \begin{bmatrix} 0 \\ e_y \\ 0 \end{bmatrix} = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \dot{p_x} \\ \dot{p_y} \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \kappa(s)\dot{s} \end{bmatrix} \times \begin{bmatrix} 0 \\ e_y \\ 0 \end{bmatrix}. \tag{5}$$

Solving for $\dot{s}$ and $\dot{e_y}$ and rearranging the variable results in followings.

$$\begin{aligned} \dot{s} &= \frac{v_x cos(e_\psi) - v_y sin(e_\psi)}{1 - e_y \kappa(s)}, \\ \dot{e_y} &= v_x sin(e_\psi) + v_y cos(e_\psi), \\ \dot{e_\psi} &= \dot{\psi} - \kappa(s) \frac{v_x cos(e_\psi) - v_y sin(e_\psi)}{1 - e_y \kappa(s)}, \\ \dot{v}_x &= \frac{1}{m}(F_{r,x} - F_{f,y} sin(\delta) + m v_y \omega \\ \dot{v}_y &= \frac{1}{m}(F_{r,y} + F_{f,y} cos(\delta)) - m v_x \omega \\ \dot{\omega} &= \frac{1}{I_z}(F_{f,y} l_f cos(\delta) - F_{r,y} l_r). \end{aligned} \tag{6}$$

where velocity terms is same with Equation (1).

2.2 Model Predictive Control

Model Predictive Control(MPC) is a control technique that uses control inputs obtained by predicting future output values using a process model. At every sampling time, It repeatedly solves an open-loop constrained optimal control problem (OCP) and uses the first element of optimized control actions in a receding-horizon manner depicted in Figure 5.

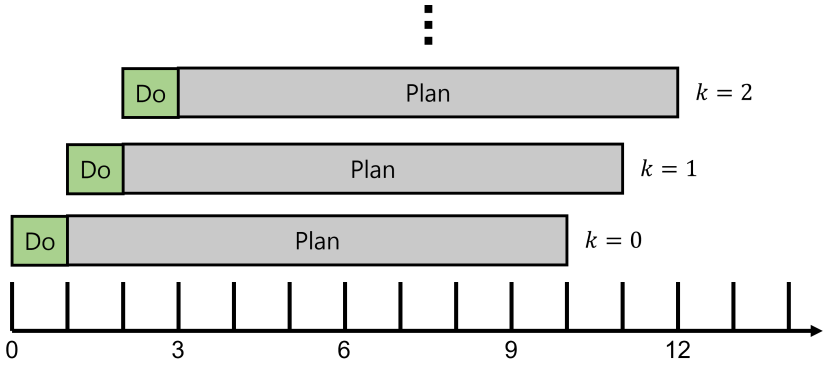


Figure 5: The receding horizon strategy of MPC formulation.

A linear time-invariant system is a system that is regularly taken into account in the literature. However, nonlinear models are also widely employed within the mpc framework. Here, we consider the following structure to describe the nonlinear system model. We also discretize the vehicle dynamics with a sampling time to formulate the MPC controller as a finite-dimensional optimization problem.

$$x_{k+1} = f(x_k, u_k) \tag{7}$$

where f is discrete nonlinear system dynamics and $x_k \in \mathbf{R}_x^n$ denotes the state vector and $u_k \in \mathbf{R}_u^n$ control input vector in time step k. The goal of mpc is to minimize a cost function while keeping state and input constraints. The general MPC controller can be formulated as an optimization problem as follows (see Figure 6).

$$\min_{U} \sum_{k=0}^{N-1} \ell(x_k, u_k) \tag{8a}$$

$$s.t.\ U = \{u_0, u_1, ..., u_{N-1}\} \tag{8b}$$

$$x_{k+1} = f(x_k, u_k), \tag{8c}$$

$$x_k \in \mathbb{X}, u_k \in \mathbb{U}, \tag{8d}$$

$$x_0 = x(k), \tag{8e}$$

Where $\ell(\cdot)$ is the stage cost we want to minimize. The optimization is carried out over a sequence of input $U = \{u_0, u_1, ..., u_{N-1}\}$. The next state x_{k+1} is the state of the system generated by (8c) with state x_k and input u_k at time step k. All states and inputs should satisfy its constraints in (8d). This nonlinear MPC is non-convex but can be solved iteratively using an off-the-shelf nonlinear solver.

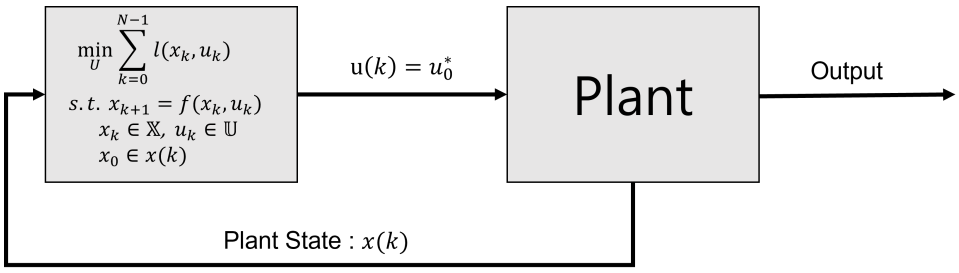


Figure 6: The Optimization loop of the MPC.

2.3 Gaussian Process

Supervised learning is a learning technique used in the field of machine learning that maps input vectors to outputs based on the relationships between input-output pairs. When data is provided, the Gaussian process(GP) regression is the method to learn a function that predicts the output at unseen input locations based on supervised learning. The definition of GP is the generalization of the multivariate normal distribution to an infinite-dimension stochastic process. And GP regression is a nonparametric bayesian regression method using the properties of the gaussian process. A GP can be interpreted in various ways from a mathematical perspective; weight-space view and function-space view [31]. Here, we will introduce the function-space view of GP which is much easier to represent.

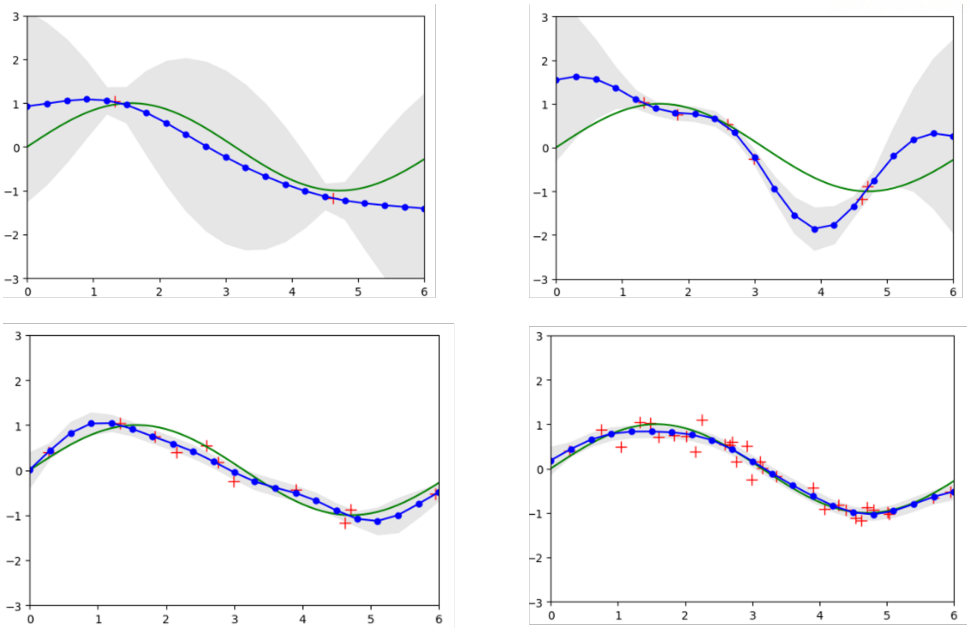


Figure 7: A visual representation of Gaussian Process Regression with data points.

