Learning-Based Model Predictive Control for Improved Mobile Robot Path Following using Gaussian Processes and Feedback Linearization

Jie Wang, Michael T. H. Fader, and Joshua A. Marshall, *Senior Member, IEEE.*

***Abstract*—This paper proposes a high-performance path following algorithm that combines Gaussian Processes (GP) based learning with Feedback Linearized Model Predictive Control (FBLMPC) for ground mobile robots operating in off-road terrains, referred to as GP-FBLMPC. The algorithm uses a baseline kinematic model and learns unmodeled dynamics as GP models by using observation data collected during field experiments. We demonstrate the effectiveness of the proposed algorithm via extensive outdoor experiments using a Clearpath Husky A200 mobile robot. The test results indicate that the proposed GP-FBLMPC algorithm's performance is comparable to GP learning-based nonlinear MPC (GP-NMPC) with respect to the path following errors. However, GP-FBLMPC is computationally more efficient than the GP-NMPC because it does not conduct iterative optimization and requires fewer GP models to make predictions over the MPC prediction horizon loop at every time step. Field tests show the algorithm requires very little training data to perform GP modeling before it can be used to reduce path following errors for new, more complex paths in the same terrain (see video at https://youtu.be/BhD-sUd2qAI).**

***Index Terms*—Mobile robots, Path following, Model predictive control, Feedback linearization, Gaussian processes.**

I. Introduction

Path following is a fundamental functionality for autonomous vehicles and mobile robots, where the vehicle is required to track a prescribed path irrespective of time. High-accuracy path following for wheeled robots operating in off-road terrains is challenging because it is difficult, sometimes even impossible, to develop accurate mathematical models of robot-terrain interaction from first principles [1]. More recently, learning-based or data-driven modeling for controllers has become a popular field of study [2].

In this work, we propose a Gaussian Process (GP) based Feedback Linearized Model Predictive Control (GP-FBLMPC) algorithm for path following control of wheeled mobile robots operating in challenging terrains. In preliminary work [3] we studied a path-following algorithm that combines Model Predictive Control (MPC) with Feedback Linearization (FBL) for wheeled-vehicle path-following applications, referred to as FBLMPC. The FBLMPC algorithm eliminates the need for iterative nonlinear optimization, thus is computationally efficient yet easy-to-implement. In this paper, we further improve path following performance by integrating GP models with a fixed kinematic vehicle model. The GP models are used to learn the discrepancies between a prior kinematic model and the true system model (the observed real-world system behaviors) as disturbance models. The GP models are functions of the robot state and input variables from the observed real-world experiences to learn system disturbances that include the unmodeled dynamics of the robot, wheel-terrain interactions, and other disturbances. While learning-based control for mobile robots has been studied extensively, learning for feedback linearization-based controllers is limited [4]. To our knowledge, the GP-FBLMPC algorithm is the first application of GP learning-based feedback linearized MPC for wheeled mobile robot path following, and we show that it is generalizable for the purposes of reducing path following errors on path different from the ones use for training.

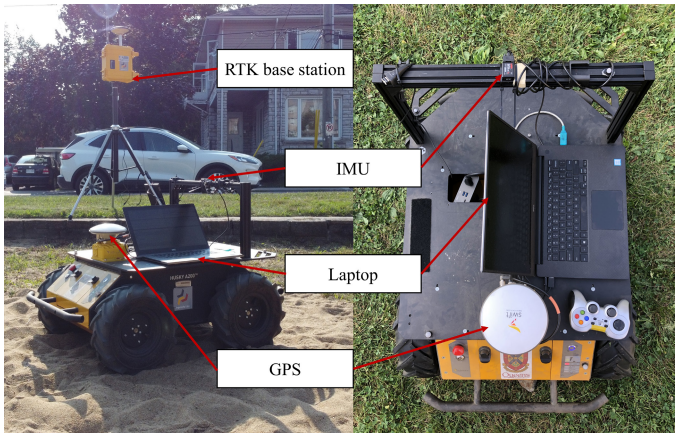

Fig. 1. Configuration of the Clearpath Husky A200 mobile robot used in this work for field experiments in sand and grass terrains.

The main contributions of this work are as follows: (i) A high-performance path-following algorithm (GP-FBLMPC) that combines a fixed kinematic vehicle model with GP learning-based disturbance models as the system process model; (ii) extensive outdoor field experiments using a Husky A200 mobile robot (see Fig. 1) to validate the effectiveness of the GP-FBLMPC algorithm; and (iii) comparison to a GP learning-based nonlinear MPC algorithm to show that the GP-

Jie Wang performed this work as a member of the Ingenuity Labs Research Institute at Queen's University, Kingston, Ontario, Canada. Email: jwangjie@outlook.com

Michael T. H. Fader performed this work as a member of the Ingenuity Labs Research Institute at the Queen's University. He is now with MDA Robotics & Space Operations, Brampton, Ontario, Canada. Email: michaelfader@outlook.com

Joshua A. Marshall is with the Ingenuity Labs Research Institute and the Department of Electrical & Computer Engineering, Queen's University, Kingston, Ontario, Canada. Email: joshua.marshall@queensu.ca

Manuscript revised December 2, 2021.

FBLMPC algorithm is also more computationally efficient and generalizable to the problem of reducing path following errors across a variety of paths.

In this paper, Section II reviews the recent research in this field. Section III describes the mathematical concepts behind FBLMPC, GP models, and GP learning-based MPC. Section IV explains the implementation details of the GP-FBLMPC algorithm. Section V describes experimental results of the filed tests that validate the effectiveness of the GP-FBLMPC algorithm. Section VI provides concluding remarks.

II. Related Work

There are already several methods that apply learning-based MPC to path following problems for nonlinear mobile robots; see [5] for a review. The current work is most inspired by [1], who proposed a GP learning-based Nonlinear MPC (NMPC) algorithm for a path-repeating mobile robot operating in challenging terrains. The algorithm, referred to herein as GP-NMPC, uses a fixed kinematic robot model and disturbance GP models in the prediction horizon loop to reduce path-tracking errors. The GP-NMPC uses previous experience to estimate disturbance GP models. In this work, we use a similar GP learning-based MPC framework but with one essential difference that we integrate GP models with FBLMPC.

