공학석사학위논문

학습 기반 모델 예측 제어 기법을 이용한 스키드 조향 차량의 경로 제어

Path Tracking for a Skid-steer Vehicle using Learning-based Model Predictive Control

2017 년 2 월

서울대학교 대학원

기계항공공학부

김 태 완

Path Tracking for a Skid-steer Vehicle using Learning-based Model Predictive Control

A Thesis

by

TAEWAN KIM

Presented to the Faculty of the Graduate School of
Seoul National University
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE

School of Mechanical & Aerospace Engineering

Seoul National University

Supervisor : Professor H. Jin Kim

February 2017

학습 기반 모델 예측 제어 기법을 이용한 스키드 조향 차량의 경로 제어

Path Tracking for a Skid-steer Vehicle using Learning-based Model Predictive Control

지도교수 김 현 진

이 논문을 공학석사 학위논문으로 제출함

2017 년 2 월

서울대학교 대학원

기계항공공학부

김 태 완

김태완의 공학석사 학위논문을 인준함

2017 년 2 월

위 원 장 김 유 단 [illegible]

부위원장 김 현 진 [illegible]

위 원 박 찬 국 [illegible]

to my

MOTHER and FATHER

with love

Abstract

Path Tracking for a Skid-steer Vehicle using Learning-based Model Predictive Control

Taewan, Kim
Department of Mechanical & Aerospace Engineering
The Graduate School
Seoul National University

Skid-steer vehicle can generate a large traction force, which is especially good for navigation on a rough terrain. However, the turning motion is so sensitive to slippage effect that designing a controller is still a challenging problem. Also, the motion of the vehicle is affected not only by wheel motion, but also by the road properties and the characteristics of wheel control. With this in mind, we employ a model predictive control (MPC) with an on-line model learning. The velocity model, which represents the relationship between true vehicle velocity and input command, is learned with an on-line sparse Gaussian process (GP). The on-line sparse GP can reduce the computational complexity of GP and also consistently update the model from the driving data. Finally, combining with MPC makes it possible to generate an optimal policy based on the learned model. Experiments are conducted to test the tracking performance of a skid-steer robot at the indoor and the outdoor environment. The results show the more reliable performance than the method based on a conventional model with parameter adaptation.

Keyword : Path tracking, skid-steer vehicle, model predictive control, sparse Gaussian process.

Student Number : 2015-20766

Table of Contents

List of Figures

1 Introduction

Skid-steer vehicle has no steering devices, so it turns by the difference of tire force between left and right wheels. This characteristic gives many advantages such as high traction force and simple mechanical structure. Thanks to these, skid-steer vehicles are especially good for navigation and exploration on a rough terrain. Although many of research have studied autonomous navigation of skid-steer vehicles, it is still a challenging problem to design a controller for the skid-steer vehicle, because they are highly prone to slippage when generating turning motion.

1.1 Literature review

One way of controlling the skid steer vehicle is to control the tire force based on the the vehicle dynamics [1], [2], [3]. After calculating the desired longitudinal force at each wheel through the tire force distribution, this method focuses on the control of tire force by torque control or slip control. Torque control, however, is not proper for mobile robots whose wheels usually do not contain the torque sensor. When it comes to slip control, generating the desired tire force depends on the tire force sensor or the specific tire model such as pacejka tire model [4]. Obtaining various parameters for such models requires complex experiments, so it is not suitable for small mobile robots.

Alternatively, instantaneous center of rotation (ICR) has been widely used to control the skid-

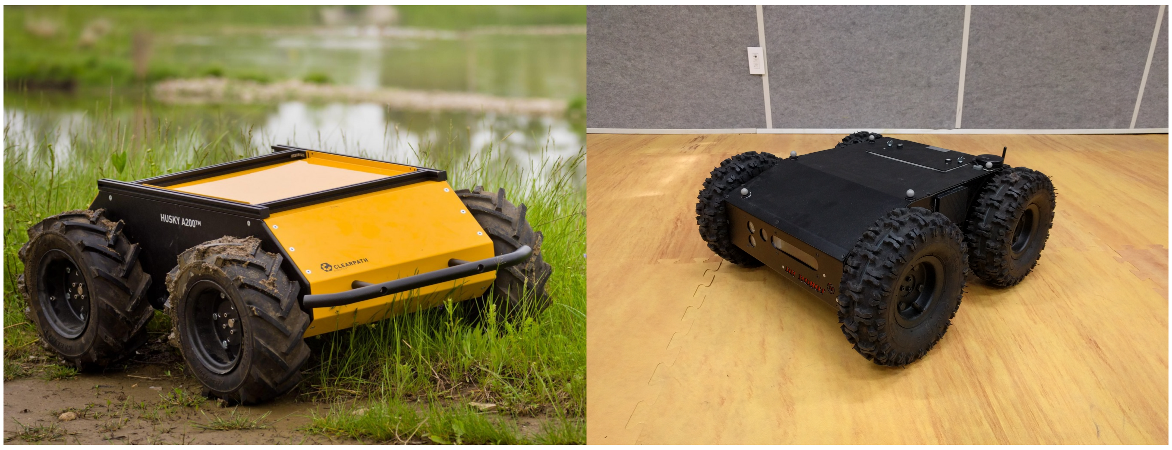

Figure 1.1: Examples of skid-steer mobile robot platform: Clearpath Robotics Husky A200 and Dr. Robot Jaguar.

steer vehicle and the tracked vehicle [5], [6], [7]. After estimating the position of ICR for left, right treads and vehicle body with respect to the body fixed coordinate, this method makes it possible to predict the kinematic motion of the vehicle. Although it gives reasonable performance during the low-speed driving, it can become unsatisfactory at relatively high-speed because the position of ICR changes dramatically when the speed of vehicle is relatively fast.

Recently some research focused on the velocity model which represents the relationship between true velocity of the vehicle and the input command for velocity [8], [9], [10]. They utilized the general slip model which is composed of deterministic model and several undetermined parameters. After fitting the parameters based on the driving data, they used the model-based controller for navigation.

The aforementioned methods constructed a explicit model by utilizing prior knowledge of the vehicle to increase the accuracy of the model. The motion of skid-steer vehicle, however, cannot be easily represented by the explicit model because the model is affected by various elements such as the vehicle body dynamics, tire-road property and the characteristic of wheel angular velocity control. Even if additional undetermined parameters are employed, the structure of the model equation is still required. In stead of using a parametric model, we tackle this problem with the learning-based approach. In detail, we use the Gaussian process (GP) to represent the velocity model of the skid-steer vehicle. GP is a Bayesian non-parametric model so that it can represent the nonlinear characteristic and uncertainty of the model. In addition, GP is often not so sensitive to the initial setting of hyper-parameters. For these reasons, the GP-based approach has been widely used for various applications.

[11] and [12] applied a learning-based approach to a path-repeating problem for the wheel mobile robots. They constructed the model error of discrete kinematics, i.e the difference between the true model and an initial estimated model, by using GP. After driving on a given trajectory, they used GP to represent the model error along the path and the input. Even though this method can deal with the model uncertainty, there are some drawbacks. One of them is that this GP error model can be only used for the same trajectory where training data was obtained. This is because the discrete kinematics contains the position and the input of vehicle as input features. In addition, standard GP suffers from computational complexity especially when predicting the future evolution in the model-based controller. On the other hand, we focus on the velocity model which does not contain the position as an input. Our approach, therefore, can deal with diverse trajectories on which the vehicle has not driven. In addition, we employ the sparse GP to overcome the computational complexity of GP, and update the sparse GP on-line to deal with the model uncertainty.

In order to track the path, we combine the learned-model with model predictive control (MPC), which is one of the most widely investigated controllers for autonomous vehicle. The sparse GP allows to address sensitivity of MPC performance to the model error. We solve the optimization problem using iterative linear quadratic regulator (ILQR). ILQR, which is a type of differential dynamic programming, is fast enough for real-time control.

1.2 Thesis contribution

Our main contributions can be summarized as follows:

1. We try to represent the motion of the skid-steer vehicle not by a explicit model but by a learning-based approach. Since the vehicle kinematics is affected by various elements such as terrain parameters or motor control, the learning-based model with driving data can be more accurate than the explicit model.

2. We reduce the computational complexity of GP by using the sparse GP and employ the on-

line learning framework to cope with model error. By reducing computational complexity, we can efficiently satisfy the control frequency requirement of MPC, and execute real-time experiments. Plus, we can consistently update the model error based on the on-line learning in order to reduce the model error.

3. Combining model learning with MPC framework is validated through experiments. Through experiments, we show that our approach shows the better performance conventional methods.

1.3 Thesis outline

The rest of this paper is organized as follows: the kinematics of the skid-steer vehicle and the on-line sparse GP are provided in section 2. In section 3, the details of MPC are presented. Experimental results of MPC with the on-line learning are shown in Section 4. Conclusion follows in Section 5.

2
On-line sparse Gaussian process for velocity model

In this section, we first provide the kinematic model for the skid-steer vehicle and the structure of the velocity model which is our target function for GP regression. Then, the overview of the sparse GP will follow in the next two sections. Among several types of the sparse GP which were well summarized in [13], here we employ the on-line sparse GP utilized in [14] and [15].

2.1 Kinematic model

The kinematic model of the vehicle shown in Fig. 2.1 can be written as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \overbrace{\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}}^{R(\theta)} \mathbf{v}(v_{xcom}, v_{\theta com})$$

where x, y, θ represent the position and yaw angle, and v_{xcom}, $v_{\theta com}$ mean the input command for forward and turning. R represents the rotation matrix between inertial frame and body-fixed frame.