Function-space view

A GP is a distribution over functions defined by its mean function $m(x)$and symmetric and positive semi-definite covariance function $k(x,x')$. In the following, we employ a squared-exponential kernel

$$\kappa(x,x') = \sigma^2 exp(\frac{(x-x')^T L^{-2}(x-x')}{2}) \tag{9}$$

The mean function and covariance function is defined as follows.

$$\begin{aligned} m(x) &= \mathbf{E}[f(x)] \\ k(x,x') &= \mathbf{E}[(f(x)-m(x))(f(x')-m(x')] \end{aligned} \tag{10}$$

As usual, the mean function is taken equal to zero. Therefore, we also suppose that our mean function is zero. Now, Assuming that we have n input-output pairs denoted by $\mathscr{D} = (x_i, y_i)_{i=1}^n$, we can represent $\mathrm{X} = [x_1, x_2, \cdots, x_n]^T$ and express the zero-mean gaussian process $f(\mathrm{X})$ as follows.

$$f(X) = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \in \mathbf{R}^n \tag{11}$$

And, for the new gaussian process $f(x_*)$, if it is also jointly gaussian, we can state jointly gaussian distribution as follows.

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \\ f(x_*) \end{bmatrix} = N \left(0, \begin{bmatrix} k(x_1,x_1) & \cdots & k(x_1,x_n) & k(x_1,x_*) \\ \vdots & \ddots & \vdots & \vdots \\ k(x_n,x_1,) & \cdots & k(x_n,x_n) & k(x_n,x_*) \\ k(x_*,x_1,) & \cdots & k(x_*,x_n) & k(x_*,x_*) \end{bmatrix} \right) \tag{12}$$

we also can rewrite the (12) as follows.

$$\begin{bmatrix} f \\ f(x_*) \end{bmatrix} = N\left(0, \begin{bmatrix} K(X,X) & K(X,x_*) \\ K(x_*,X) & K(x_*,x_*) \end{bmatrix}\right) \tag{13}$$

where $K(\cdot,\cdot)$ is gram matrix. Using the concept of the conditional distribution of a jointly gaussian random vector. We can get GP denoted as follows.

$$\begin{aligned} p(f_*|x_*,X,f) &\sim N(\mu_*,\sigma_*^2) \\ where\ \ \mu_* &= K(x_*,X)K(X,X)^-1f \\ \sigma_* &= k(x_*,x_*) - K(x_*,X)K(X,X)^-1K(X,x_*) \end{aligned} \tag{14}$$

where μ_* is mean function and σ_* is a covariance function.

Sparse GP

The Gaussian process has the advantage of being able to explain the regressed function and learn with less data. But it is limited to the real world when there is much data with increasing computational complexity. The computational complexity of GP regression strongly depends on the number of data points N. The quantity of data points(N) has a significant impact oncomputation complexity of GP regression. It requires mathcal $\mathcal{O}(n^3)$ operations for training and $\mathcal{O}(n^2)$ operations for evaluation when the predicted mean and variance are taken into account. Several approaches address this issue using sparse approximations, such as inducing variables (see figure 8). It is a method that can reduce computational costs much getting a set of inducing points that can essentially represent the entire original input space. There are many different approximation approaches in [31], but we focus on the variational formulation presented in [32].

Using this technique, the distance between the exact GP and a variational approximation is minimized. The inducing inputs are now variational parameters that are chosen to minimize the distance. The predictive gaussian is described as follows

$$\int p(\mathbf{z} \mid \mathbf{f}) p(\mathbf{f} \mid \mathbf{y}) d\mathbf{f}, \tag{15}$$

where $p(\mathbf{z} \mid \mathbf{f})$ are the conditional prior over the set of function points $\mathbf{z}$. It is same with the posterior GP which we want to represent. To represent (15) using a small m set of inducing variables f_m at the pseudo inputs X_m, we reformulate the integral as follows.

$$\begin{aligned} p(\mathbf{z} \mid \mathbf{y}) &= \int p(\mathbf{z} \mid \mathbf{f}_m, \mathbf{f})\, p(\mathbf{f} \mid \mathbf{f}_m, \mathbf{y})\, p(\mathbf{f}_m \mid \mathbf{y})\, d\mathbf{f} d\mathbf{f}_m \\ &= \int p(\mathbf{z} \mid \mathbf{f}_m)\, p(\mathbf{f}_m \mid \mathbf{y})\, d\mathbf{f}_m = q(\mathbf{z}, \mathbf{f}_m)\, d\mathbf{f}_m \end{aligned} \tag{16}$$

where $q(\mathbf{z}) = p(\mathbf{z} \mid \mathbf{y})$ and $p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{z}, \mathbf{f}_m, \mathbf{f})$ is the augmented joint and f_m are the function points taken from the GP prior. Additionally, we assume that f_m is a sufficient statistic for the parameter f, allowing that z and f are independent given f_m. Using (16), we can represent the approximate posterior GP mean

and covariance as follows

$$
\begin{aligned}
m_{\mathbf{y}}^{q}(\mathbf{x}) &= K_{\mathbf{x}m}K_{mm}^{-1}\mu \\
\mathrm{k}_{\mathbf{y}}^{q}(\mathbf{x},\mathbf{x}') &= k\left(\mathbf{x},\mathbf{x}'\right) - K_{\mathbf{x}m}K_{mm}^{-1}K_{m\mathbf{x}'} + K_{\mathbf{x}m}K_{mm}^{-1}AK_{mm}^{-1}K_{m\mathbf{x}'}
\end{aligned}
\tag{17}
$$

This equation represents the form of the sparse GP posterior which have $\mathcal{O}(nm^2)$ computational complexity. Detailed equation and proofs are explained in [32].

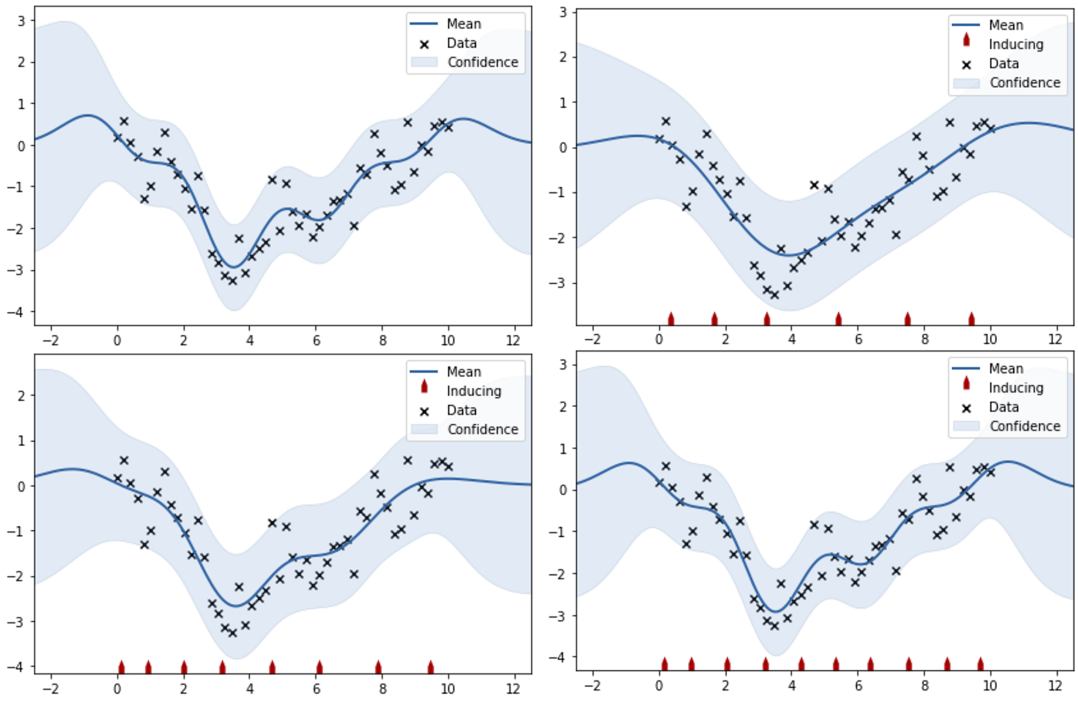


Figure 8: A visual representation of Sparse Approximation of Gaussian Process Regression (SGP).(Top, left) : Original Gaussian process Regression, (Top, right) : Sparse Gaussian process Regression with 6 inducing points, (Bottom, left) : Sparse Gaussian process Regression with 8 inducing points, (Bottom, right) : Sparse Gaussian process Regression with 10 inducing points

III Problem Formulation

Here, we take into account a scenario in which two racing vehicles are going head-to-head race on the racetrack. One is the controlled racing car, denoted by the ego vehicle (EV) and the other is the target vehicle (TV) which will be overtaken by the EV. We consider the following structure to describe the vehicle model.