Existing research on learning-based control combined with feedback linearization is limited as it relates to mobile robots [4]. The most popular learning strategies applied to feedback linearizing control is to learn the forward model of the nonlinear terms of the robot system [4], [6]. Because feedback linearization requires an exact cancellation of the system's nonlinearities, learning forward models contributes to this need directly. Other methods learn transformed nonlinear terms, nonlinear mismatch [7], which is defined as the model error between the nominal feedback linearization controller and nonlinear term. In [8], a forward model of the nonlinear mismatch is learned by GP models to generate a bound indicating how well the nominal feedback linearization controllers can linearize the system. The bound is used in a linear outer-loop controller. The continuing work in [7] further learns an inverse model of the nonlinear mismatch by GP models to cancel the nonlinear mismatch, and is thus able to improve the feedback linearization. Different from these methods, this paper uses a nominal kinematic model as the process model for our FBLMPC and the unmodeled system dynamics are learned by GP models to update the process model in the prediction horizon loop of the GP-FBLMPC algorithm.

III. Mathematical Formulation

We investigate the application of GP-FBLMPC on a Clearpath Husky A200 skid-steer mobile robot, as shown in Fig. 1. This section presents the necessary mathematical formulation of the GP-FBLMPC setup for this case.

A. Feedback Linearized Model Predictive Control (FBLMPC)

The kinematics of a skid-steer robot can be approximately modeled by a simple "unicycle" vehicle model (see Fig. 2). The configuration of the model can be described by the coordinates $\mathbf{q} = (x, y, \theta) \in \mathbb{R}^2 \times \mathbb{S}^1$, where x and y are the position of the geometric center and θ is the orientation of the vehicle with respect to the global frame $[X, Y]$, respectively. The continuous-time nonholonomic kinematic model is

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} v \\ \omega \end{bmatrix}}_{\mathbf{u}}, \tag{1}$$

where the input $\mathbf{u} = (v, \omega)$ comprises the commanded forward velocity v [m/s] and yaw rate ω [rad/s].

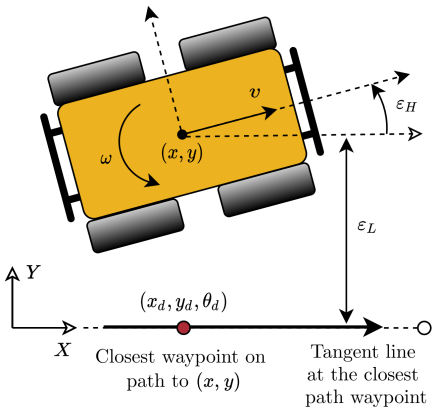

Fig. 2. Diagram of path following errors defined for Husky robot with respect to a desired path.

Fig. 2 shows the path following errors (lateral ε_L and heading ε_H) as the vehicle travels along a desired path. The desired path is a sequence of evenly and closely spaced discrete waypoints, where each waypoint specifies a desired vehicle pose (x_d, y_d, θ_d). The reference point (x, y) of the robot is at the geometric center of the chassis. We assume the X-axis of the global reference frame is continuously aligned with an instantaneous tangent at the closest waypoint to the vehicle so the path following errors are $\varepsilon_L \coloneqq y$ and $\varepsilon_H \coloneqq \theta$. Given (1), the instantaneous rate of change of the path following errors are $\dot{\varepsilon}_L = \dot{y} = v \sin\varepsilon_H$ and $\dot{\varepsilon}_H = \dot{\theta} = \omega$. Similar to [9], a change of coordinates $z_1 \coloneqq \varepsilon_L = y$ and $z_2 \coloneqq \dot{\varepsilon}_L = \dot{y}$ is applied. We refer to these new coordinates $\mathbf{z} = [z_1, z_2]^\top$ as the feedback linearized (FBL) states and

$$\dot{\mathbf{z}} = \begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u = \mathbf{A}\mathbf{z} + \mathbf{B}u, \tag{2}$$

is the linearized system model where u is a new control input. With a constant velocity v, $u = \dot{z}_2 = \omega v \cos\varepsilon_H$. Inverting to solve for the corresponding steering rate yields $\omega = u/(v\cos\varepsilon_H)$. Clearly, this nonlinear transformation from u to ω works only when $v \neq 0$ and $\varepsilon_H \in (-\pi/2, \pi/2)$.

The FBLMPC formulation generates a sequence of control inputs $\mathbf{u}_k$ at discrete time indices k such that $\mathbf{u}_k \coloneqq \Delta\mathbf{u}_k + \mathbf{u}_{k-1} = (u_k, u_{k+1}, ..., u_{k+p-1})$ over a finite time horizon of length p to minimize predicted path following errors estimated by a model. Consider a discretized version of (2),

$$\mathbf{z}_{k+1} = \mathbf{F}\mathbf{z}_k + \mathbf{G}u_k = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} z_{1,k} \\ z_{2,k} \end{bmatrix} + \begin{bmatrix} \frac{T^2}{2} \\ T \end{bmatrix} u_k, \tag{3}$$

for the sample time $0 < T \ll 1$ and $\mathbf{z}_k \equiv \mathbf{z}(t = kT)$, $k = 0, 1, \ldots, m$ with m represents the number of time steps to finish a trial. Hence, the states $\mathbf{z}$ over the prediction horizon are $\mathbf{y}_{k+1} = \Delta\mathbf{y}_{k+1} + \mathbf{y}_k = (\mathbf{z}_{k+1}, \mathbf{z}_{k+2}, ..., \mathbf{z}_{k+p})$, where the discrete updates in control input and FBL states are $u_k \coloneqq \Delta u_k + u_{k-1}$ and $\mathbf{z}_k \coloneqq \Delta\mathbf{z}_k + \mathbf{z}_{k-1}$ respectively. Starting at time index k, (3) can be used to describe how the states $\mathbf{z}$ change over the prediction horizon as