Here, $\mathbf{v}(\cdot)$ is the velocity model. We can see that the velocity model describes the relationship of true velocity and the input command. As an example, the velocity model for an ideal differential

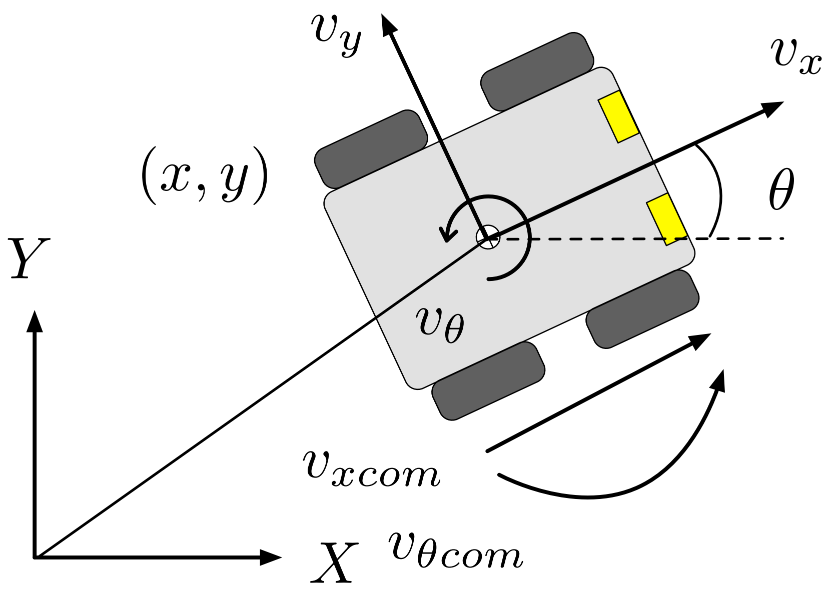

Figure 2.1: Kinematics of a skid-steer vehicle and mobile robot platform used in experiments. We want to approximate the dynamic relationship of input command $v_{xcom}, v_{\theta com}$ and true velocity v_x, v_y, v_θ using GP.

drive (IDD) model, which assumes that the slip does not occur, is written as follows

$$\mathbf{v}_{idd}(v_{xcom}, v_{\theta com}) = \begin{bmatrix} v_{xcom} \\ 0 \\ v_{\theta com} \end{bmatrix}.$$

The general kinematic slip (GKS) model is dealt in [16]. This model considers that the slip effect may lead to velocity changes. The equation of the GKS model is written as

$$\begin{aligned}
\mathbf{v}_{gks}(v_{xcom}, v_{\theta com}) &= \begin{bmatrix} v_{xcom} \\ 0 \\ v_{\theta com} \end{bmatrix} + C(v_{xcom}, v_{\theta com}) \cdot \alpha \\
C(v_{xcom}, v_{\theta com}) &= \begin{bmatrix} \mathbf{c}_x & & \\ & \mathbf{c}_y & \\ & & \mathbf{c}_\theta \end{bmatrix} \\
\mathbf{c}_x &= \begin{bmatrix} v_{xcom} & |v_{\theta com}| & v_{xcom}|v_{\theta com}| \end{bmatrix} \\
\mathbf{c}_y &= \begin{bmatrix} v_{xcom} & v_{\theta com} & v_{xcom}v_{\theta com} \end{bmatrix} \\
\mathbf{c}_\theta &= \begin{bmatrix} v_{xcom} & v_{\theta com} & v_{xcom}v_{\theta com} \end{bmatrix},
\end{aligned}$$

where the vector α is composed of nine parameters.

Meanwhile, we try to approximate the velocity model by using GP. Through the following experiments, we show that GP model has better performance than IDD and GKS model when combined with MPC.

2.2 Sparse Gaussian process

GP regression tries to learn the unknown velocity model $\mathbf{v}(\cdot)$ with input command $u = (v_{xcom}, v_{\theta com})$. The main downside of GP is computational complexity. To overcome this drawback, the sparse GP was developed. The sparse GP makes it possible to reduce the runtime by making assumptions about a prior distribution. In this section, we briefly summarize GP and the sparse GP.

To predict the test point, we first acquire the driving data $(u_1, z_1) \cdots (u_n, z_n)$, where z_i, the measurement output of the velocity corresponding to the input u_i. We assume that the measurement output z_i is corrupted by white noise with variance σ_n^2 from the true velocity of the vehicle

$\mathbf{v}$. For shorthand notation, we denote the set of the input command data as U. Then, we assume that the prior distribution, $\mathbf{V} = \mathbf{v}(U)$ and $\mathbf{V}_* = \mathbf{v}(U_*)$ have a joint Gaussian distribution with zero-mean written as

$$\begin{aligned}\begin{bmatrix}\mathbf{V} \\ \mathbf{V}_*\end{bmatrix} &\sim \mathcal{N}\left(\begin{bmatrix}0 \\ 0\end{bmatrix}, \begin{bmatrix}k(U,U) & k(U,U_*) \\ k(U_*,U) & k(U_*,U_*)\end{bmatrix}\right) \\ &= \mathcal{N}\left(\begin{bmatrix}0 \\ 0\end{bmatrix}, \begin{bmatrix}K_{vv} & K_{v*} \\ K_{*v} & K_{**}\end{bmatrix}\right),\end{aligned}$$

where u_* and U_* mean the trial input command and the set of them. In this paper, the squared-exponential kernel function k is used, which is explained in [17]. The posteriori distribution of $\mathbf{V}_*$ is derived as follows:

$$\begin{aligned}p\left(\mathbf{V}_*|\mathbf{V}\right) &= \mathcal{N}\left(\mu_*, \Sigma_*\right) \\ \mu_* &= K_{*v}\left(K_{vv} + \sigma_n^2 I\right)^{-1}\mathbf{z} \\ \Sigma_* &= K_{**} - K_{*v}\left(K_{vv} + \sigma_n^2 I\right)^{-1} K_{v*},\end{aligned}$$

where $\mathbf{z}$ is the vector of measurement output data.

The sparse GP starts with the assumption that $\mathbf{V}$ and $\mathbf{V}_*$ are conditionally independent when the inducing input points U_c and corresponding function $\mathbf{V}_c$ are given. This assumption can be written as,

$$p\left(\mathbf{V}, \mathbf{V}_*, \mathbf{V}_c\right) = p\left(\mathbf{V}|\mathbf{V}_c\right) p\left(\mathbf{V}_*|\mathbf{V}_c\right) p\left(\mathbf{V}_c\right).$$

Then, using the product of Gaussian exponential, the priori distribution of training points, test points and inducing points become

$$\begin{aligned}p\left(\mathbf{V}, \mathbf{V}_*, \mathbf{V}_c\right) &= \mathcal{N}\left(\begin{bmatrix}0 \\ 0 \\ 0\end{bmatrix}, \begin{bmatrix}K_{vv} & M_{v*} & K_{vc} \\ M_{*v} & K_{**} & K_{*c} \\ K_{cv} & M_{c*} & K_{cc}\end{bmatrix}\right) \\ M_{ab} &= K_{ac}K_{cc}^{-1}K_{cb}.\end{aligned}$$

Furthermore, with the fully independent training conditional algorithm (FITC), we also assume that all values of the function $\mathbf{V}$ are pairwise independent given $\mathbf{V}_c$, which can be written as,

$$p\left(\mathbf{v}_i, \mathbf{v}_j | \mathbf{V}_c\right) = p\left(\mathbf{v}_i | \mathbf{V}_c\right) p\left(\mathbf{v}_j, | \mathbf{V}_c\right).$$

Based on the both assumptions, the prior distribution $p(\mathbf{V}, \mathbf{V}_*, \mathbf{V}_c)$ can be calculated as follows:

$$\begin{aligned} p\left(\mathbf{V}, \mathbf{V}_*, \mathbf{V}_c\right) &= p\left(\mathbf{v}_1 | \mathbf{V}_c\right) \cdots p\left(\mathbf{v}_n | \mathbf{V}_c\right) p\left(\mathbf{V}_*, | \mathbf{V}_c\right) p\left(\mathbf{V}_c\right) \\ &= \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_{v_1 v_1} & \ldots & M_{v_1 v_n} & M_{v_1 *} & K_{v_1 c} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ M_{v_n v_1} & \ldots & K_{v_n v_n} & M_{v_n *} & K_{v_n c} \\ M_{* v_1} & \ldots & M_{* v_n} & K_{**} & K_{*c} \\ K_{c v_1} & \ldots & K_{c v_n} & K_{c*} & K_{cc} \end{bmatrix} \right) \end{aligned}$$

The remaining step is to predict the test input command u_* and the posterior distribution of $\mathbf{V}_*$. We first find the posterior distribution of $\mathbf{V}_c$ using a new prior $p(\mathbf{V}, \mathbf{V}_*, \mathbf{V}_c)$. The result is

$$\begin{aligned} p\left(\mathbf{V}_c | \mathbf{V}\right) &= \mathcal{N}\left(\mu_c, \Sigma_c\right) \\ \mu_c &= K_{cc} \Delta^{-1} K_{cv} \left(\Lambda_{vv} + \sigma_n^{-2} I\right)^{-1} \mathbf{z} \\ \Sigma_c &= K_{cc} \Delta^{-1} K_{cc} \\ \Delta &= K_{cc} + K_{cv} \left(\Lambda_{vv} + \sigma_n^{-2} I\right)^{-1} K_{vc} \\ \Lambda_{vv} &= diag\left(K_{vv} - K_{vc} K_{cc}^{-1} K_{cv}\right). \end{aligned}$$

where $\mathbf{z}$ is the set of measurement output, $(z_1, \cdots, z_n)$. Then, the posterior distribution of $\mathbf{V}_*$ can be derived through marginalization as

$$\begin{aligned} p\left(\mathbf{V}_* | \mathbf{V}\right) &= \int p\left(\mathbf{V}_* | \mathbf{V}_c\right) p\left(\mathbf{V}_c | \mathbf{V}\right) d\mathbf{V}_c \\ \mu_* &= K_{*c} K_{cc}^{-1} \mu_c \\ \Sigma_* &= K_{**} - K_{*c} K_{cc}^{-1} \left(K_{cc} - \Sigma_c\right) K_{cc}^{-1} K_{c*}. \end{aligned}$$

Here the size of the matrix K_{cc}^{-1} is $m \times m$, and m refers to the number of inducing input points. The computational cost at a prediction of spare GP is $\mathcal{O}(m^3)$. On the other hand, the cost of nominal GP is $\mathcal{O}(n^3)$. Since the number of inducing input points is fewer than that of input of training data, the computational complexity of the sparse GP is much reduced compared with that of nominal GP model.