$$\dot{x} = f(x,u) \approx f_n(x,u) + g_c(x,u) \tag{18}$$

where f_n are the nominal system dynamics of the vehicle. g_c is unmodeled residual dynamics. $x \in X \subset R^{n_x}$ represents the system state and $u \in U \subset R^{n_u}$ is the control actions with following vehicle state and input vectors.

$$x = [p_x, p_y\phi, v_x, v_y, \omega]^T, \quad u = [a, \delta]^T$$

3.1 Assumption Observation Model

As we don't know each other's racing strategy in advance, we make a few assumptions regarding TV's racing policy. Without loss of generality, the racing vehicle's physical specifications are assumed to be identical according to racing standards. Therefore, individual vehicles differ only in their driving style, i.e., control actions under a given racing track condition. In addition, to learn the TV's control action model, it is necessary to observe the control action of the TV. However, direct access to the TV's control action is not available. This leads us to utilize the EV's race data, which can provide access to the control action, in order to learn the inverse kino-dynamics model using GP or deep neural network [33]. We assume that we can estimate the TV's control action by learning an inverse kino-dynamics model from the TV's state history. Let us denote the inverse kino-dynamics model using neural Net as $I_{NN}^{-1}(\cdot)$ which outputs the control action from the vehicle state transition history. Then, the output control action is represented as follows.

$$u^{tv}(k) = [a^{tv}(k), \delta^{tv}(k)] \approx I_{NN}^{-1}(x^{tv}(k+1) - x^{tv}(k)) = [\hat{a}^{tv}, \hat{\delta}^{tv}] \tag{19}$$

where $[\hat{a}^{tv}, \hat{\delta}^{tv}]$ represent estimate the TV's control action from inverse kino-dynamics model. Now, we can estimate the control action though control actions of opponent vehicles are not directly available. And, we can use it to learn the control action prediction model with reasonable accuracy.

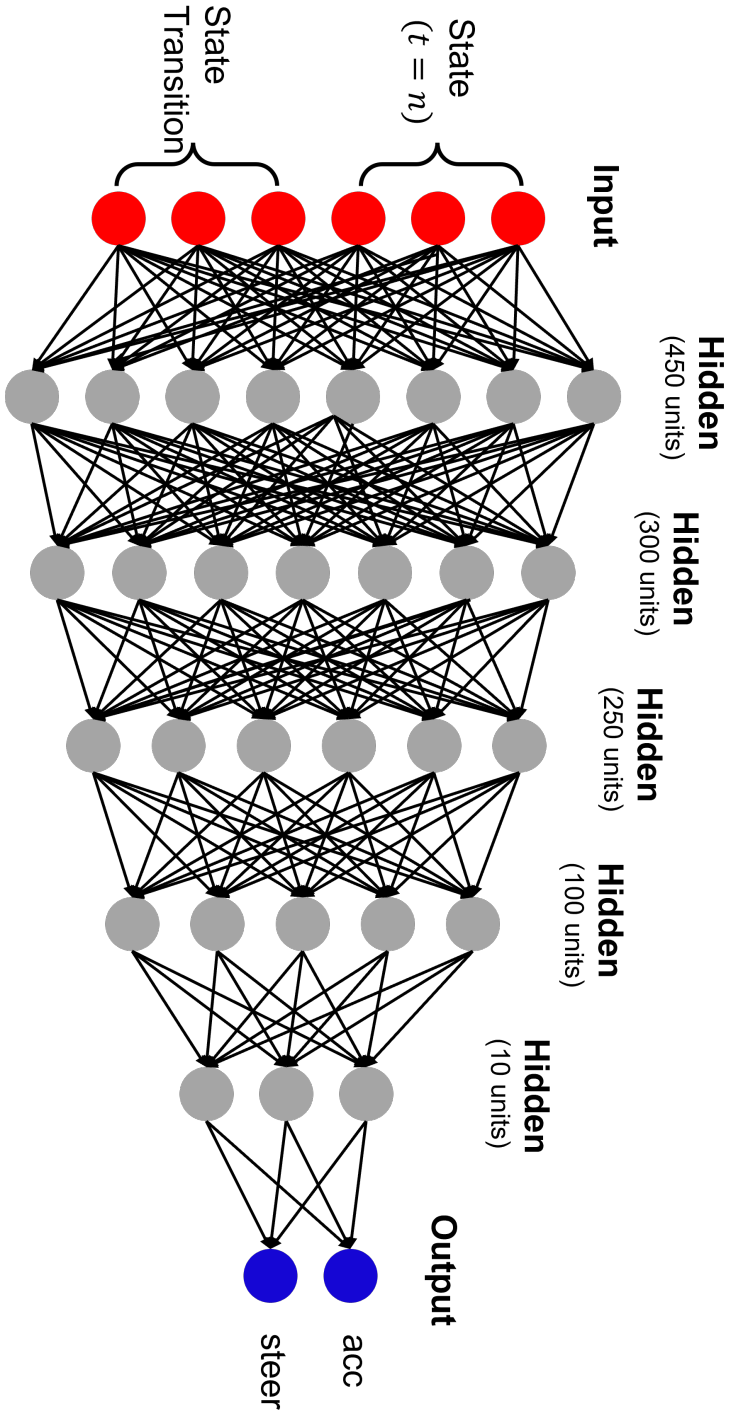


Figure 9: The structure of inverse kino-dynamics neural network based observation model. We use the linear and angular velocity and their transition as input and acceleration and steer angle as output. It consists of input, output and 5 hidden layers with each units.

3.2 MPC Formulation

Inspired by the model predictive contouring control(MPCC) formulation for racing vehicles presented in [34], we design the optimization problem for controlling the vehicle (1). The nonlinear projection of the vehicle's position onto the center line is taken into account when the designed problem plans a progress-optimal path control the vehicle. The corresponding MPCC formulation is as follows:

$$\min_{U} \quad \sum_{t=0}^{N-1} q_c e_c(x_k, u_k) + u_t^T R_u u_t^T + \Delta u_t^T \Delta R_u \Delta u_t^T - q_s \bar{s}_N \tag{20a}$$

$$s.t. \;\; x_{k+1} = f(x_k, u_k), \qquad t = 0, ..., N-1 \tag{20b}$$

$$x_0 = X_0, \; \bar{s}_0 = s(x_0) \tag{20c}$$

$$\bar{s}_{t+1} = \bar{s}_t + T_s v_{x,t}, \qquad t = 0, ..., N-1 \tag{20d}$$

$$x_t \in \mathbb{X}, \; u_t \in \mathbb{U}, \qquad t = 0, ..., N \tag{20e}$$

$$h(x_t, x_t^{tv}) < 0, \qquad t = 0, ..., N \tag{20f}$$

where $U = \{u_0, ..., u_{N-1}\}$, $\Delta u_t = u_t - u_{t-1}$. The objective of problem (20a) is to maximize approximations of track progress $\bar{s}_N$ and to minimize centerline deviation e_c. $\bar{s}$ means the distance vehicle drives along the track from the start position, and e_c means lateral deviation from the centerline. Each is controlled by penalizing the parameters q_c and q_s, respectively.Each is controlled by penalizing the parameters q_c and q_s, respectively. In addition, the control effort and its changes are minimized by R_u and ΔR_u. The above value of vehicle states which are represented by the parametric variable(s, e_y, e_{psi}) are constrained by the state set (20e). First, vehicle progress(s) is constrained by the track length(T_l) from 0 to (L). The lateral deviation from the centerline is constrained to the track width$(w_t rack)$, angle difference with the center line is constrained to $(-\pi, \pi)$. Also, the maximum velocity and minimum linear and angular velocity of the vehicle are also applied to states constrained. And the control action (acceleration, steering angle) can also be constrained by their maximum and minimum values. These values are calculated by the system identification vehicles. When TV is in front of the EV, another constraint is necessary for collision avoidance. The constraint (20f) is formulated based on EV's state x_t and TV's state x^{tv}. As we can't know the opponent's plans or their racing control policy, we, therefore, need a prediction model to generate TV's trajectories. It is generated by our prediction framework, which we will discuss in the following section.