$$\begin{aligned}
\mathbf{z}_{k+1} &= \mathbf{F}\mathbf{z}_k + \mathbf{G}u_k \\
\Delta\mathbf{z}_{k+1} + \mathbf{z}_k &= \mathbf{F}(\Delta\mathbf{z}_k + \mathbf{z}_{k-1}) + \mathbf{G}(\Delta u_k + u_{k-1}) \\
\Delta\mathbf{z}_{k+1} &= \mathbf{F}\Delta\mathbf{z}_k + \mathbf{G}\Delta u_k \\
&\vdots \\
\Delta\mathbf{z}_{k+p} &= \mathbf{F}^p\Delta\mathbf{z}_k + \mathbf{F}^{p-1}\mathbf{G}\Delta u_k + \cdots + \mathbf{G}\Delta u_{k+p-1}\,,
\end{aligned}$$

rewriting in a compact matrix form as

$$\Delta\mathbf{y}_{k+1} = \mathbf{L}\Delta\mathbf{z}_k + \mathbf{M}\Delta\mathbf{u}_k. \tag{4}$$

The MPC approach seeks to find a sequence of control inputs over the prediction horizon that minimizes a cost function. In this work, we do not apply additional state or input constraints, and define a (standard) cost function for the control objective as $J(\mathbf{u}_k) = \mathbf{y}_{k+1}^\top\mathbf{Q}\mathbf{y}_{k+1} + \mathbf{u}_k^\top\mathbf{R}\mathbf{u}_k$, which is a function of the predicted FBL states $\mathbf{y}_{k+1}$ and the current sequence of control inputs $\mathbf{u}_k$. By substituting $\mathbf{u}_k$, $\mathbf{y}_{k+1}$, and (4) to the cost function, we obtain

$$\begin{aligned}
J(\Delta\mathbf{u}_k) =& (\mathbf{L}\Delta\mathbf{z}_k + \mathbf{M}\Delta\mathbf{u}_k + \mathbf{y}_k)^\top\mathbf{Q}(\mathbf{L}\Delta\mathbf{z}_k + \mathbf{M}\Delta\mathbf{u}_k \\
&+ \mathbf{y}_k) + (\Delta\mathbf{u} + \mathbf{u}_{k-1})^\top\mathbf{R}(\Delta\mathbf{u} + \mathbf{u}_{k-1})\,.
\end{aligned}$$

Because J is quadratic with respect to $\Delta\mathbf{u}_k$ and is convex, the optimal sequence of control updates, denoted $\Delta\mathbf{u}_k^*$, that minimizes J is found by solving $\partial J(\Delta\mathbf{u}_k)/\partial\Delta\mathbf{u}_k = 0$ as

$$\Delta\mathbf{u}_k^* = -(\mathbf{M}^\top\mathbf{Q}\mathbf{M} + \mathbf{R})^{-1}(\mathbf{M}^\top\mathbf{Q}(\mathbf{y}_k + \mathbf{L}\Delta\mathbf{z}_k) + \mathbf{R}\mathbf{u}_{k-1}). \tag{5}$$

By taking the optimal sequence of control updates $\Delta\mathbf{u}_k^*$ produced by (5) and adding it to the previous sequence of control inputs $\mathbf{u}_{k-1}$, we generate a new optimal sequence, $\mathbf{u}_k^* = \Delta\mathbf{u}_k^* + \mathbf{u}_{k-1}$. The first input from $\mathbf{u}_k^*$, defined as u_k^*, is used in real-time on the vehicle by applying the steering rate $\omega_k^* = u_k^*/(v\cos\varepsilon_{H,k})$ to steer the vehicle along the desired path. Note that the matrices $\left(\mathbf{M}^\top\mathbf{Q}\mathbf{M} + \mathbf{R}\right)^{-1}$ and $\mathbf{M}^\top\mathbf{Q}$ are constants and thus, need to be computed only once, which is a key difference between FBLMPC and NMPC.

B. Gaussian Processes Regression

Gaussian process (GP) regression has become one of the most commonly employed machine learning techniques in learning-based control [5]. Consider m input data points $\mathbf{a} = [\mathbf{a}_1, \cdots, \mathbf{a}_m]^\mathsf{T} \in \mathbb{R}^{n_a \times m}$ and the corresponding measurements $\mathbf{d} = [\mathbf{d}_1, \cdots, \mathbf{d}_m]^\mathsf{T} \in \mathbb{R}^{n_d \times m}$, related through an unknown function $\mathbf{d}_k = \mathbf{g}(\mathbf{a}_k) + \boldsymbol{\omega}_k$: $\mathbb{R}^{n_z} \to \mathbb{R}^{n_d}$, where $\boldsymbol{\omega}_k$ is i.i.d. Gaussian noise with $\boldsymbol{\omega}_k \sim \mathcal{N}(0, \boldsymbol{\Sigma}^{\boldsymbol{\omega}})$ and $\boldsymbol{\Sigma}^{\boldsymbol{\omega}} = \mathrm{diag}([\sigma_1^2, \cdots, \sigma_{n_d}^2])$. The function $\mathbf{g}(\cdot)$ can be identified by the observed input-output dataset $\mathcal{D}_m = \{\mathbf{a}, \mathbf{d}\}$. Assume each output dimension of $\mathbf{d}_k$ is independent of each other given the input $\mathbf{a}_k$. For each dimension $M \in \{1, \cdots, n_d\}$ of the function output $\mathbf{d}_k$, specifying a GP with zero mean and prior kernel $k^M(\cdot,\cdot)$, the measurement data $[\mathbf{d}_k]_{M,\cdot}$ is normally distributed as $[\mathbf{d}_k]_{M,\cdot} \sim \mathcal{N}\left(0, K_{\mathbf{aa}}^M + \sigma_M^2\right)$, where $K_{\mathbf{aa}}^M$ is the Gram matrix of the data points; i.e. $[K_{\mathbf{aa}}^M]_{ij} = k^M(\mathbf{a}_i, \mathbf{a}_j)$. The choice of kernel functions $k^M(\cdot,\cdot)$ is specified by a prior knowledge of the observed data. In this work, we use the Squared Exponential (SE) kernel like many other GP learning-based robotic control applications [2],

$$K^M(\mathbf{a}, a) = \sigma_{f,M}^2 \exp\left(-\frac{1}{2}(\mathbf{a} - a)^\mathsf{T} L_M^{-1}(\mathbf{a} - a)\right), \tag{6}$$

where a represents new data points where predictions are made, $\sigma_{f,M}^2$ and $L_M \in \mathbb{R}^{n_a \times n_a}$ are hyperparameters [10] that are optimized by the log marginal likelihood function [11]. The GP models with the optimized hyperparameters are the trained models by using the observed dataset [10].