2.3 On-line updating

This section describes how to update the posterior distribution of the $\mathbf{V}_c$ with a new measurement (u_{n+1}, z_{n+1}). We first predict the posterior distribution of $\mathbf{v}_+ = \mathbf{v}(u_{n+1})$ as follows,

$$\begin{aligned} \mu_+ &= K_{+c}K_{cc}^{-1}\mu_c \\ \Sigma_+ &= K_{++} - K_{+c}K_{cc}^{-1}\left(K_{cc} - \Sigma_c\right)K_{cc}^{-1}K_{c+}. \end{aligned} \tag{2.1}$$

With the proper arrangement, the updated posteriori distribution $\mathbf{V}_c^{n+1}$ can be derived with respect to the old one $\mathbf{V}_c$ as follows

$$\begin{aligned} \mu_c^{n+1} &= \mu_c + \left(K_{+c}K_{cc}^{-1}\Sigma_c\right)^T\left(\Sigma_+ + \sigma_n^2\right)^{-1}\left(z_{n+1} - \mu_+\right) \\ \Sigma_c^{n+1} &= \Sigma_c - \left(K_{+c}K_{cc}^{-1}\Sigma_c\right)^T\left(\Sigma_+ + \sigma_n^2\right)^{-1}K_{+c}K_{cc}^{-1}\Sigma_c. \end{aligned} \tag{2.2}$$

The hyperparameters in the kernel function are learned by maximizing the log-likelihood of the training outputs [17]. We simply fix the hyperparameters during on-line updating because of computational time.

3 Model predictive control

We apply MPC to generate the optimal policy making vehicle to track the path. At every step, MPC calculates the optimal input sequence $\mathbf{u}_{0:N-1}$ which minimizes the given cost function. Then, the action is set to the first input among this input sequence. After applying the first action to the vehicle, the entire process is repeated until the end. This repeating process enables the vehicle to maintain the low path error, even when the vehicle slightly diverges to the given trajectory. In this section, we provide the overview of ILQR and the cost formulation. The details of ILQR can be found in [18].

3.1 Iterative linear quadratic regulator

To find a local optimal solution given the cost formulation, we use the ILQR, which is a variant of classic differential dynamic programming. ILQR, which is one of the indirect optimal control method, is enough fast to be implemented in experiments. The rest of this section is a summary of the algorithm. Algorithm 1 shows the summary of the method.

After the discretization, vehicle kinematics can be represented as follows

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k + T_s R(\mathbf{x}_k)\mathbf{v}(\mathbf{u}_k) \\ &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \\ \mathbf{x} &= \begin{bmatrix} x & y & \theta \end{bmatrix}^T, \mathbf{u} = \begin{bmatrix} v_{xcom}, & v_{\theta com} \end{bmatrix}^T,\end{aligned}$$

where T_s means the time step, and $\mathbf{x}_k$, $\mathbf{u}_k$ are state and input at time k. Here, the velocity model $\mathbf{v}(\cdot)$ is used with the mean function of sparse GP learned-model, which is explained in section 2. Typical cost function can be derived as follows

$$J(\mathbf{x}, \mathbf{u}) = \sum_{k=0}^{N} l(\mathbf{x}_k, \mathbf{u}_k),$$

where N is the finite horizon length and $l(\mathbf{x}_k, \mathbf{u}_k)$ is the running cost. Then the optimal input sequence can be written as

$$\begin{aligned}\mathbf{u}_{0:N-1} &\equiv \arg\min J(\mathbf{x}, \mathbf{u}) \\ &\quad \text{subject to } \ \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k).\end{aligned}$$

To solve above optimal control problem, backward and forward process in time are repeated in the ILQR framework. First, in the backward process, the value function is defined as cost-to-go, written as

$$V_t(\mathbf{x}_t) = \min_{\mathbf{u}_{t:N-1}} \sum_{k=t}^{N} l(\mathbf{x}_k, \mathbf{u}_k).$$

The dynamic programming principle reduces the entire optimization problem into the sub-problem that computes the control input which minimizes the cost-to-go V:

$$V_t(\mathbf{x}_t) = \min_{\mathbf{u}_{t:N-1}} l(\mathbf{x}_t, \mathbf{u}_t) + V_{t+1}(\mathbf{x}).$$

To solve the subproblem, Q function is defined as the perturbation function of right hand side in above equation around the nominal trajectory $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ written as

$$\begin{aligned}Q_t(\delta\mathbf{x}_t, \delta\mathbf{u}_t) &= l(\bar{\mathbf{x}}_t + \delta\mathbf{x}_\mathbf{t}, \bar{\mathbf{u}}_t + \delta\mathbf{u}_t) - l(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t) \\ &\quad V\left(\mathbf{f}(\bar{\mathbf{x}}_t + \delta\mathbf{x}_t, \bar{\mathbf{u}}_t + \delta\mathbf{u}_t)\right) - V\left(\mathbf{f}(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)\right).\end{aligned}$$

To minimize the Q function, we first approximate Q function by using the Taylor's second-order expansion. The result is

$$\begin{aligned} Q_t(\delta\mathbf{x}_t, \delta\mathbf{u}_t) &\approx \frac{1}{2}\begin{bmatrix}\delta\mathbf{x}_t\\ \delta\mathbf{u}_t\end{bmatrix}^T l_{\mathbf{xu},\mathbf{xu},t}\begin{bmatrix}\delta\mathbf{x}_t\\ \delta\mathbf{u}_t\end{bmatrix} + \begin{bmatrix}\delta\mathbf{x}_t\\ \delta\mathbf{u}_t\end{bmatrix} l_{\mathbf{xu},t} + \\ &\quad \frac{1}{2}\begin{bmatrix}\delta\mathbf{x}_t\\ \delta\mathbf{u}_t\end{bmatrix}^T \mathbf{f}_{\mathbf{xu},t}^T V_{\mathbf{x},\mathbf{x},t+1}\mathbf{f}_{\mathbf{xu},t}\begin{bmatrix}\delta\mathbf{x}_t\\ \delta\mathbf{u}_t\end{bmatrix} + \begin{bmatrix}\delta\mathbf{x}_t\\ \delta\mathbf{u}_t\end{bmatrix}^T \mathbf{f}_{\mathbf{xu},t}^T V_{\mathbf{x},t+1}, \end{aligned}$$

where subscripts mean the differentiation. For instance, $l_{\mathbf{xu},\mathbf{xu},t}$ is the second derivative of l function with respect to $(\mathbf{x}_t, \mathbf{u}_t)$. The details of expansion coefficients are

$$\begin{aligned} Q_{\mathbf{x},t} &= l_{\mathbf{x},t} + \mathbf{f}_{\mathbf{x},t}^T V_{\mathbf{x},t+1} \\ Q_{\mathbf{u},t} &= l_{\mathbf{u},t} + \mathbf{f}_{\mathbf{u},t}^T V_{\mathbf{x},t+1} \\ Q_{\mathbf{xx},t} &= l_{\mathbf{xx},t} + \mathbf{f}_{\mathbf{x},t}^T V_{\mathbf{x},\mathbf{x},t+1}\mathbf{f}_{\mathbf{x},t} \\ Q_{\mathbf{uu},t} &= l_{\mathbf{uu},t} + \mathbf{f}_{\mathbf{u},t}^T V_{\mathbf{x},\mathbf{x},t+1}\mathbf{f}_{\mathbf{u},t} \\ Q_{\mathbf{ux},t} &= l_{\mathbf{ux},t} + \mathbf{f}_{\mathbf{u},t}^T V_{\mathbf{u},\mathbf{x},t+1}\mathbf{f}_{\mathbf{x},t}. \end{aligned}$$

Then, the optimal $\delta\mathbf{u}_t^*$, which minimizes approximated Q function, is calculated proceeding backwards in time. This policy is written as

$$\delta\mathbf{u}_t^* = -Q_{\mathbf{u},\mathbf{u},t}^{-1}(Q_{\mathbf{u},t} + Q_{\mathbf{u},\mathbf{x},t}\delta\mathbf{x}_t) = \mathbf{k}_t + \mathbf{K}_t\delta\mathbf{x}_t.$$

Here $\mathbf{k}_t$ is the feed-forward gain and $\mathbf{K}_t$ means the feedback gain along time.

Since the minimum of the Q function is the value function, we can get the value function by inserting optimal policy into the Q function. The result is

$$\begin{aligned} V_t(\delta\mathbf{x}_t) &\approx -\frac{1}{2}Q_{\mathbf{u},t}^T Q_{\mathbf{u},\mathbf{u},t}^{-1} Q_{\mathbf{u},t} + \frac{1}{2}\delta\mathbf{x}_t^T\left[Q_{\mathbf{x},\mathbf{x},t} - Q_{\mathbf{x},\mathbf{u},t}^T Q_{\mathbf{u},\mathbf{u},t}^{-1} Q_{\mathbf{u},\mathbf{x},t}\right]\delta\mathbf{x}_t + \\ &\quad \delta\mathbf{x}_t^T\left[Q_{\mathbf{x},t} - Q_{\mathbf{u},\mathbf{x},t}^T Q_{\mathbf{u},\mathbf{u},t}^{-1} Q_{\mathbf{u},t}\right]. \end{aligned}$$

The corresponding gradient and Hessians function of the value function are

$$\begin{aligned} V_{\mathbf{x},t} &= Q_{\mathbf{x},t} - Q_{\mathbf{u},\mathbf{x},t}^T Q_{\mathbf{u},\mathbf{u},t}^{-1} Q_{\mathbf{u},t} \\ V_{\mathbf{x},\mathbf{x},t} &= Q_{\mathbf{x},\mathbf{x},t} - Q_{\mathbf{x},\mathbf{u},t}^T Q_{\mathbf{u},\mathbf{u},t}^{-1} Q_{\mathbf{u},\mathbf{x},t}. \end{aligned}$$

Therefore, proceeding backward in time, we can get optimal policy.

After calculating the optimal perturbation of input $\delta\mathbf{u}_t^*$ the update of the trajectory $(\hat{\mathbf{x}}, \hat{\mathbf{u}})$ is proceeded through the forward pass,

$$\begin{aligned}\hat{\mathbf{x}}(0) &= \mathbf{x}(0)\\ \hat{\mathbf{u}}_{t+1} &= \bar{\mathbf{u}}_t + \delta\mathbf{u}_t^*\\ \hat{\mathbf{x}}_{t+1} &= \mathbf{f}(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t).\end{aligned}$$

The nominal trajectory $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ is replaced with the updated trajectory $(\hat{\mathbf{x}}, \hat{\mathbf{u}})$. Then, the entire process including backward pass and forward pass is repeated until the convergence of the nominal input and state.