3.3 Obstacle Avoidance Constraint

We make the collision avoidance constraints with the predicted TV's trajectory in Figure 10. As the major importance of proposed prediction algorithms is uncertainty, we modify the traditional constraints to leverage the uncertainties of predicted trajectories. We first define the original obstacle avoidance constraints using a circle and ellipse. We represent the EV's state as a set of four circles with a center and radius. Next, we represent the state of the TV as an ellipse with the center position (p_x, p_y, ψ) and axis length. The major and minor axis are represented using the vehicle's length and width to cover the vehicle's size. And, to cover the uncertainty of the prediction, the uncertainty magnitudes of the

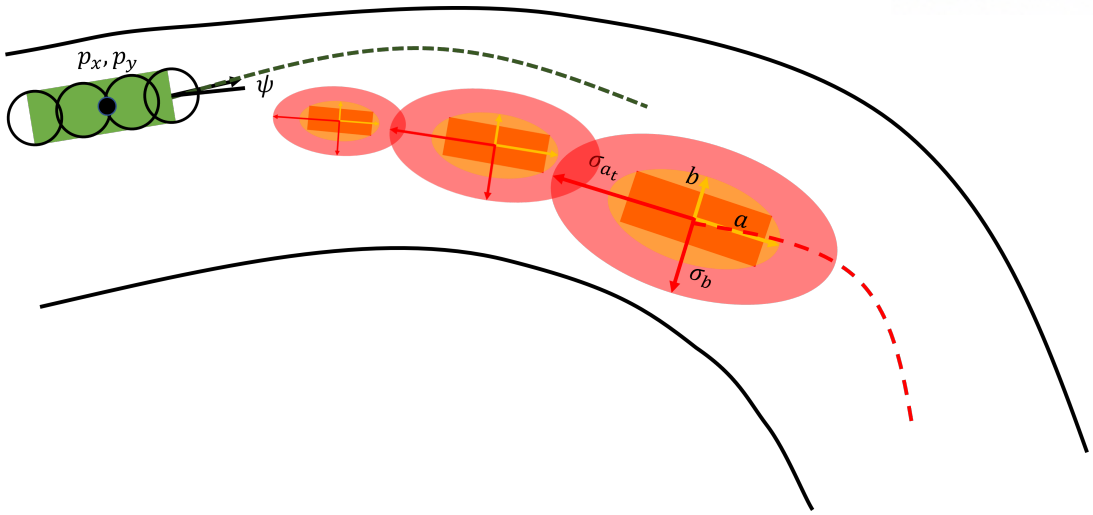


Figure 10: The collision avoidance constraints of mpcc problem. The ego vehicle is represented as a green box with four circles, and the target vehicle is represented as a red box with a yellow ellipse and a red ellipse. The yellow ellipse means the traditional constraints for collision avoidance, and the red ellipse represents the vehicle's state which incorporates the uncertainty of prediction with its confidence level.

longitudinal and lateral direction variance are added to the original ellipse's length. The constraints of the collision avoidance is represented as follows.

$$h_j(x^{ev}_{j,k}, x^{tv}_{j,k}) = 1 - \frac{((p_{x^{ev}_{j,t}} - p_{x^{tv}_{j,t}}\cos(\psi^{tv}_t) + (p_{y^{ev}_{j,t}} - p_{y^{tv}_{j,t}})\sin(\psi^{tv}_t))^2}{a^2} - \frac{((p_{x^{ev}_{j,t}} - p_{x^{tv}_{j,t}})\sin(\psi^{tv}_t) + (p_{y^{ev}_{j,t}} - p_{y^{tv}_{j,t}})\cos(\psi^{tv}_t))^2}{b^2} \tag{21}$$

To incorporate the uncertainties $(\sigma_{a_t}, \sigma_{b_t})$ of each direction, we represented the uncertainty using the variance as follows.

$$\begin{aligned} Var(a_t) &= Var(cos(e^{tv}_{\psi,t})s^{tv}_t + sin(e^{tv}_{\psi})e^{tv}_{y,t}) \\ &= cos^2(e^{tv}_{\psi})Var(s^{tv}_t) + sin^2(e^{tv}_{\psi})Var(e^{tv}_{y,t}) \\ Var(b_t) &= sin^2(e^{tv}_{\psi,t})Var(s^{tv}_t) + cos^2(e^{tv}_{\psi,t})Var(e^{tv}_{y,t}) \end{aligned} \tag{22}$$

Then we resize the major and minor axis lengths using the following equations. The gamma(γ) is chosen number which controls how much we would like to account for uncertainties for our safety constraints.

$$\begin{bmatrix} \sigma_{a_t} \\ \sigma_{b_t} \end{bmatrix} = \gamma \begin{bmatrix} \sqrt{Var(a_t)} \\ \sqrt{Var(b_t)} \end{bmatrix} (1 - \varepsilon_t) + \begin{bmatrix} a \\ b \end{bmatrix} \tag{23}$$

Finally, this resized value of the major and minor axis is applied to the equation (21) for collision avoidance safety which reflects the uncertainty

IV Algorithm Development

In this section, we will introduce the framework to learn the TV's trajectory prediction model. The frameworks are divided into *offline* and *online*. Afterward, we show how to exploit the prediction model to construct the MPC constraints for overtaking.

4.1 Learning Dynamics GP

To predict the accurate trajectory of TV, we need to learn the behavior model and incorporate accurate dynamics information for generating the trajectory. To do that, we learn two GP models: residual dynamics and a nominal behavior model. To begin, we learn the residual dynamics with the EV's single race data to infer the accurate model information for generating the trajectory. As the vehicle's physical specifications are the same, if we choose the TV's nominal vehicle model as (1), the residual dynamics of the two cars will be identical. We use the assumption that the model uncertainty and process noise w only impact the dynamic portions of the system, which are vx, vy, andomega, as the dynamics of the first three states are entirely determined by kinematic equations. Let us denote the modeling errors of dynamic parts at the discrete time step k as follows.

$$
\begin{aligned}
e_{v_x}(k) &= v_x(k+1) - \hat{v}_x(k+1) \\
e_{v_y}(k) &= v_y(k+1) - \hat{v}_y(k+1) \\
e_{\omega}(k) &= \omega(k+1) - \hat{\omega}(k+1)
\end{aligned}
$$

where $\hat{v}_x(k+1)$, $\hat{v}_y(k+1)$,and $\hat{\omega}(k+1)$ are propagated from the equation (1) with vehicle state and input $(x(k), u(k))$. Then the residual dynamics GP model is trained to represent the following vector-valued function.

$$
g_c(x(k), u(k)) \approx g_{gp}(x(k), u(k)) = [e_{v_x}(k),\ e_{v_y}(k),\ e_{\omega}(k)]^T \tag{24}
$$

where $g_c(\cdot,\cdot)$ is original residual dynamics model and $g_{gp}(\cdot,\cdot)$ is the learned residual GP model which approximates the modeling errors. A GP for a function g_{gp} is set to have zero prior mean and a squared exponential kernel function as (9). Based on the data of measurements $x(k+1)$ corresponding to the inputs $x(k), u(k)$, the predictions of GP at $x(k)$ are represented by the predictive mean μ and variance σ. Now, we can obtain the residual dynamics in form of gaussian distribution as follows.

$$
e_{v_x} \sim \mathcal{N}(\mu_{v_x}, \sigma^2_{v_x}), \quad e_{v_y} \sim \mathcal{N}(\mu_{v_y}, \sigma^2_{v_y}), \quad e_{\omega} \sim \mathcal{N}(\mu_{\omega}, \sigma^2_{\omega}) \tag{25}
$$

Note, we can incorporate the learned model to increase control performance by inferring the learned model to MPC formulation (8c). Some previous works incorporate learned models successfully in autonomous racing [29]. However, it can increase computational complexity for solving the optimization problem, making the system non-real-time. In other words, there is a trade-off between control performance and the feasibility of an optimization solution.