In the output dimension M, the joint distribution of the observed output $[\mathbf{d}]_{M,\cdot}$ and the prediction output $[\mathbf{d}]_M$ at new data points a is $P([\mathbf{d}]_{M,\cdot}, [\mathbf{d}]_M \mid \mathbf{a}, a)$, and

$$\begin{bmatrix}[\mathbf{d}]_{M,\cdot} \\ [\mathbf{d}]_M\end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix}K_{\mathbf{aa}}^M + I\sigma_M^2 & K_{\mathbf{a}a}^M \\ K_{a\mathbf{a}}^M & K_{aa}^M\end{bmatrix}\right), \tag{7}$$

where $[K_{\mathbf{a}a}^M]_j = k^M(a_j, a)$, $K_{a\mathbf{a}}^M = (K_{\mathbf{a}a}^M)^\mathsf{T}$, and $K_{aa}^M = k^M(a, a)$. The posterior distribution of $[\mathbf{d}]_M$ conditioned on the observed data is also Gaussian, by using the Marginal and Conditional Distributions of Multivariate Normal Distribution theorem [10], it can be derived from (7) as $P([\mathbf{d}]_M \mid [\mathbf{d}]_{M,\cdot}, \mathbf{a}, a) = \mathcal{N}\left(\mu^M(a), \Sigma^M(a)\right)$ with

$$\mu^M(a) = K_{a\mathbf{a}}^M\left(K_{\mathbf{aa}}^M + I\sigma_M^2\right)^{-1}[\mathbf{d}]_{M,\cdot}, \tag{8}$$

$$\Sigma^M(a) = K_{aa}^M - K_{a\mathbf{a}}^M\left(K_{\mathbf{aa}}^M + I\sigma_M^2\right)^{-1}K_{\mathbf{a}a}^M. \tag{9}$$

The resulting GP model estimation $\mathbf{d}(\cdot)$ of the unknown function $\mathbf{g}(\cdot)$ is obtained by vertically concatenating the individual output dimension $M \in \{1, \cdots, n_d\}$ as

$$\mathbf{d}(a) \sim \mathcal{N}\left(\boldsymbol{\mu}^{\mathbf{d}}(a), \boldsymbol{\Sigma}^{\mathbf{d}}(a)\right), \tag{10}$$

with $\boldsymbol{\mu}^{\mathbf{d}}(a) = [\mu^1(a), \cdots, \mu^{n_d}(a)]^\mathsf{T} \in \mathbb{R}^{n_d}$ and $\boldsymbol{\Sigma}^{\mathbf{d}}(a) = \mathrm{diag}\left([\Sigma^1(a), \cdots, \Sigma^{n_d}(a)]^\mathsf{T}\right) \in \mathbb{R}^{n_d \times n_d}$.

C. Combining GP Modelling with MPC

Consider a nonlinear system $\mathbf{x}_{k+1} = \mathbf{f}_{\text{true}}(\mathbf{x}_k, \mathbf{u}_k)$ with observable states $\mathbf{x}_k \in \mathbb{R}^n$ and control input $\mathbf{u}_k \in \mathbb{R}^m$. The true system model can be represented by the sum of a nominal model and a learning-based model as

$$\mathbf{x}_{k+1} = \overbrace{\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)}^{\text{nominal model}} + \overbrace{\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k)}^{\text{learning-based model}}. \tag{11}$$

The model $\mathbf{f}(\cdot)$ is a nominal process model representing our knowledge of $\mathbf{f}_{\text{true}}(\cdot)$, and $\mathbf{g}(\cdot)$ is a model representing discrepancies between the nominal model and the true system model. We would like to approximate $\mathbf{g}(\cdot)$ as GP models $\mathbf{d}(\cdot)$ that can be estimated by the observation data.

Because GP models output probability distributions, iterative predictions for multiple time steps are analytically intractable and the resulting distribution is no longer Gaussian [12]. Many approximation methods for multi-step-ahead

predictions have been proposed [13]. A common approach is to approximate the distributions of the states at each prediction time step as a Gaussian $x_k \sim \mathcal{N}(\mu_k^x, \Sigma_k^x)$ [14]. In this work, we only utilize the mean predictions of the GP models in the controller. By applying a first-order Taylor approximation about $\mathbf{x}_k = \boldsymbol{\mu}_k^{\mathbf{x}}$ and $\mathbf{u}_k = \boldsymbol{\mu}_k^{\mathbf{u}}$ of the nominal model $\mathbf{f}(\cdot)$ and posterior mean function (10) following [14], the mean prediction approximations of the system model represented by (11) is derived as

$$\mathbf{x}_{k+1} \approx \mathbf{f}\left(\boldsymbol{\mu}_k^{\mathbf{x}}, \boldsymbol{\mu}_k^{\mathbf{u}}\right) + \boldsymbol{\mu}^{\mathbf{d}}\left(\boldsymbol{\mu}_k^{\mathbf{x}}, \boldsymbol{\mu}_k^{\mathbf{u}}\right) . \tag{12}$$