To prohibit the divergence of control input, we apply the regularization for backward process and line-search for forward proceeding. During the backward process, we add a diagonal matrix $\mu\mathbf{I}$ to the local control hessian $Q_{\mathbf{u},\mathbf{u}}$ for numerical stability. Since $Q_{\mathbf{u},\mathbf{u}}$ often becomes a singular matrix, adding a diagonal term would make the non singular.

$$\tilde{Q}_{\mathbf{u},\mathbf{u}} = Q_{\mathbf{u},\mathbf{u}} + \mu\mathbf{I}.$$

Line-search is applied not to allow the divergence of trajectory. As we deal with nonlinear system based on a linearization, the new trajectory from forward pass becomes quiet different with the linearized system. As s result, the total cost of new trajectory easily increases, and even diverges. Therefore, we use line-search, which is written as,

$$\hat{\mathbf{u}}_{t+1} = \bar{\mathbf{u}}_t + \beta\mathbf{k}_t + \mathbf{K}_t\delta\mathbf{x}_t.$$

Here parameter β, which is $0 < \beta \leqq 1$, starts initially with 1. Whenever the result cost of updated trajectory increases than that of the trajectory at the previous iteration, we reduce parameter β properly.

Algorithm 1 Iterative linear quadratic regulator

```
1: pre-calculate the nominal state $\bar{\mathbf{x}}$ using $\bar{\mathbf{u}}$ and discrete model
2: for t = 0,1,$\cdots$,N do
3:     calculate derivative of model, $\mathbf{f}_{\mathbf{x},t}, \mathbf{f}_{\mathbf{u},t}$
4:     calculate gradients and Hessians of the cost, $l_{\mathbf{x},t}, l_{\mathbf{u},t}, l_{\mathbf{xx},t}, l_{\mathbf{xu},t}, l_{\mathbf{uu},t}$
5: for t = N,N-1,$\cdots$,0 do
6:     obtain expansion coefficients of $Q$ function, $Q_{\mathbf{x},t}$, $Q_{\mathbf{u},t}$, $Q_{\mathbf{xx},t}$, $Q_{\mathbf{uu},t}$, $Q_{\mathbf{ux},t}$
7:     get the optimal policy $\delta\mathbf{u}_t^*$
8:     update the gradients and Hessian of value function$V_{\mathbf{x},t}$, $V_{\mathbf{x},\mathbf{x},t}$
9: for t = 0,1,$\cdots$,N do
10:    update nominal $\bar{\mathbf{u}}_t$
11:    update nominal $\bar{\mathbf{x}}_t$ using forward dynamics
```

3.2 Cost formulation

The cost function is designed to make the vehicle stay on the track. To satisfy this purpose, we set the similar types of cost function similar to the one used for the aggressive driving in [19]. The running cost is wrtten as,

$$l(\mathbf{x}_k, \mathbf{u}_k) = k_v(v_{des} - v_{xcom})^2 + k_h h(x_k, y_k)^2 + k_w v_{\theta com}^2,$$

where k_v, k_h and k_w are weight factors for each cost. The running cost consists of three parts. The first part is to maintain the velocity of the vehicle to stay near the desired velocity. The second term is to make the robot stay on the track. The function h returns 0 if the vehicle is on the white line in the Fig. 3.1, -1 if off the track inside, 1 if off the track outside, and a value from -1 to 1 when the vehicle is within track. The last one is the cost term for turning command.

The benefit of this cost function is that it does not need to design the extra velocity scheduler. Usually, minimizing the lateral path error and regulating the speed are dealt as the separate tasks, for example in [12]. Particularly, when it comes to the skid-steer vehicle, it has no steering devices so that decoupling the speed control from the path tracking is difficult. Through our cost function, the vehicle can regulate its speed along the curvature of path by itself.

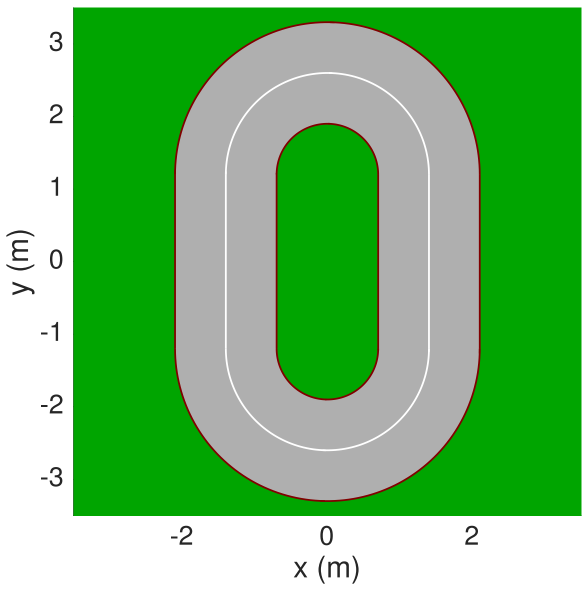

Figure 3.1: The track for path tracking experiments. The red-line means the inside and outside boundary. The white-line refers to the middle of the track.

3.3 Summary of the algorithm

The flow of the proposed method is shown in Algorithm 2. Given a initial driving data, the inducing input points are evenly picked over the input space of data. Using the inducing points, we find the posterior distribution of $\mathbf{V}_c$ using equation 2.1 based on the priori distribution $p(\mathbf{V}, \mathbf{V}_*, \mathbf{V}_c)$. Also, we can pre-calculate the inverse of the kernel matrix K_{cc}^{-1} because this matrix is consistent during driving once the inducing input points are fixed. After completing this step, it is ready for MPC to use the sparse GP model as the prediction model. After the vehicle starts to follow the path, the optimal policy is generated from MPC and applied to the real skid-steer mobile robot at each time step. The current velocity is simultaneously calculated to update the sparse GP model. The velocity is estimated as follows:

$$\hat{\mathbf{v}}_t \quad = \quad R^{-1}(\mathbf{x}_t)(\mathbf{x}_{t+1} - \mathbf{x}_{t+1})/T_s.$$

Given the new data $(\mathbf{u}_t, \hat{\mathbf{v}}_k)$, the posteriori distribution of the inducing points is updated through equation 2.1. Then entire process is repeated until the episode is finished. The 3.2 depicts the

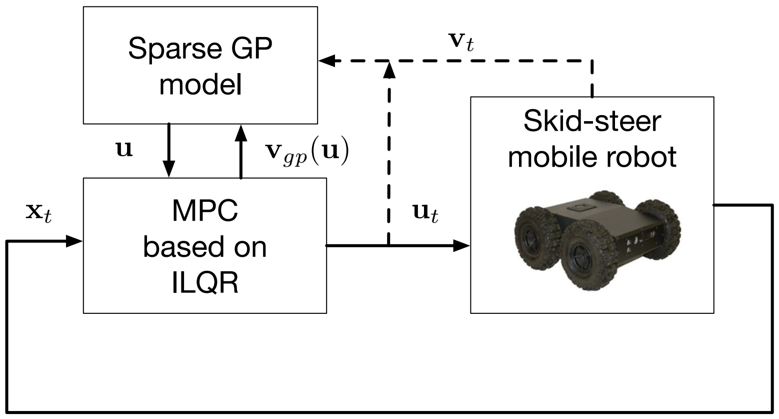

Figure 3.2: The block diagram of the control framework.

overall scheme of the proposed method.

Algorithm 2 Model predictive control with sparse GP

```
1: pick inducing input points $U_c$ evenly from the initial data
2: find the posterior distribution of $\mathbf{V}_c$ and calculate $K_{cc}^{-1}$
3: get the current position $\mathbf{x}_0$
4: for t = 0,1,$\cdots$,T do
5:     calculate the optimal input $\mathbf{u}_t$ from ILQR
6:     apply $\mathbf{u}_t$ to the skid-steer robot
7:     obtain the next state $\mathbf{x}_{t+1}$ from sensors
8:     estimate the current velocity $\mathbf{v}_t$
9:     update the posteriori distribution $\mathbf{V}_c$ using the current velocity $\mathbf{v}_t$ and $\mathbf{u}_t$
```

4 Experiments

The proposed methods are validated with the skid-steer vehicle through experiments. We present the experimental setup and results of indoor experiments and outdoor experiments. Then, the discussion of results follows.

4.1 Experimental setup

The skid-steer vehicle used in experiments is a Dr. Robot Jaguar 4x4 mobile robot platform in Fig. 1.1. The weight of the vehicle is 20.5kg, and the width and length are 573mm and 615mm. The main PC, which is MacBook Pro 2015 with 2.9GHz Intel Core i5, calculates the control input and transmits the command to the on-board ARM cortex through WiFi. A motion capture system (VICON) measures the position of the vehicle for the indoor experiment. To measure the position of the vehicle in outdoor environment, pixhawk module is used. This module has its own algorithm that conduct GPS and IMU sensor fusion in order to increase accuracy of position estimation. We utilize this value from VICON and pixhawk module as an position data, and estimate the current velocity by equation 3.1. The control system architecture is implemented in ROS (Robot Operating System) with C++.

To validate the performance of this research, the experiments are conducted on three kinds of different path in the indoor environment: circular, triangular and rectangular paths. In each path, the results of the proposed methods are compared with the results of the cases which use IDD model and GKS model as the prediction model in MPC. The parameters in GKS model are selected on-line by minimizing the below objective function through a linear least-squares method,

$$w(\alpha) = \frac{1}{2}\sum_{k=0}^{T} \| \mathbf{v}_{meas,k} - \mathbf{v}_{gks,k}(\alpha) \|^2,$$

where $\mathbf{v}_{meas}$ is the set of the measured velocity up to the current time T.

When it comes to the outdoor experiments, we execute an experiment with circular path. We also compare the result of the developed method with that of IDD model with MPC. In this case, we analyze the cost result for every lap in each model and lap time.

The driving data during 40 seconds is used as the initial data for a circular path and 60 seconds for the triangular and rectangular paths in indoor environment. Note that the initial driving data used in GKS model and the sparse GP model is same in each experiment. On the other hand, the 120 seconds driving data is used for the circular path in outdoor environment. In the sparse GP model, we evenly select 225 inducing input points over the input of initial data in indoor experiment and 400 points in outdoor experiment. In MPC, the finite-horizon is set to 50 steps. The control frequency is approximately 10Hz, and the frequency of the on-line updating for learning velocity model is also 10Hz. The time for each experiment is equally 60 seconds. The target forward command of the cost function in experiments is described in each result of input command and velocity.