4.2 Learning Offline Control Action Prediction GP

We learn the behavior model which can predict the target vehicle's control action with reduced inputs regarding the vehicle state and race corner. To represent the race track and vehicle position, we use a curvilinear reference frame w.r.t track center line. The race track is represented using a curvature κ, which contains the full information of the track center line [35]. The vehicle's position is represented using the track progress s, lateral deviation from the centerline e_y, and its heading deviation from the centerline tangent angle e_ϕ. Now, we express the vehicle state as follows.

$$\hat{x} = [s, e_y, e_\phi, v_x, v_y, \omega]^T$$

where $\hat{x}$ represents the transformed vehicle state into a curvilinear reference. Motivated by [27] which learns the policy using spatial information of the race corner, we propose a feature space that may describe the characteristics of the track and the TV's current state. Then the nominal behavior prediction GP model is trained to represent the following vector-valued function.

$$u^{tv}(k) = [a^{tv}, \delta^{tv}] \approx u^{tv}_{gp}(z(k)) = [\hat{a}^{tv}, \hat{\delta}^{tv}] \tag{26}$$

where u^{tv}_{gp} represents the behavior prediction model which is the target model to train. $z(k) = [e_y(k), \kappa(k), v_x(k)]$ is the model input measurement vectors and $[\hat{a}^{tv}, \hat{\delta}^{tv}]$ is the model output measurement vectors which are estimated from the function (19). Like (25), we can obtain the control action prediction in form of the gaussian distribution as follows.

$$a^{tv} \sim N(\mu_{a^{tv}}, \sigma^2_{\hat{a}^{tv}}), \quad \delta^{tv} \sim N(\mu_{\delta^{tv}}, \sigma^2_{\delta^{tv}}) \tag{27}$$

Additionally, we used the sparse GP approximation for all the offline model training to efficiently infer the learned model.

4.3 Learning Online Control Action Prediction GP

Previously, the offline GP model for TV's behavior did not account for the interaction between vehicles; only vehicle state related to racing track information is taken into account. To reflect the interaction of vehicles and, therefore, refine the accuracy of predicted trajectory, we collect the data following the opponents and learn the behavior model online. We again represent the vehicle state into a curvilinear reference frame and define the feature space as the difference between EV's state and TV's state as follows.

$$\begin{aligned}
\Delta e_y(k) &= e^{tv}_y(k) - e^{ev}_y(k) \\
\kappa(k) &= \kappa^{tv}(k) \\
\Delta v_x(k) &= v^{tv}_x(k) - v^{ev}_x(k)
\end{aligned}$$

where the superscript (ev, tv) denotes the state of EV and TV, respectively. Δ denotes the state difference between EV and TV. Then the online behavior prediction GP model is trained to represent as (26) and outputs the distribution as (27) with different model input measurements $\Delta z(k) = [\Delta e_y(k), \kappa(k)\Delta v_x(k)]$.

Algorithm 1 Online process for trajectory prediction with uncertainties

0: Input : initial state of EV, TV : $x^{ev}(k)$, $x^{tv}(k)$

0: parameter : horizon length N, the number of samples M =0

0: $x^{tv,i}(k) \leftarrow x^{tv}(k) \quad \forall i = 1,...,M$;

0: **for** $t = 1,...,N-1$ **do**

0: Frame transformation from global to curvilinear : $\hat{x}^{tv,i}(k+t), \hat{x}^{ev}(k+t)$

0: $z^{tv,i}(k+t) \leftarrow$ Extract feature from $\hat{x}^{tv,i}(k+t)$;

0: $\Delta z^{tv,i}(k+t) \leftarrow$ Extract feature from $\hat{x}^{tv,i}(k+t), \hat{x}^{ev}(k+t)$;

0: Get distribution $u^{tv}_{off}(k+t) \sim \mathcal{N}(\mu^{tv,i}_{off}(k+t), (\sigma^{tv,i}_{off}(k+t))^2)$;

0: Get distribution $u^{tv}_{on}(k+t) \sim \mathcal{N}(\mu^{tv,i}_{on,k+t}, (\sigma^{tv,i}_{on}(k+t))^2)$;

0: $u^{tv}_{comb}(k+t) \leftarrow$ Do information fusion with (28);

0: **for** $i = 1,...,M$ **do**

0: Sample $u^{tv,i}_{comb}(k+t) \sim \mathcal{N}(\mu^{tv,i}_{comb}(k+t), (\sigma^{tv,i}_{comb}(k+t))^2)$;

0: Transfrom vehicle state curvilinear frame into global frame;

0: $x^{tv,i}(k+t+1) \leftarrow f_n(x,u) + g_{gp}(x,u)$ using the equation (1) and (25);

0: **end for**

0: **end for**

0: $\bar{x}^{tv}(k+t) \leftarrow \frac{1}{M}\Sigma_{i=1}^{M} x^{tv,i}(k+t)$;

0: $\Sigma^{tv}(k+t) \leftarrow \frac{1}{M-1}\Sigma_{i=1}^{M}(x^{tv,i}(k+t) - \bar{x}^{tv}(k+t))(x^{TV,i}(k+t) - \bar{x}^{tv}(k+t))^T$;

0: Output : $\bar{x}^{tv}(k+t), \Sigma^{tv}(k+t) \quad \forall t = 1,..,N-1$ =0

4.4 Overtaking with Online Prediction Refinement

We incorporate offline model predictions into online model predictions. It prevents the online model to make weird predictions in cases when it does not have enough data and improves the prediction according to TV's behavior as data collects. Since the learned model employing GP is a kind of Gaussian distribution, we may assume that the offline behavior model's control action follows $u_{off} \sim \mathcal{N}(\mu_{off}, \sigma^2_{off})$ and the online behavior model's control action follows $u_{on} \sim \mathcal{N}(\mu_{on}, \sigma^2_{on})$. Now, we can combine two gaussian distributions as follows.

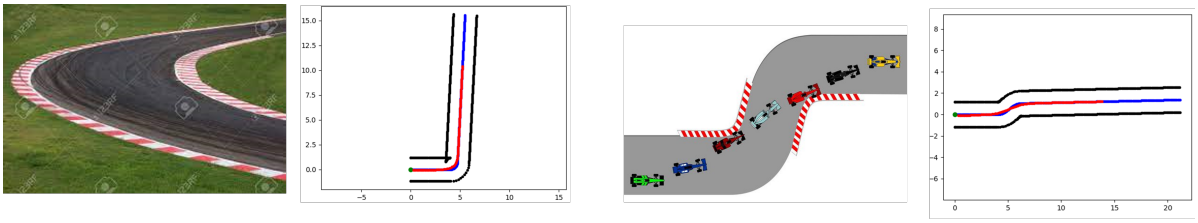
$$\mu_{comb} = \frac{\sigma^2_{off}\mu_{on} + \sigma^2_{on}\mu_{off}}{\sigma^2_{off} + \sigma^2_{on}}, \quad \sigma^2_{comb} = \frac{\sigma^2_{off}\sigma^2_{on}}{\sigma^2_{off} + \sigma^2_{on}} \tag{28}$$

We produce the projected trajectory by inferring it from the vehicle model based on the predicted control action. As our predictions are not precisely correct, we employ a sampling-based method to estimate the TV state and propagate the uncertainty using our model. The detailed procedure for online learning and trajectory generation is presented in Algorithm 1. Algorithm starts with each vehicle's state (x^{ev}_k, x^{tv}_k) and parameters N, M. Each vehicle's state transformed to a curvilinear frame (z^{ev}_k, z^{tv}_k) to predict the behavior with low dimension. Lines 5 and 6 evaluate the distribution of predicted behavior using the offline GP model and the online GP model, respectively. Information fusion (28) between two Gaussian distributions is performed in Line 7. Then, control actions are sampled from the resulting distribution

in Line 9. In Line 11, sampled control actions feed into the nominal vehicle dynamics and residual dynamics to compute the propagating state of TV. Note, the function $g_{gp}(\cdot,\cdot)$ is the learned residual dynamics using GP. This state generation procedure is repeated to generate the TV's trajectory of N steps. The N steps TV trajectory is generated by iterating this process for N-1 times. Finally, Line 16 computes the mean and standard deviation of M samples at each time step and returns the predicted trajectory along with uncertainty. From these process, we can generate the trajectory prediction with long horizon which are relected in to the MPC constraints to ensure collision avoidance.

V Numerical Simulation

5.1 Simulation Setup



(a) Curve (b) Chicane

Figure 11: Training track configuration which is generated randomly.

We consider a race scenario, where the ego vehicle intends to overtake the opponent. For both vehicle, MPCC policy in (8) is used with a sampling time of $T_s = 0.1s$ and a prediction horizon $N = 10$. Training data is accumulated through driving the test track which generated with different curvatures and configuration such as straight, curve, and chicane.(see Figure 11) Additionally, for the opponent vehicle, Additional cost is added to make the behavior of interaction such as blocking, lane keeping.

$$q_y(\Delta e_y)^2)/(1+(\Delta s)^2) \tag{29}$$

where $\Delta e_y, \Delta s$ are the difference of progress and deviation from the center line between TV and EV, and q_y is the tuning parameter that determines the aggressiveness of blocking. The difference trajectories of target by the different cost are represented in figure 12. When we set qy equal to zero, the target vehicle doesn't show a blocking motion like the left figure. And, when we set qy equal to one hundred, it shows moderate blocking motion. And, if we set the qy equal to 500, TV moves aggressively to block the ego vehicle.