IV. Implementation

Similar to [1], let the kinematics and dynamics of our vehicle be given by

$$\text{kinematics: } \mathbf{x}_{k+1} = \mathbf{f}_{\mathbf{x},\text{ true}}\left(\mathbf{x}_k, \mathbf{v}_k\right) , \tag{13}$$

$$\text{dynamics: } \mathbf{v}_{k+1} = \mathbf{f}_{\mathbf{v},\text{ true}}\left(\mathbf{v}_k, \mathbf{u}_k\right) , \tag{14}$$

where $\mathbf{x}_k = (x_k, y_k, \theta_k)$ represents the vehicle's pose, $\mathbf{v}_k = (v_k^{act}, \omega_k^{act})$ represents the actual velocity (linear and rotational speed) of the vehicle, and $\mathbf{u}_k = (v_k^{cmd}, \omega_k^{cmd})$ are the control inputs (linear and angular velocity) at time k. By substituting $\mathbf{v}_k = \mathbf{f}_{\mathbf{v}\text{ ,true}}\left(\mathbf{v}_{k-1}, \mathbf{u}_{k-1}\right)$ into (13), the dynamics can be cascaded into the kinematics as $\mathbf{x}_{k+1} = \mathbf{f}'_{\text{true}}\left(\mathbf{x}_k, \mathbf{v}_{k-1}, \mathbf{u}_{k-1}\right)$. If we assume the robot dynamics are negligible, then the actual velocity $\mathbf{v}_k$ equals the commanded ones $\mathbf{u}_k$, thus the true process model, $\mathbf{f}'_{\text{true}}(\cdot)$ can be represented by the sum of the nominal and learning-based models as

$$\mathbf{x}_{k+1} = \overbrace{\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)}^{\text{nominal model}} + \overbrace{\mathbf{g}(\underbrace{\mathbf{x}_k, \mathbf{v}_{k-1}, \mathbf{u}_k, \mathbf{u}_{k-1}}_{a_k})}^{\text{learning-based model}}, \tag{15}$$

with the disturbance, $a_k = (\mathbf{x}_k, \mathbf{v}_{k-1}, \mathbf{u}_k, \mathbf{u}_{k-1})$, yielding

$$\mathbf{x}_{k+1} - \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right) = \mathbf{g}(a_k) . \tag{16}$$

In GP-FBLMPC, use the FBL states $\mathbf{z}_k$ as the system state variables instead of the robot pose states $\mathbf{x}_k$. The left side of (16) are the difference between the robot actual and predictive pose state $\mathbf{x}_{k+1}$, which are the FBL states, as explained in Section III-A. The learning-based model of the FBL states is

$$\mathbf{z}_{k+1} = \mathbf{g}(a_k) , \tag{17}$$

with the disturbance state, $a_k = (\mathbf{z}_k, \mathbf{v}_{k-1}, \mathbf{u}_k, \mathbf{u}_{k-1})$. If we represent the learning-based model $\mathbf{g}(\cdot)$ as GP models $\mathbf{d}(\cdot)$, by (12), the mean prediction of (17) is

$$\mathbf{z}_{k+1} \approx \boldsymbol{\mu}^{\mathbf{d}}(a_k) . \tag{18}$$

We estimate a separate GP model for each dimension of $\mathbf{g}(\cdot) \in \mathbb{R}^{n_d}$ to model disturbances assuming that they are not correlated. In the FBLMPC, there are two FBL states, thus the output dimension $n_d = 2$. In each trial of the path following experiment, the observed input-output disturbance data at all time steps are collected as a disturbance dataset $\mathcal{D}_T = \left\{\mathcal{D}_m^1 = \{\mathbf{a}, [\mathbf{d}]_{\cdot,1}\}, \mathcal{D}_m^2 = \{\mathbf{a}, [\mathbf{d}]_{\cdot,2}\}\right\}$, with m means the total time steps to finish a trial. At every time step k, the output data $\mathbf{d}_k$ is calculated by substituting $k-1$ to (18) as

$$\mathbf{d}_k = \mathbf{g}\left(\mathbf{a}_{k-1}\right) = \hat{\mathbf{z}}_k , \tag{19}$$

which equals to the FBL states measurement $\hat{\mathbf{z}}_k$. The disturbance state $\mathbf{a}_{k-1} = (\hat{\mathbf{z}}_{k-1}, \hat{\mathbf{v}}_{k-2}, \mathbf{u}_{k-1}, \mathbf{u}_{k-2})$ are the input data $\mathbf{a}$ of the observation dataset $\mathcal{D}_T$. The input-output dataset $\mathcal{D}_T$ is used to train GP models after finishing a trial.

In the MPC prediction horizon loop, we use GP models $\mathbf{d}(\cdot)$ trained by the dataset $\mathcal{D}_T$ (observation data in previous trials) to make mean predictions by applying (8). All the parameters of $a_{k+i} = \left(\mathbf{z}_{k+i}, \mathbf{v}_{k-1+i}, \mathbf{u}_{k+i}, \mathbf{u}_{k-1+i}\right), i \in K = \{0, \cdots, p-1\}$ are required. The FBL state variables $\mathbf{z}$ are obtained from our guidance systems [3, Sec. 4.2]. The third and forth variables $\mathbf{u}$ are known because they are the commanded control inputs. The velocity state variables $\mathbf{v}_{k-1+i} = \begin{bmatrix} v_{k-1+i}^{act} & \omega_{k-1+i}^{act} \end{bmatrix}^{\mathsf{T}}$ are calculated as

$$v_{k-1+i}^{act} = \frac{\sqrt{\left(x_{k+i} - x_{k-1+i}\right)^2 + \left(y_{k+i} - y_{k-1+i}\right)^2}}{T} ,$$

$$\omega_{k-1+i}^{act} = \frac{\left(\theta_{k+i} - \theta_{k-1+i}\right)}{T} .$$

with $x_{k-1} = \hat{x}_{k-1}$. The robot pose (x, y, θ) come from our navigation (localization) system.