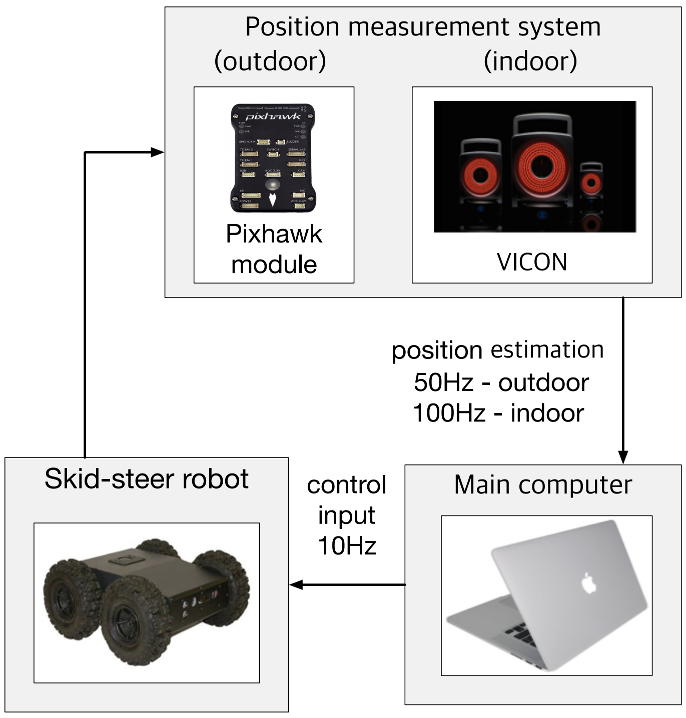

Figure 4.1: Experiment setting for indoor and outdoor environment.

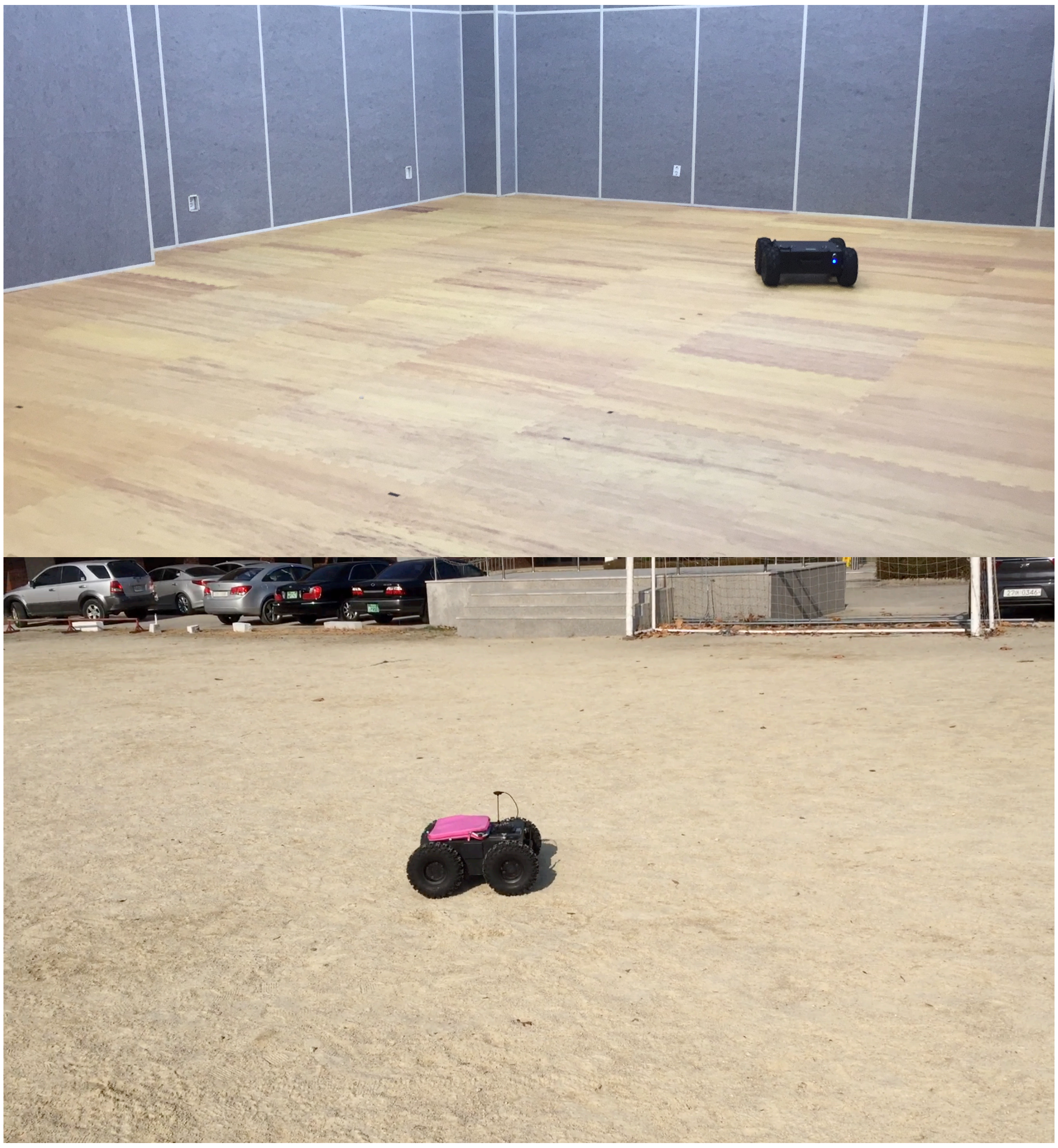

Figure 4.2: The terrain environment of experiments: indoor (up) and outdoor (bottom)

4.2 Indoor experimental results

Table 4.1: Comparison of the cost results for three paths.

	Circular	Triangular	Rectangular
SparseGP	653.41	1014.02	1011.58
General slip	699.65	1016.54	1379.10
IDD	941.83	1588.11	1525.37

Fig. 4.3, Fig. 4.4 and Fig. 4.5 show the results along the three paths. For the purpose of numerical comparison, we denote the cost results in Table 4.1. The written cost is calculated by the cost function used in MPC with the entire state and input results. The sparse GP model records the lowest cost results for three path. In the circular path figure, the path results of GKS model is more slightly off the course initially than the sparse GP model. After the first lap, both the GKS model and the sparse GP model converge to the similar path results. Three models show the different results in the triangular paths. Although the path of the sparse GP initially deviates from the desired path in the upper portion of the figure, it is gradually close to the given path, which is closer the desired path than those of the GKS and IDD in the upper part of the path. In the rectangular path, the sparse GP model shows the better result than other methods. First, the path error of the sparse GP model becomes lower than those of GKS and IDD models especially at the beginning. Plus, MPC in the sparse GP model generates the larger average forward command, 0.6021 m/s than that of the GKS model, 0.5061 m/s, which is described in Fig. 4.8. We can also check that the total distance the vehicle moved using the sparse GP model is much longer than the distance in other models by comparing the final positions in Fig. 4.5. As a result, the cost results for the sparse GP model in Table 4.1 are much lower than the cases which use other models.

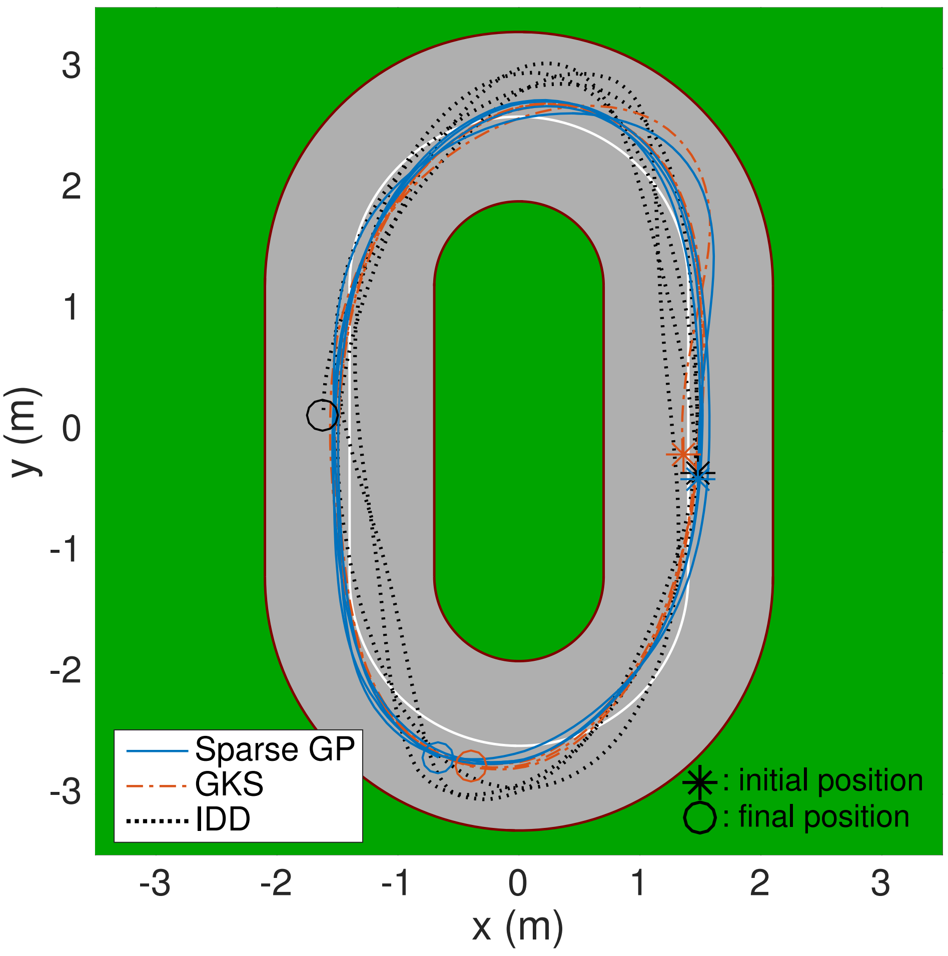

Figure 4.3: Experiment results for indoor environment along the circular paths.