No Blocking ($q_y = 0$) Moderate Blocking ($q_y = 100$) Aggressive Blocking ($q_y = 500$)

Figure 12: The different trajectory of target vehicle at the same scenario with different cost of target vehicle's blocking motion (left: the case where the cost is 0, middle : the case where cost is 100, right : the case where cost is 500.

Also, as we make the assumption of the observation model, we can estimate the TV's control action by learning an inverse kino-dynamics model from the TV's state history. We use the neural net in

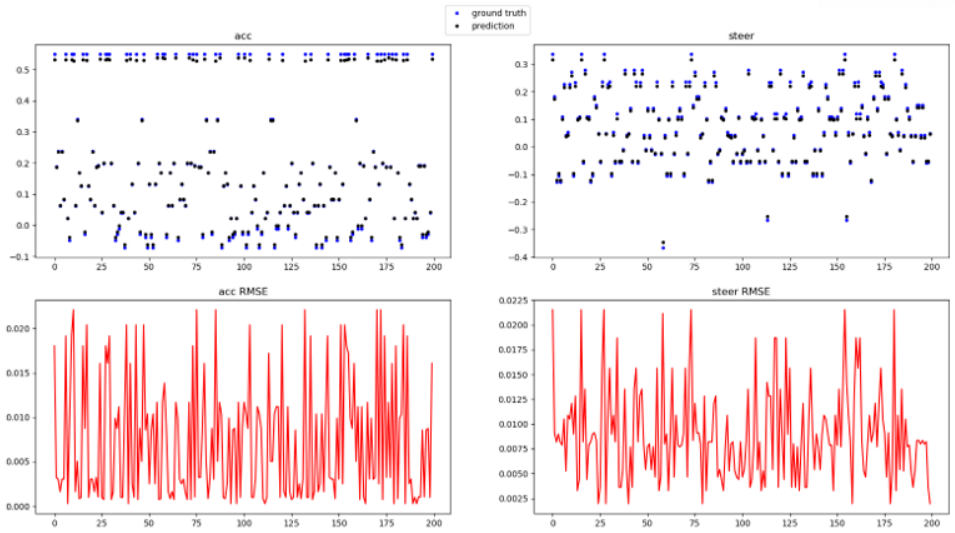
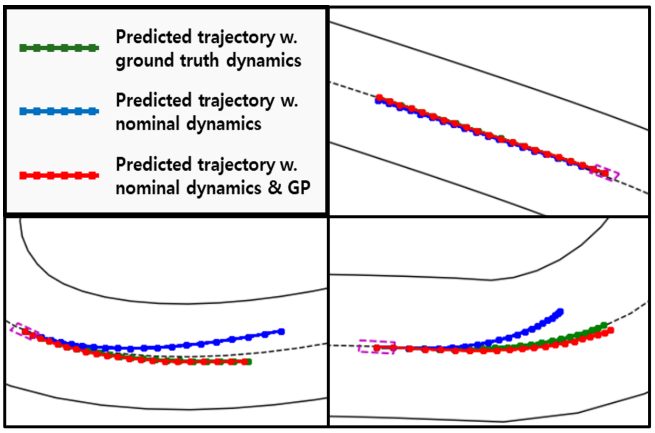


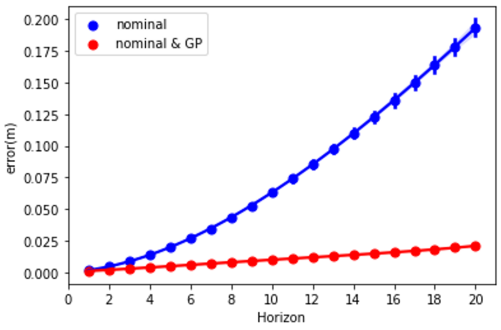
Figure 13: The result of Inverse kinodynamics model learning. Top : Inverse kino-dynamic prediction result. blue dot means ground truth of control action and black dot means estimated value of control action. Bottom : estimated error of each control action.

figure(9) to learn the inverse kino-dynamics model, and the result shows that we can estimate each control action with reasonable accuracy. And, Each GP for prediction model is trained with a GPyTorch [36] which is a widely used software platform for scalable GP inference. Its GPU acceleration allows for fast, effective computing of the trajectory via parallelization. The MPC problem is solved using a primal-dual interior point method through an off-the-shelf nonlinear optimization solver, Forces Pro [37, 38].

5.2 Discussions

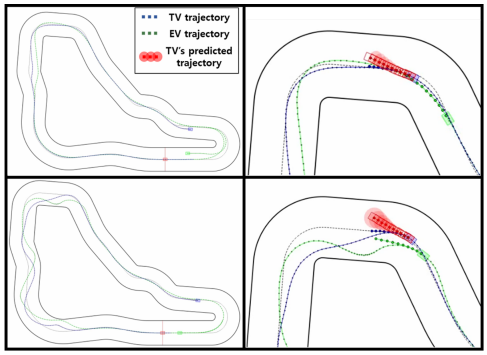


(a) Trajectory prediction without GP and with GP

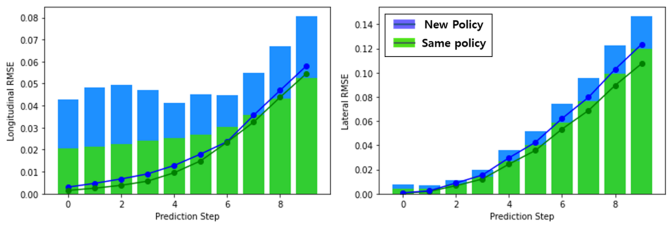


(b)Predicted error with Horizon

Figure 14: The result of the predicted trajectory without and with learned dynamics GP. (left) shows the predicted trajectory with and without GP model and ground trught dynamic model. (right) shows the predicted error with prediction horizon compared to predicted trajectory with ground truth dynamics.



(a) Predicted trajectory along the track
(Top : **TV's policy is same** with the training scenario,
Bottom: **TV's policy is different** with the training scenario)



(b) Line : Prediction Accuracy, Box : Prediction covariance
(Left : longitudinal error, Right : lateral error)

Figure 15: The result of predicted trajectory with offline learned control action prediction model. (a) : The trajectory history and predicted trajectory with uncertainty. (b):prediction Accuracy and prediction covariance.

Here, we simulate a scenario in which an EV tries to overtake a TV that is likely to block the EV's progress. To show the performance of proposed algorithms, we separate the algorithm step to 3 parts and show the performance of each part; Dynamics GP, Offline control action prediction GP, and Online control action prediction GP. First, in figure 14, we compare the predicted trajectory of target vehicle with and without learned dynamics GP. Green represents Target vehicle's ground truth plan, and blue represents the use of the existing vehicle model only, and red represents the use of the existing vehicle model and the learned dynamics together. We can see the red trajectory is much more similar to the green than the blue trajectory, which means the predicted trajectory with dynamics GP increases the prediction accuracy. Also, we can see the the predicted error is reduced significantly by about 10 times at last horizon.