The Algorithm 1 summarizes how to implement the GP-FBLMPC algorithm in a way that exploits the feedback linearized coordinates for MPC optimization and a prior kinematic nominal process model together with GP learning-based disturbance models for the prediction of future path-following errors. In this algorithm, the current vehicle state $\mathbf{q}_k$ and path following errors $(\varepsilon_{L,k}, \varepsilon_{H,k})$ are used in the first step when $i = 0$.

V. Field Testing

We tested the proposed GP-FBLMPC algorithm by extensive field experiments in five experiments with three different paths (Fig. 4) in sand and grass terrains. The five experiments demonstrated the algorithm was able to reduce the path following errors effectively. Compared to the GP-NMPC algorithm, the GP-FBLMPC was more computationally efficient, and the estimated disturbance GP models were generalizable to reduce the path following errors on different paths with the same terrain profile.

A. Overview

The robot used in all the field experiments was a skid-steer Clearpath Husky A200 mobile robot (Fig. 1). The computer used to operate the vehicle was a Dell XPS 15 laptop with an eight-core 2.80 GHz Intel Core i7-7700HQ processor and 12 GB of RAM that was running ROS Kinetic on an Ubuntu 16.04 operating system. Fig. 3 shows a block diagram of implemented path following control by using the GP-FBLMPC algorithm. The outdoor navigation was accomplished using a LORD microstrain 3DM-GX5-25 IMU (gyroscope) and a Swift Navigation Duro GPS with an RTK (Real-time Kinematic) base station with a Novatel FlexPak6 GPS receiver shown in Fig. 1. The vehicle guidance and control systems were written in Python, the outdoor navigation used the open-source ROS package *robot_localization* to fuse GPS and IMU data for state estimation. The navigation provided position accuracy of approximately ±1 cm, which was sufficient for field

Algorithm 1 GP-FBLMPC at time step k.

Input: $i = 0$, $\mathbf{q}_k = \mathbf{q}_{k+i}$, $\varepsilon_{L,k}$, $\varepsilon_{H,k}$, $\mathbf{u}_0 = \mathbf{0}$,
Output: ω_k^*

$$\begin{bmatrix}\mathbf{y}_{k,0}\\ \mathbf{y}_{k,1}\end{bmatrix} = \begin{bmatrix}z_{1,k}\\ z_{2,k}\end{bmatrix} = \begin{bmatrix}\varepsilon_{L,k}\\ v\sin\varepsilon_{H,k}\end{bmatrix} = \begin{bmatrix}\mathbf{d}_{k,1}\\ \mathbf{d}_{k,2}\end{bmatrix} = \mathbf{g}\left(\mathbf{a}_{k-1}\right)$$

$$\mathbf{a}_{k-1} = \left(\hat{\mathbf{z}}_{k-1}, \hat{\mathbf{v}}_{k-2}, \mathbf{u}_{k-1}, \mathbf{u}_{k-2}\right)$$

Save the input-output data pair $[\mathbf{a}_{k-1}, \mathbf{g}\left(\mathbf{a}_{k-1}\right)]$ to the disturbance dataset $\mathcal{D}_T$

for $i = 0$ **to** $p - 1$ **do**

$$\omega_{k+i} = \frac{\eta_{k+i}}{v\cos\varepsilon_{H,k+i}}$$

$$\mathbf{q}_{k+i+1} = \begin{bmatrix}x_{k+i}\\ y_{k+i}\\ \theta_{k+i}\end{bmatrix} + T\begin{bmatrix}\cos\theta_{k+i} & 0\\ \sin\theta_{k+i} & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega_{k+i}\end{bmatrix}$$

Calculate current $\varepsilon_{L,k+i+1}$ and $\varepsilon_{H,k+i+1}$

Make predictions by the GP models $\boldsymbol{d}_1(\cdot)$ and $\boldsymbol{d}_2(\cdot)$ at $a_{k+i} = (\mathbf{z}_{k+i}, \mathbf{v}_{k-1+i}, \mathbf{u}_{k+i}, \mathbf{u}_{k-1+i})$ as $m\left(\mathbf{d}_1(a_{k+i})\right) = \mu_1^{\mathbf{d}}(a_{k+i})$, $m\left(\mathbf{d}_2(a_{k+i})\right) = \mu_2^{\mathbf{d}}(a_{k+i})$

$$\begin{bmatrix}\mathbf{y}_{k,2i+2}\\ \mathbf{y}_{k,2i+3}\end{bmatrix} = \begin{bmatrix}\varepsilon_{L,k+i+1}\\ v\sin\varepsilon_{H,k+i+1}\end{bmatrix} + \begin{bmatrix}\mu_1^{\mathbf{d}}(a_{k+i})\\ \mu_2^{\mathbf{d}}(a_{k+i})\end{bmatrix}$$

end for

Calculate $\Delta\mathbf{u}_k^*$ using (5)

$$\omega_k^* = \frac{\eta_k^*}{v\cos\varepsilon_{H,k}}$$

Constrain ω_k^* to ± 2 rad/s for maximum turning rate

return ω_k^*

Two GP models $\mathbf{d}_1$ and $\mathbf{d}_2$ are estimated by the disturbance dataset $\mathcal{D}_T$ when a trial is finished.

test experiments. The robot localization EKF was operating at 30 Hz, while all path following control was operating at 10 Hz.