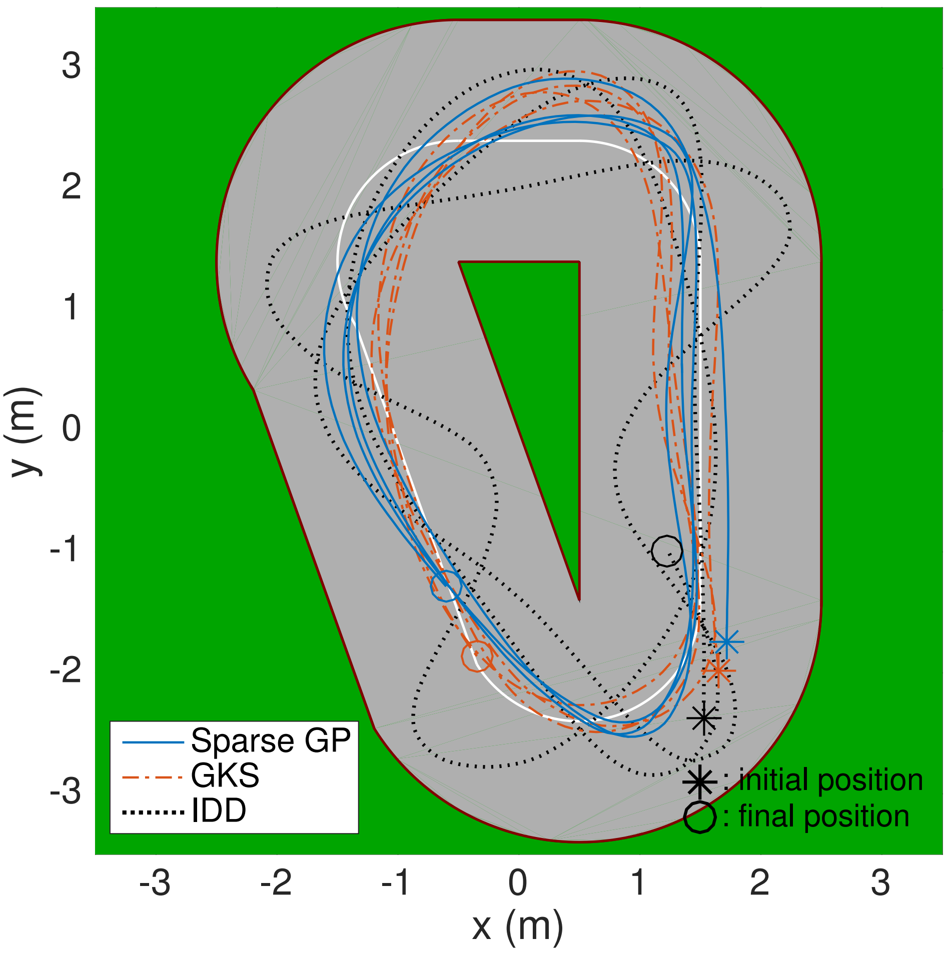

Figure 4.4: Experiment results for indoor environment along the triangular paths.

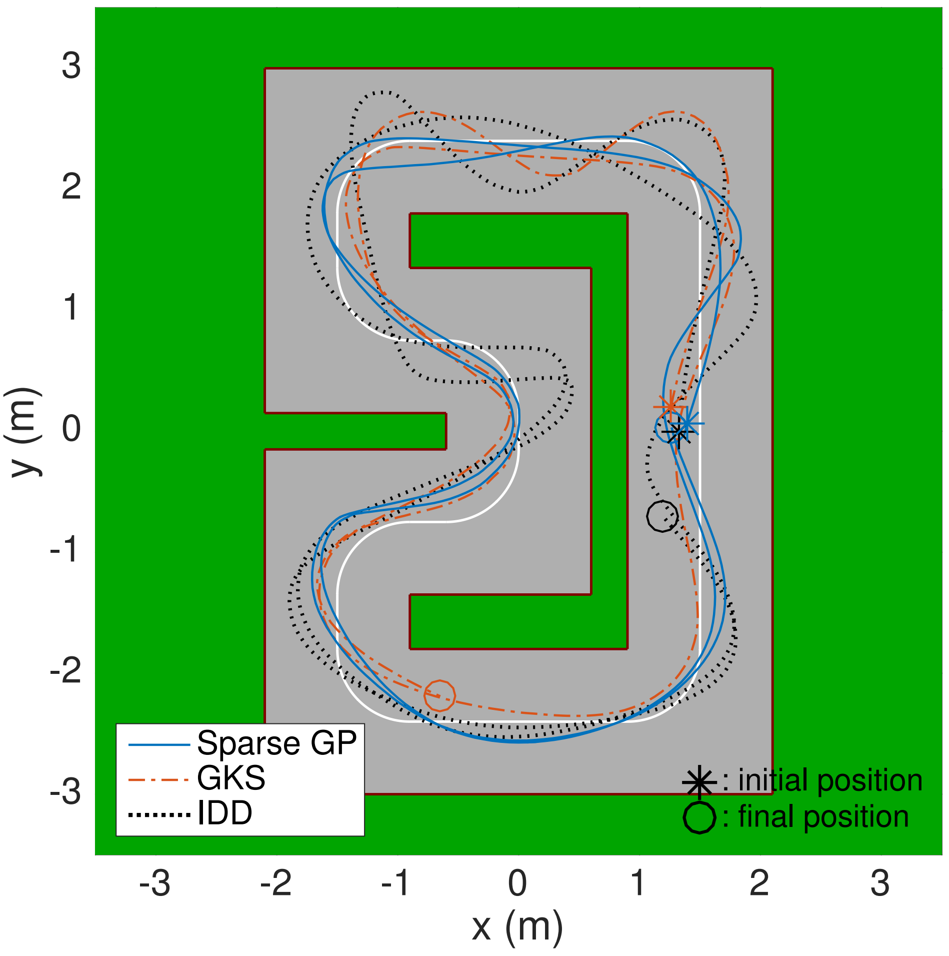

Figure 4.5: Experiment results for indoor environment along the rectangular paths.

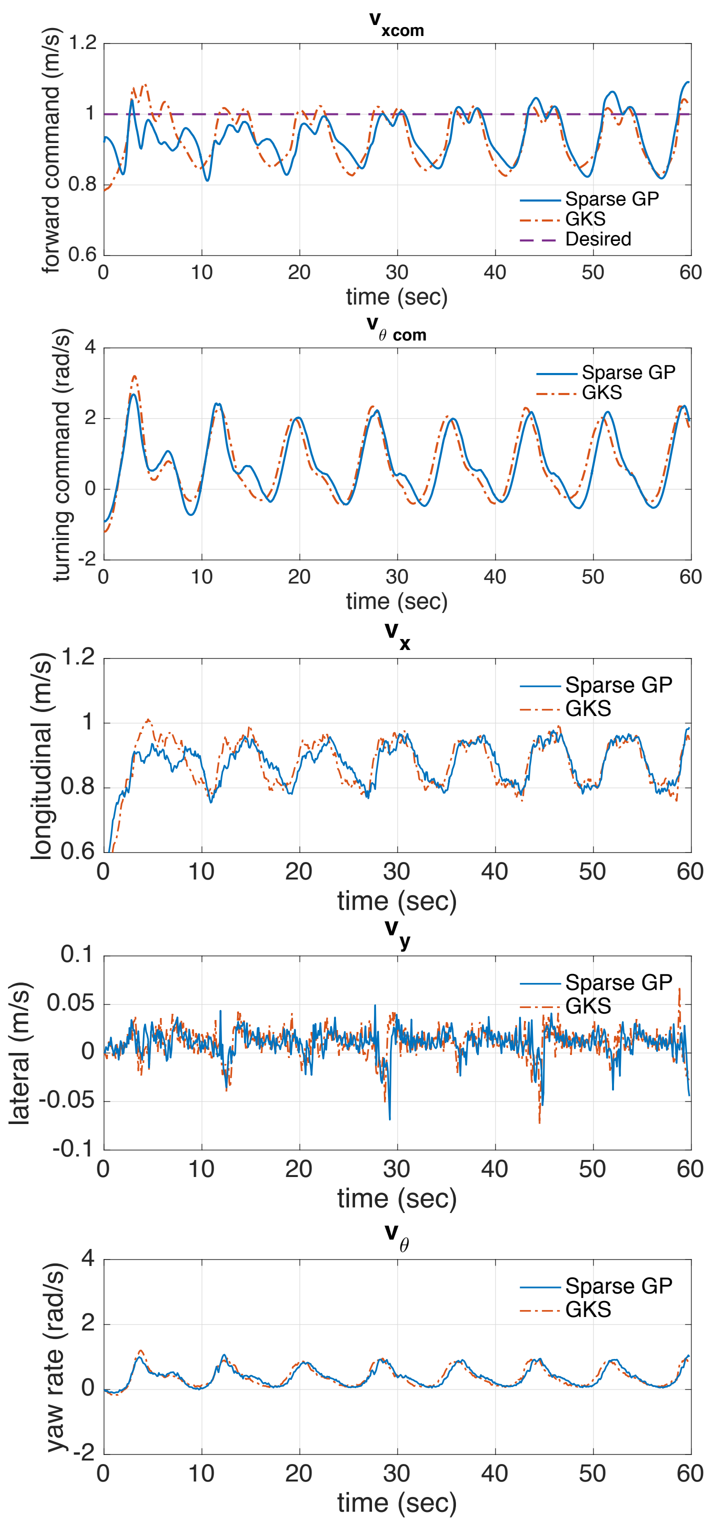

Figure 4.6: Input command and estimated velocity results of circular path for indoor environment. The desired forward command is set to 1.0 m/s