In Figure 15, we show the result when we predict the control action using the offline model and learned dynamics model together. To show the prediction accuracy, we use two scenarios. Top of the Figure 15 shows the first scenario when the policy of the target vehicle is the same as the learned policy. Bottom of the Figure 15 shows the second scenario when the policy of the target vehicle is the same as the learned policy. Comparing the figure of two scenarios, the predicted accuracy gets lower when the Ego meets a new scenario. And it makes the ego vehicle move conservatively. In other words, It makes ego vehicle overtake Target inefficiently. Also, comparing the prediction accuracy and prediction covariance, it shows great performance of prediction when EV meets the learned policy. But, when the EV meets TV which uses new policy, the prediction error are higher than learned one and increase higher with increased prediction horizon. So, we can conclude that it is not enough to predict the trajectory using the offline learned model. In Figure 16, we display the predicted trajectory together with its associated uncertainty to demonstrate that the learned model is adapted well to the real TV's behavior. When the EV meets the TV at the first corner, because there is not enough data for the TV's behavior, the predicted uncertainty is large and EV is trying to follow the TV until there is enough place to overtake. As EV

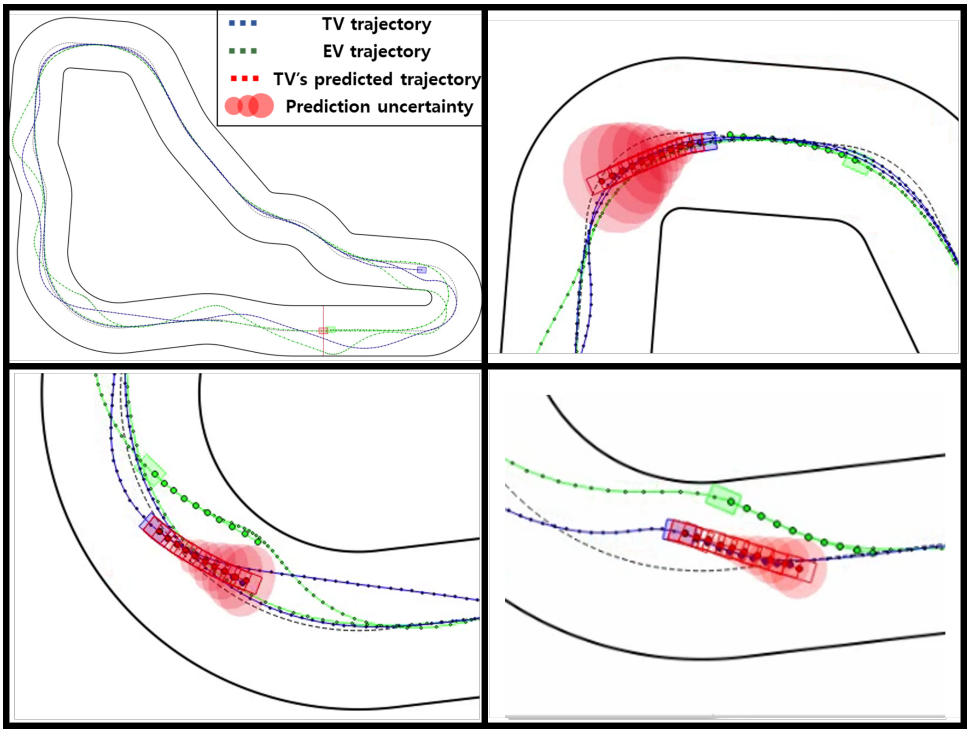


Figure 16: Predicted Trajectory with its covariance. (top,left) shows the trajectory of EV and TV during the race. (top,right) EV follows the TV while collecting information. (bottom) shows the corners where EV tries to overtake TV

collects data, GP for behavior prediction has less covariance and becomes accurate. In addition, EV attempts to overtake if it determines that there is enough space to pass through considering the predicted trajectory of TV and covariance. Furthermore, it can be seen that by increasing the efficiency of learning, sufficient data is collected in one or two turns, and the model is learned appropriately and quickly.

Also, in order to evaluate the accuracy of the predicted trajectory, we perform a Monte-Carlo (MC) simulation with 200 different starting positions and compare the results with the other methods shown in Fig 17. First, we investigate the longitudinal and lateral errors of nominal open-loop predictions with respect to the ground truth trajectory planned by the TV's actual control action policy. Fig 17 (left) shows that the proposed method outperforms other methods comparing the longitudinal and lateral errors. To quantify overtaking performance of the proposed framework, we compare the overtaking success rate and collision rate with other prediction methods in Fig 17(right). The simulation is conducted with the added objective (29) changing the q_y. The offline GP predictor [25] is trained with the objective $q_y = 50$. All the prediction module shows great performance when there is no blocking motion. However, when TV shows different behavior with the trained condition, Offline GP and NLP predictor shows poor performance. Whereas EV can't overtake the TV using the offline GP predictor even TV shows less aggressive blocking motion, EV overtakes the TV successfully. Remarkably, regardless of how aggressively the TV behaves, overtaking performance with the proposed predictor performs better than existing methods.

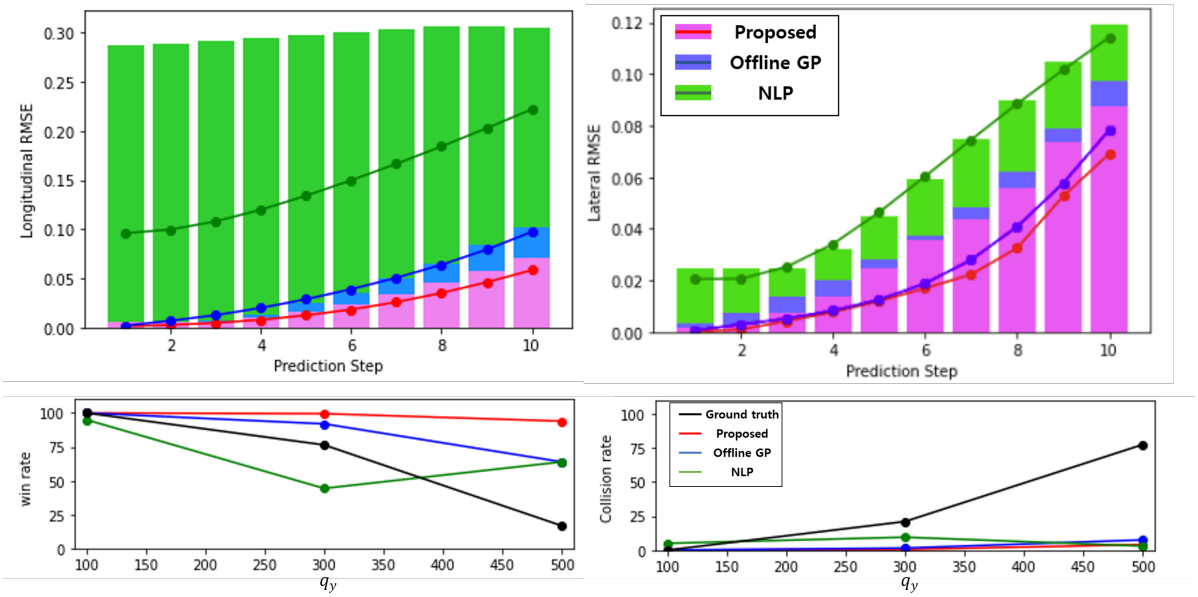


Figure 17: Top: Prediction accuracy of longitudinal(left) and lateral(right) RMSE. The length of the bar represents the average size of the uncertainty at each step of prediction. Bottom: Overtaking success rate and collision rate using the proposed method, and others: Offline GP, nonlinear programming(NLP) method.

VI Conclusion

In this paper, we have proposed a learning-based opponent vehicle trajectory prediction framework that combines offline and online GP models. The proposed method learns the vehicle model and nominal prediction model using the past available offline. When the vehicle starts to drive and collect data, the offline learned model is used to predict the opponent's trajectory for the safety constraints of MPC. At the same time, the vehicle learns a new prediction model online based on the collected data. To facilitate the online learning process, the original state data size is reduced by transforming into the feature state having a smaller dimension. Additionally, all the control actions from the learned prediction model are fed into the vehicle dynamics model to generate a dynamically feasible trajectory. In order to show the effectiveness of our methods, we have demonstrated that our framework predicts accurately and achieves higher win rates with less uncertainty in new environments compared to the existing learning-based prediction model through simulation.

References

[1] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," *IEEE Open Journal of Intelligent Transportation Systems*, 2022.

[2] S. He, J. Zeng, and K. Sreenath, "Autonomous racing with multiple vehicles using a parallelized optimization with safety guarantee using control barrier functions," in *IEEE International Conference on Robotics and Automation*, 2022.

[3] D. Kloeser, T. Schoels, T. Sartor, A. Zanelli, G. Prison, and M. Diehl, "Nmpc for racing using a singularity-free path-parametric model with obstacle avoidance," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14 324–14 329, 2020.

[4] D. S. González, J. S. Dibangoye, and C. Laugier, "High-speed highway scene prediction based on driver models learned from demonstrations," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 149–155.

[5] V. Karasev, A. Ayvaci, B. Heisele, and S. Soatto, "Intent-aware long-term prediction of pedestrian motion," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2543–2549.