In all of the field experiments, the Husky robot was controlled to drive at a constant desired forward speed at 0.9 m/s (maximum 1.0 m/s) to avoid possible motor saturation. The speed was chosen mainly because that path following tasks at high speed is more challenging compared to low-speed applications [3]. There were three paths, infinite, track, and infinite3 were designed as desired paths in the field experiments shown in Fig. 4. Each path consisted of a sequence of evenly and closely spaced (0.05 m apart) discrete waypoints to specify the desired vehicle pose. In each path, starting at the red circle and orienting to the right, the Husky A200 robot drove autonomously to follow the path and finished at the green star. The FBLMPC was tuned to give a desirable path following behavior before integrating GP models. The optimal tuning found was $p = 10, \mathbf{Q} = 5.0 \times \mathbf{I}_{2p\times 2p}, \mathbf{R} = \mathbf{I}_{p\times p}$. When running NMPC, the number of optimization iterations was limited to allow the controller to complete its calculations within 0.1 s. It was found that the NMPC performed best

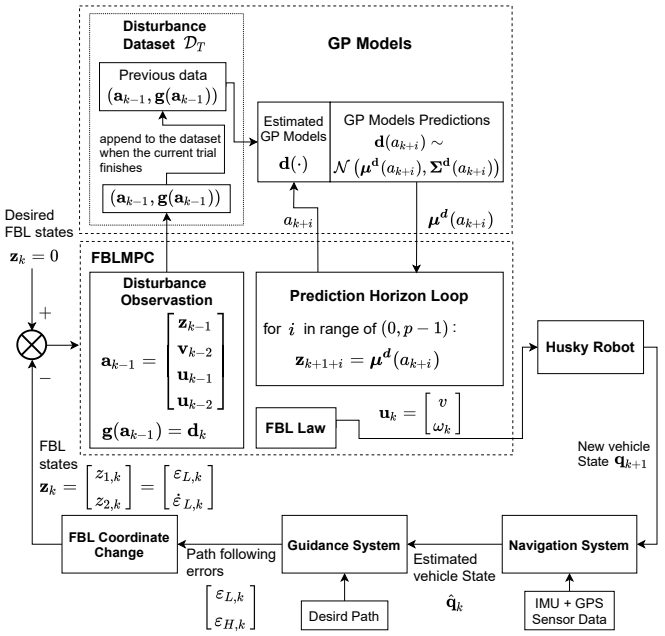

Fig. 3. Block diagram of GP-FBLMPC.

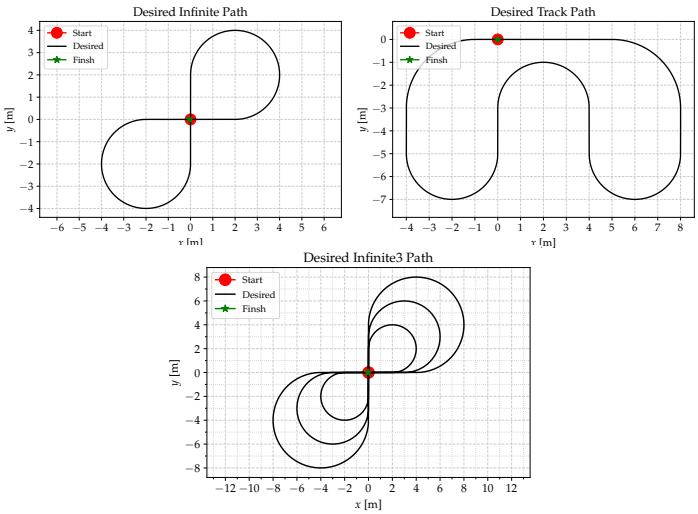

Fig. 4. Three paths, infinite, track, and infinite3, were designed and used in the field experiments of the path following experiments.

with $p = 20$ and six iterations and the optimal tuning was $\mathbf{Q} = \mathbf{I}_{3p\times 3p}$ and $\mathbf{R} = 4.0 \times \mathbf{I}_{2p\times 2p}$. The complete controller parameters tuning process can be found in [3, Sec. 4.4].

B. Experiment 1: GP-FBLMPC reduce path-following errors compare to FBLMPC

In the first experiment, the robot followed the infinite path in a sand terrain shown in Fig. 5. In trial 1, the GP-FBLMPC was the same as FBLMPC. In trial 2, FBLMPC was integrated with GP models trained by the observation data collected in trial 1. In trial 1 and 2, we repeated 10 tests respectively.

Fig. 6 plotted the maximum and root mean square (RMS) of the lateral and heading errors, ε_L and ε_H vs. the test number respectively in trial 1 and 2. It's obvious that the lateral and heading errors decreased over the course of the 10 tests in

Fig. 5. The Husky A200 robot drove autonomously to follow the infinite path in a sand terrain. The desired infinite path (black-dot line) was blended to the field test scene image took by a DJI Mini.

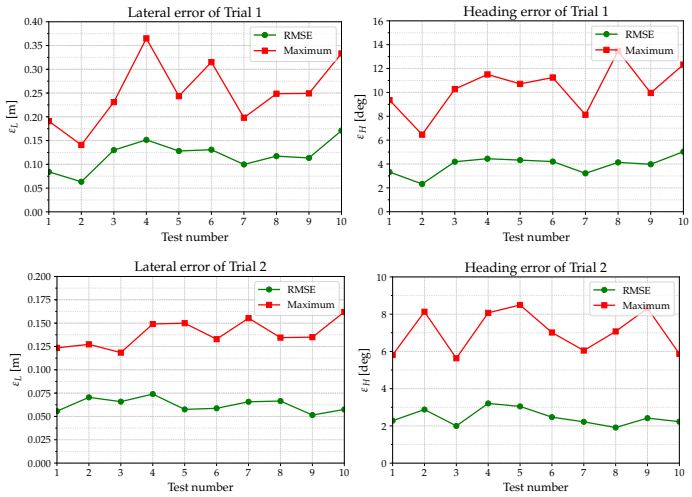

Fig. 6. The GP-FBLMPC algorithm was tested on the infinite path in sand terrains. The maximum and RMS of the lateral and heading errors vs. test number in trial 1 and trial 2 were plotted.

trial 2. We calculated the mean values to reflect the overall performance in each trial. There were 85.31% lower lateral error mean and 59.2% lower heading error mean in trial 2 than trial 1. We sorted the 10 tests by the RMS of the lateral errors in trial 1 and 2, and treated the test with the highest RMS lateral error as an outlier in each trial. Fig. 7 plotted the path following errors (lateral and heading) and the commanded steering inputs vs. travel time of the maximum, minimum, and median ones of the remaining 9 tests in trial 1 and 2 respectively. It can be seen that the overall magnitude of the lateral and heading errors ε_L and ε_H were smaller in trial 2.