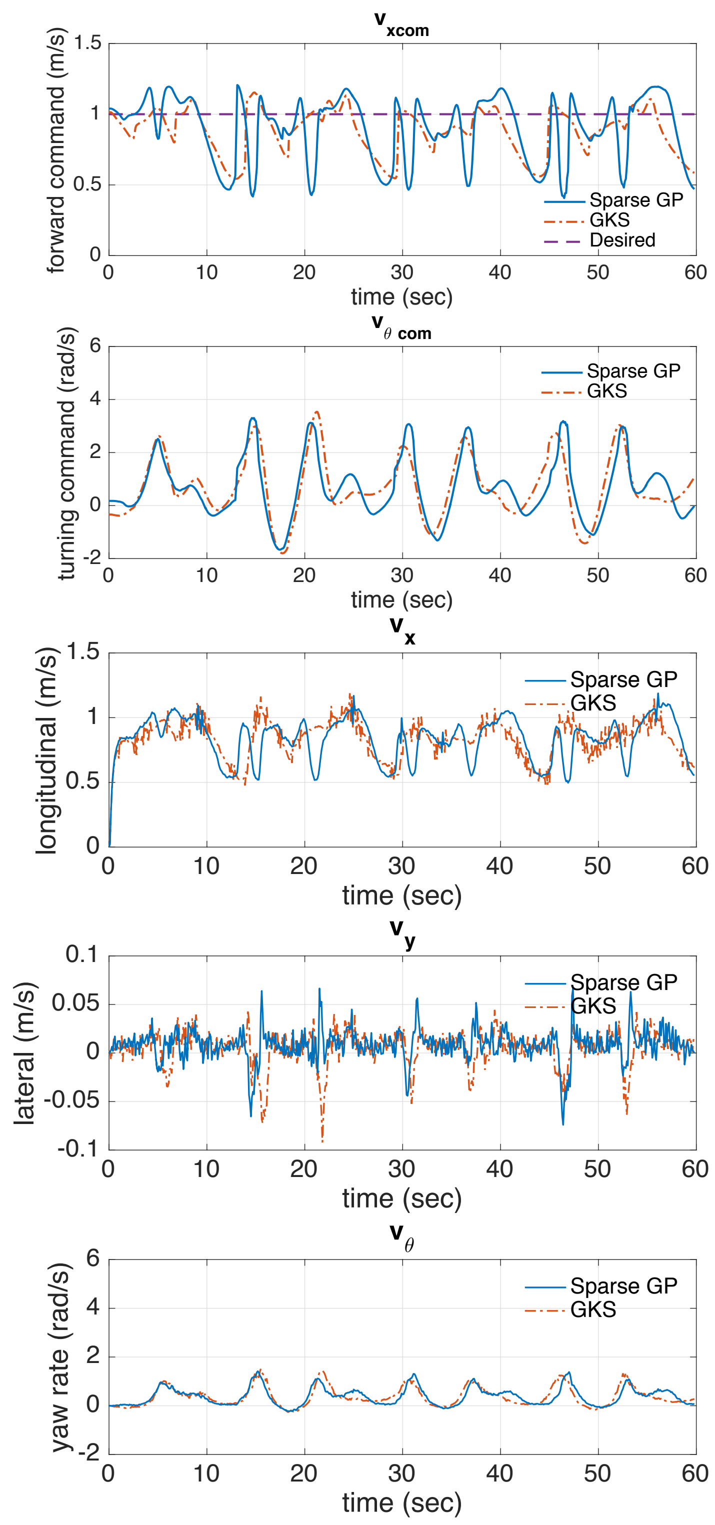

Figure 4.7: Input command and estimated velocity results of triangular path for indoor environment. The desired forward command is set to 1.0 m/s

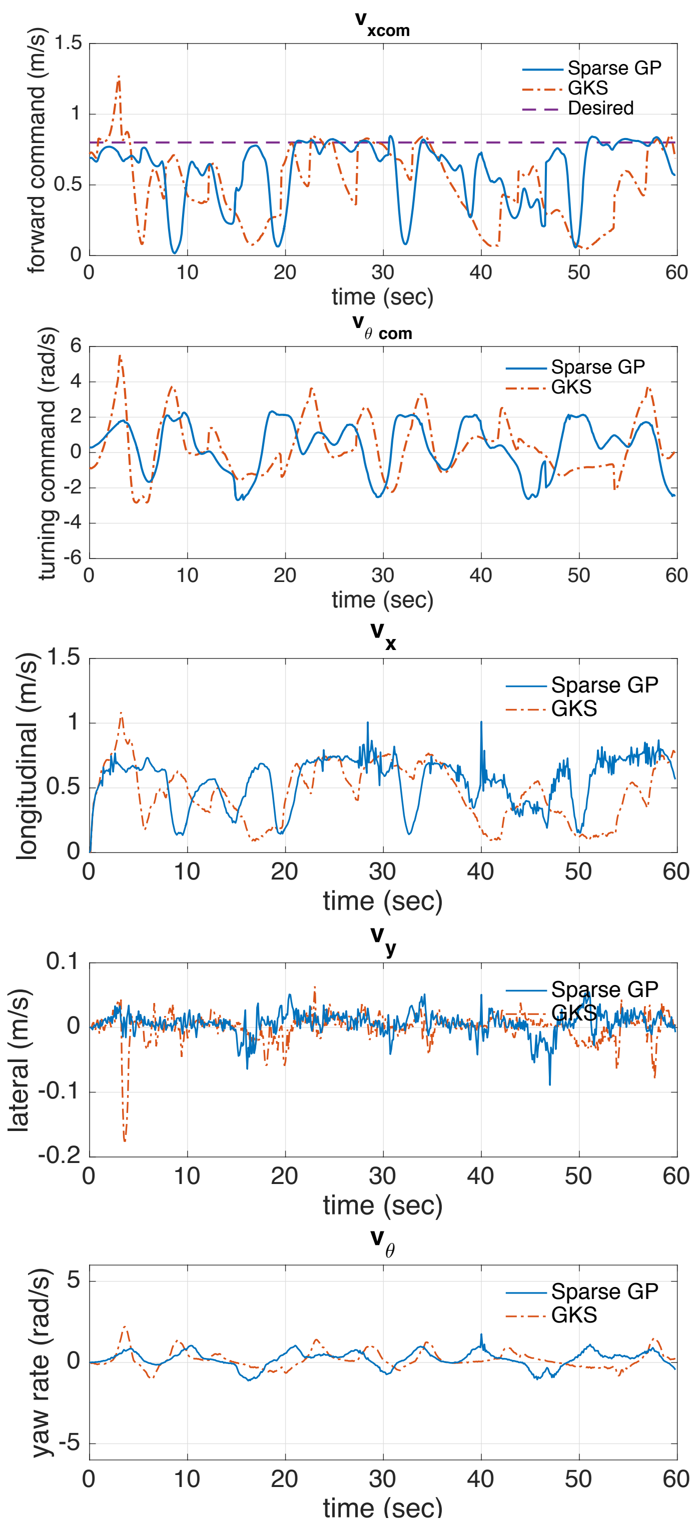

Figure 4.8: Input command and estimated velocity results of rectangular path for indoor environment. The desired forward command is set to 0.8 m/s which is lower than other paths because this path is much complex than others.

4.3 Outdoor experimental results

Fig. 4.9 shows the result along the circular paths in outdoor environments. For the purpose of numerical comparison, we also denote the cost results in Table 4.2. The cost results are written for each lap and the model used in each experiment, along with the lap time. This lap time is measured as the time until vehicle returns to its initial position after driving. The path result of IDD model is more slightly off the course initially at the upper-right region. After driving the upper area of the circuit, the path result of IDD model still deviates to the given path at the bottom area. The path result of GP model in first lap is more far to the given path than that of IDD model. After driving upper region of the path, however, the vehicle starts to converge to the desired path. Then, the path result of second lap maintains lower error than before. As a result, the cost result and lap time of GP case are lower than those of IDD case in the first lap. Furthermore, the vehicle has better performance in the second lap than the first lap in GP case by comparing cost result and lap time.

Table 4.2: The cost and lap time results for outdoor experiment at each lap.

	Cost		Time (s)	
	IDD	**SparseGP**	IDD	**SparseGP**
lap no. 1	1936.10	1816.53	36.7	34.5
lap no. 2	-	1764.63	-	33.3

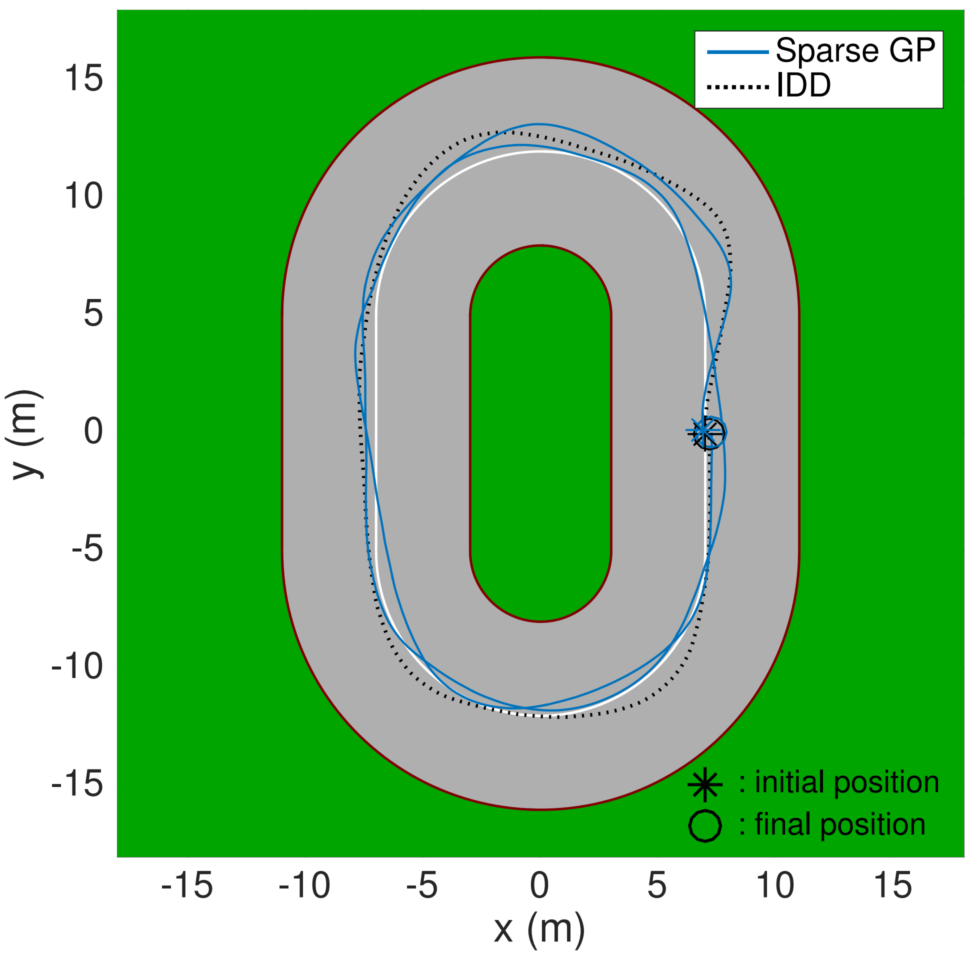

Figure 4.9: Experiment results for outdoor environment along the circular paths: laps no. 2 for sparse GP case and lap no. 1 for IDD case.