[6] M. Wang, Z. Wang, J. Talbot, J. C. Gerdes, and M. Schwager, "Game-theoretic planning for self-driving cars in multivehicle competitive scenarios," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1313–1325, 2021.

[7] C. Jung, S. Lee, H. Seong, A. Finazzi, and D. H. Shim, "Game-theoretic model predictive control with data-driven identification of vehicle model for head-to-head autonomous racing," *arXiv preprint arXiv:2106.04094*, 2021.

[8] L. Sun, W. Zhan, and M. Tomizuka, "Probabilistic prediction of interactive driving behavior via hierarchical inverse reinforcement learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2111–2117.

[9] S. Ammoun and F. Nashashibi, "Real time trajectory prediction for collision risk estimation between vehicles," in *2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing*. IEEE, 2009, pp. 417–422.

[10] C. Barrios and Y. Motai, "Improving estimation of vehicle's trajectory using the latest global positioning system with kalman filtering," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 12, pp. 3747–3755, 2011.

[11] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 232–248, 2020.

[12] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.

[13] A. Houenou, P. Bonnifait, V. Cherfaoui, and W. Yao, "Vehicle trajectory prediction based on motion model and maneuver recognition," in *2013 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2013, pp. 4363–4369.

[14] S. Atev, G. Miller, and N. P. Papanikolopoulos, "Clustering of vehicle trajectories," *IEEE transactions on intelligent transportation systems*, vol. 11, no. 3, pp. 647–657, 2010.

[15] G. S. Aoude, V. R. Desaraju, L. H. Stephens, and J. P. How, "Driver behavior classification at intersections and validation on large naturalistic data set," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 724–736, 2012.

[16] X. Geng, H. Liang, B. Yu, P. Zhao, L. He, and R. Huang, "A scenario-adaptive driving behavior prediction approach to urban autonomous driving," *Applied Sciences*, vol. 7, no. 4, p. 426, 2017.

[17] C. Dong, Y. Zhang, and J. M. Dolan, "Lane-change social behavior generator for autonomous driving car by non-parametric regression in reproducing kernel hilbert space," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4489–4494.

[18] C. Dong, Y. Chen, and J. M. Dolan, "Interactive trajectory prediction for autonomous driving via recurrent meta induction neural network," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 1212–1217.

[19] S. Su, K. Muelling, J. Dolan, P. Palanisamy, and P. Mudalige, "Learning vehicle surrounding-aware lane-changing behavior from observed trajectories," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1412–1417.

[20] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1672–1678.

[21] H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, J. Schneider, D. Bradley, and N. Djuric, "Deep kinematic models for kinematically feasible vehicle trajectory predictions," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 563–10 569.

[22] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *European Conference on Computer Vision*. Springer, 2020, pp. 683–700.

[23] T. Brüdigam, A. Capone, S. Hirche, D. Wollherr, and M. Leibold, "Gaussian process-based stochastic model predictive control for overtaking in autonomous racing," *arXiv preprint arXiv:2105.12236*, 2021.

[24] Y. Yoon, C. Kim, J. Lee, and K. Yi, "Interaction-aware probabilistic trajectory prediction of cut-in vehicles using gaussian process for proactive control of autonomous vehicles," *IEEE Access*, vol. 9, pp. 63 440–63 455, 2021.

[25] F. L. Busch, J. Johnson, E. L. Zhu, and F. Borrelli, "A gaussian process model for opponent prediction in autonomous racing," *arXiv preprint arXiv:2204.12533*, 2022.

[26] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH journal*, vol. 1, no. 1, pp. 1–14, 2014.

[27] J. Bhargav, J. Betz, H. Zehng, and R. Mangharam, "Deriving spatial policies for overtaking maneuvers with autonomous vehicles," in *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*. IEEE, 2022, pp. 859–864.

[28] C. S. Vallon and F. Borrelli, "Data-driven strategies for hierarchical predictive control in unknown environments," *IEEE Transactions on Automation Science and Engineering*, 2022.

[29] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, 2019.

[30]

[31] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[32] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Artificial intelligence and statistics*. PMLR, 2009, pp. 567–574.

[33] X. Xiao, J. Biswas, and P. Stone, "Learning inverse kinodynamics for accurate high-speed off-road navigation on unstructured terrain," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 6054–6060, 2021.

[34] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[35] Ph.D. dissertation.

[36] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration," *Advances in neural information processing systems*, vol. 31, 2018.

[37] A. Domahidi and J. Jerez, "Forces professional," Embotech AG, url=https://embotech.com/FORCES-Pro, 2014–2019.

[38] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "Forces nlp: an efficient implementation of interior-point... methods for multistage nonlinear nonconvex programs," *International Journal of Control*, pp. 1–17, 2017.

Acknowledgements

먼저 존경하는 교수님, 항상 누구보다 많은 열정으로 저를 생각하며 이끌어주신 교수님께 감사를 드립니다. 항상 연구의 방향성을 함께 고민해주시고 항상 학자의 자세와 목표를 향한 열정을 보여주셔서 석사 과정중에 많은 것을 배울 수 있었고 존경스럽습니다. 또한, 제가 부족함이 있음에도 배려해주시고 이끌어주셔서 감사를 드립니다. 앞으로도 교수님의 열정을 바탕으로 하시는 일 모두 번창하고 건강하시길 바랍니다. 또한, 바쁘심에도 불구하고 흔쾌히 석사 학위 심사를 맡아주신 오현동 교수님과 손홍선 교수님께도 특별한 감사를 표하고 싶습니다. 마지막까지 좋은 말씀과 가르침 감사합니다.

다음으로 2년이라는 길다면 길고 짧다면 짧은 시간동안 제 옆에서 늘 도와주고 가르쳐주고 좋은 시간 보내면서 힘이되어준 HMCL 맴버들에게 정말 정말 고맙다는 말을 남깁니다. 먼저 함께 연구 주제를 고민하고 항상 지지해주고 밤낮 가리지 않고 노력해준 호진이형에게 감사를 표합니다. 처음 연구실을 와서 아무것도 모를때 항상 옆에서 차근차근 한개씩 알려주고 적응을 할 수 있도록 도와준 호정이, 은민이, 형철이에게도 감사를 표합니다. 거의 모든 수업도 함께 듣고, 대회 준비를 하며 평일, 주말 가리지 않고 출근해 힘든일도 즐거운일도 함께 나눌수 있어 늘 재밌고 좋은기억을 가지게 해준 민우, 일승이, 영임이에게도 감사를 표합니다. 늦게 들어와 적응하기에도 바빳을텐데에도 함께 잘따라와주고 열정을 다해준 현빈, 상현, 광록이에게 감사를 표합니다.다른 팀이지만 항상 열심히 노력해주고 안보이는 곳에서 많은 일들을 처리해준 형준이형 동화형, 강민, 상훈, 주상이에게도 감사를 드립니다.

마지막으로, 대학원 생활동안 저에대한 굳건한 믿음과 지지로 잘 마무리 할 수 있도록 도와주신 아버지와 어머니에게 감사합니다

Acknowledgements

First of all, I would like to thank my advisor for always leading me with more passion than anyone else. I was able to learn a lot during the master's course and respect you for always thinking about the direction of research together and always showing the attitude and passion for the goal of the scholar. Also, thank you for being considerate and leading me even though I am lacking. I hope everything you do based on your passion will prosper and be healthy in the future. Also, I would like to express my special gratitude to Professor Oh Hyondong and Professor Son Hung-sun for willingly reviewing my master's degree despite their busy schedule. Thank you for your good words and teaching until the end.

Next, I would like to say thank you to HMCL members who always helped me, taught me, and had a good time for two years. First of all, I would like to express my gratitude to Hojin for thinking about the research topic together, always supporting it, and working hard day and night. I would also like to thank Hojung, Eunmin, and Hyungcheol for helping me adapt to the situation and teaching me one by one when I first came to the lab. I also thank Minu, Ilseung, and Youngim for always having fun and good memories by taking almost all classes together, preparing for the competition, going to work on weekdays and weekends, and sharing hard and fun things together. I would like to express my gratitude to Hyeonbin, Sanghyeon, and Gwangrok for following along well and doing their best even though they must have been busy adjusting late. I would also like to thank Hyeongju, Donghwa, Kangmin, Sanghoon, and Jusang for always working hard in a different team and taking care of many things in an invisible place.

Lastly, I would like to thank my father and mother for helping me finish my graduate school life with strong faith and support.