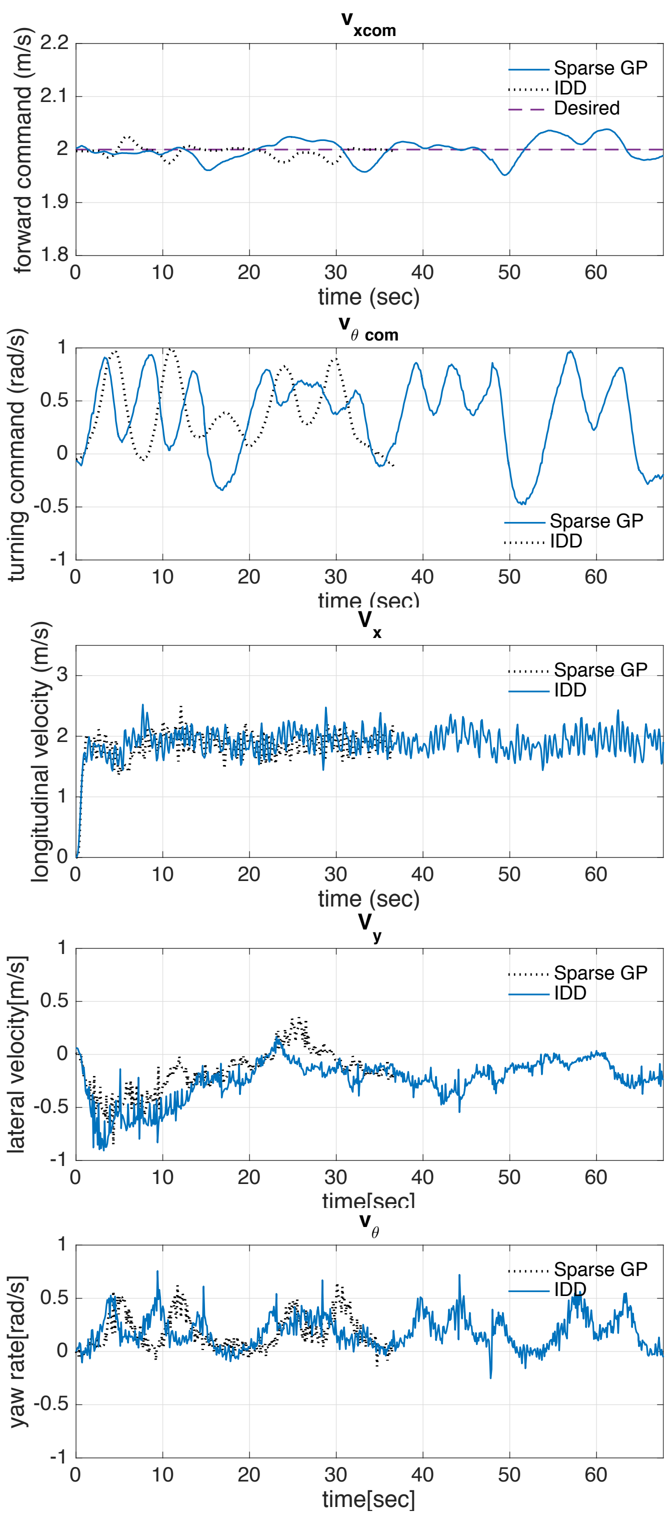

Figure 4.10: Input command and estimated velocity results of circular path for outdoor environment. The desired forward command is set to 2.0 m/s which is higher than that of indoor experiements.

5 Conclusion

In this paper, the on-line sparse GP with MPC was applied to the path tracking control of a skid-steer vehicle. First, we discussed the kinematics of skid-steer vehicle in the Chapter 2.1. The target function of GP was set to the velocity model in the kinematics. Then, the on-line sparse GP was used for learning the velocity model without explicit knowledge of the vehicle in Chapters 2.2 and 2.3. In order to apply MPC, the optimal control policy was generated by ILQR based on the learned model in Chapter 3.1. For the improved driving performance, the cost function was formulated to combine the speed regulation and the path following in Chapter 3.2. Experiments with skid-steer mobile robot platform were conducted in indoor and outdoor environments. The results in Chapters 4.2 and 4.3 showed that this approach can attain better performance than the methods based on the conventional models.

In the rest of this chapter, we will discuss the future directions and applications of this method.

5.1 Challenges and future works

One possible future work is to utilize the covariance function of the learned model in order to improve the exploration during the model learning. Although we obtained the learned model from data, we still did not make use of the covariance function of the learned model. Drawbacks of this approach are that the trajectories resulted from optimal controller remain within the region of state space represented by training data, and that ILQR often fails to generate policy when the vehicle enters the state space not sufficiently depicted by training data. This is because ILQR assumes that the current prediction model is perfectly accurate. One way to solve this problem is to augment the prediction variance in the cost function. By augmenting the variance function in the cost function, the policy which makes robot search the high-variance region will be generated. As a result, it is expected that the efficiency of model learning will improve.

Another future work is to apply on-line hyperparameter optimization. We currently fixed the hyperparameters for computational efficiency. The drawback of this approach is that the hyperparameters can deviate from their optimal values as the on-line model update progresses. In this case, the controller tends to lose its performance. To maintain the performance, it seems that the on-line hyperparameter adaptation is needed.

As an application, the autonomous navigation on a rough terrain is well suited with this method, such as off-road vehicles or rovers on a lunar-like terrain which involve several challenges because of unevenness, slipping, subsidence and so on. Since the developed method can cope with the unknown terrain, this method would become a proper controller for those challenges of outdoor navigation or planetary exploration.

References

[1] D. Lhomme-Desages, C. Grand, and J. Guinot, “Trajectory control of a four-wheel skid-steering vehicle over soft terrain using a physical interaction model.” in *ICRA*, 2007, pp. 1164–1169.

[2] E. Lucet, C. Grand, D. Sallé, and P. Bidaud, “Dynamic control of the 6wd skid-steering mobile robot roburoc6 using sliding mode technique,” in *IEEE Int. Conf. on Robotics and Automation*, 2009.

[3] J. Kang, W. Kim, J. Lee, and K. Yi, “Skid steering-based control of a robotic vehicle with six in-wheel drives,” *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 224, no. 11, pp. 1369–1391, 2010.

[4] H. B. Pacejka and E. Bakker, “The magic formula tyre model,” *Vehicle system dynamics*, vol. 21, no. S1, pp. 1–18, 1992.

[5] L. Caracciolo, A. De Luca, and S. Iannitti, “Trajectory tracking control of a four-wheel differentially driven mobile robot,” in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 4. IEEE, 1999, pp. 2632–2638.

[6] H. Wang, J. Zhang, J. Yi, D. Song, S. Jayasuriya, and J. Liu, “Modeling and motion stability analysis of skid-steered mobile robots,” in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on.* IEEE, 2009, pp. 4112–4117.

[7] J. Pentzer, S. Brennan, and K. Reichard, “Model-based prediction of skid-steer robot kinematics using online estimation of track instantaneous centers of rotation,” *Journal of Field Robotics*, vol. 31, no. 3, pp. 455–476, 2014.

[8] A. Kelly, “Linearized error propagation in odometry,” *The International Journal of Robotics Research*, vol. 23, no. 2, pp. 179–218, 2004.

[9] F. Rogers-Marcovitz, N. Seegmiller, and A. Kelly, “Continuous vehicle slip model identification on changing terrains,” in *RSS 2012 Workshop on Long-term Operation of Autonomous Robotic Systems in Changing Environments*, 2012.

[10] V. Rajagopalan, A. Kelly *et al.*, “Slip-aware model predictive optimal control for path following,” in *Robotics and Automation (ICRA), 2016 IEEE International Conference on.* IEEE, 2016, pp. 4585–4590.

[11] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, “Conservative to confident: treating uncertainty robustly within learning-based control,” in *2015 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2015, pp. 421–427.

[12] C. J. Ostafew, A. P. Schoellig, T. D. Barfoot, and J. Collier, “Learning-based nonlinear model predictive control to improve vision-based mobile robot path tracking,” *Journal of Field Robotics*, vol. 33, no. 1, pp. 133–152, 2016.

[13] J. Quiñonero-Candela and C. E. Rasmussen, “A unifying view of sparse approximate gaussian process regression,” *Journal of Machine Learning Research*, vol. 6, no. Dec, pp. 1939–1959, 2005.

[14] H. Bijl, J.-W. van Wingerden, T. B. Schön, and M. Verhaegen, “Online sparse gaussian process regression using fitc and pitc approximations,” *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 703–708, 2015.

[15] H. Bijl, T. B. Schön, J.-W. van Wingerden, and M. Verhaegen, “Online sparse gaussian process training with input noise,” *arXiv preprint arXiv:1601.08068*, 2016.

[16] J. R. Fink and E. A. Stump, “Experimental analysis of models for trajectory generation on tracked vehicles,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems.* IEEE, 2014, pp. 1970–1977.

[17] C. E. Rasmussen, “Gaussian processes for machine learning,” 2006.

[18] Y. Tassa, N. Mansard, and E. Todorov, “Control-limited differential dynamic programming,” in *2014 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2014, pp. 1168–1175.

[19] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, “Aggressive driving with model predictive path integral control,” in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1433–1440.

국 문 초 록

본 논문은 스키드 조향 로봇의 경로 제어에 관해 다룬다. 스키드 조향이란 별도의 조향 장치 없이 양측 바퀴의 속도 차이로 주행하는 메커니즘을 말한다. 이러한 조향 기법은 일반적으로 지면으로부터 큰 종력을 얻을 수 있어 거친 지면에서의 주행에 유리하다. 하지만 양측 바퀴 제어만을 통한 선회 운동의 경우에는 지면으로 부터 받는 영향이 조향이 있는 경우보다 크게 작용하여 로봇 경로 제어를 어렵게 만든다. 또한 로봇의 주된 동력원인 휠의 경우 바퀴 자체의 특성뿐만 아니라 주행하고 있는 지면의 성질, 바퀴 제어 성능에 영향을 받아 제어 설계를 어렵게 만든다. 이러한 특징을 고려하기 위해, 본 연구에서는 온라인 모델 학습에 기반한 모델 예측 제어 기법을 스키드 조향 차량의 경로 제어에 적용하였다. 로봇의 입력과 실제 속도와의 관계를 나타내는 속도 모델을 실시간으로 학습하고, 학습된 모델을 기반으로 하는 예측 제어 기법을 구현하였다. 모델 학습은 온라인 sparse 가우시안 프로세스를 이용하여 기존의 가우시안 프로세스의 단점 중 하나인 계산 복잡성을 낮추고자 하였다. 이후 로봇의 주행을 위해 학습된 모델을 최적 제어 기법의 일종인 모델 예측 제어 기법과 결합하였다. 개발한 기법을 실내외 실험을 통해 검증하였고, 기존 연구 기법의 일종인 파라미터 적응에 기반한 모델을 이용한 경우와의 비교를 통해 우수한 주행 성능을 확인하였다.

주요어 : 경로 제어, 스키드 조향, 모델 예측 제어, 가우시안 프로세스.

학번 : 2015-20766