Fig. 8 plotted the GP predictions and the actual lateral and

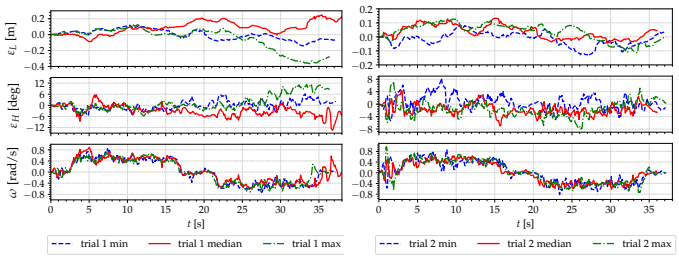

Fig. 7. The path following errors and the commanded steering inputs vs. travel time of the maximum, minimum and median ones of the 9 tests in trial 1 and 2 respectively for the GP-FBLMPC on the infinite path.

heading errors of trial 2 (the one with maximum RMS lateral error among the 9 tests) vs. the travel time. The GP model predictions of the lateral errors corresponded the actual lateral errors well. Even there were more deviations between the predictive and actual heading errors, the estimated GP model of the heading error still provided good mean predictions of the actual heading errors. Together with the significant lower path following error results in trial 2 (Fig. 6), we can conclude that the estimated GP models effectively learned the unmodeled dynamics and was capable to provide good predictions that are close to the actual path following errors.

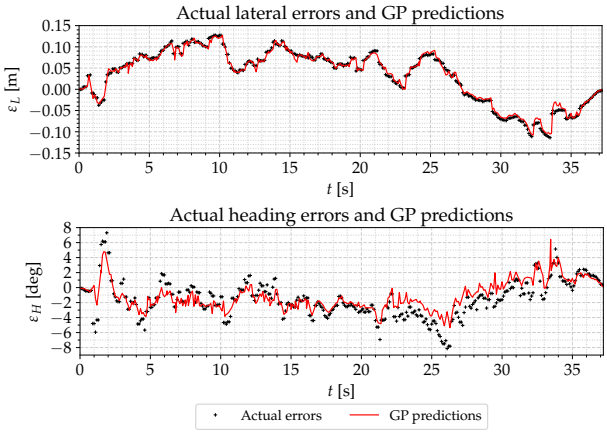

Fig. 8. The lateral and heading errors of the GP model predictions and the actual errors vs. travel time in trial 2 of the GP-FBLMPC on the infinite path.

C. Experiment 2: GP models trained by one trial data works equally well to the ones trained by multiple trials

In the second experiment, we continued to test the GP-FBLMPC algorithm on the infinite path (Fig. 5) over 9 more field trials. We demonstrated that the GP models trained by one trial data (trial 2) worked equally well to the GP models estimated with multiple trials data (trial 3-11). In trial 3-11, like what we did in trial 1 and 2 in Section V-B, 10 tests were repeated in each trial. Similar to the data processing in trial 1 and 2, we selected the test with median RMS lateral error in each trial to demonstrate their "typical" performance. Fig. 9 plotted the maximum and RMS path following errors (lateral and heading) of the median one in each trial vs. trial number. Similar to the mean of path following errors (Fig. 6), the median of path following errors were also reduced obviously in trial 2. The lateral and heading errors were reduced by roughly 80% and 60% respectively starting in trial 2 and over the course of the following all trials.

There were no notable improvements of the RMS path following errors in trial 3-11 compared to trial 2. In fact, it can be seen that the maximum and RMS path following errors varied over the course of the trial 2-11. The similar phenomenons that the maximum and RMS path following errors continued to vary even after many trials and the RMS path following errors maintained similar over trials were also noticed in [1]. We believed that the RMS errors maintained similar due to two main reasons: i) the disturbance data collected in one trial was enough to train GP models to represent the unmodeled

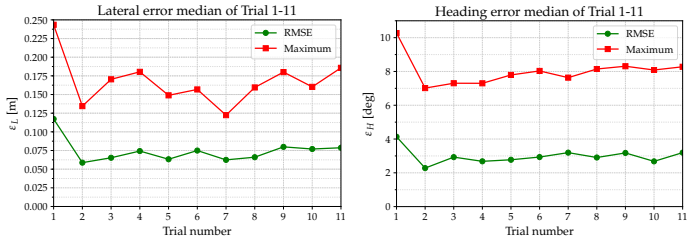

Fig. 9. The maximum and RMS path following errors vs. trial number of the GP-FBLMPC algorithm tested on the infinite path were plotted. In each trial, the plotted dot was the one with median RMS lateral error among 9 tests to demonstrate the "typical" performance.

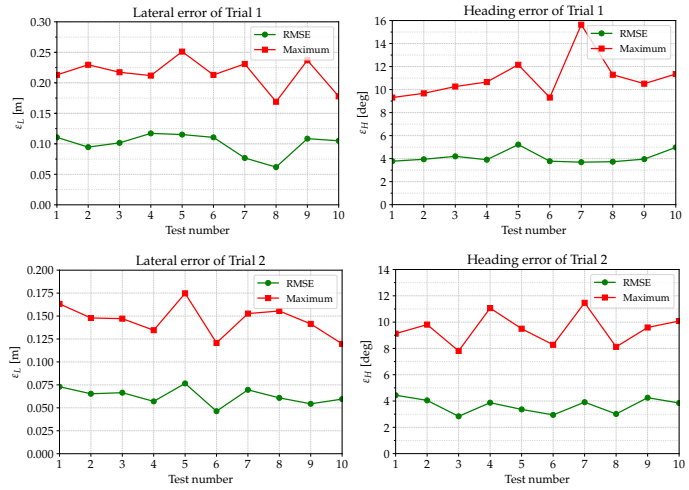

Fig. 10. The GP-NMPC algorithm was tested on the infinite path in sand terrains. The maximum and RMS of the lateral and heading errors vs. test number in trial 1 and trial 2 were plotted. This plot is compared to the path following performance of the GP-NMPC to the GP-FBLMPC shown in Fig. 6.

dynamics that can make good predictions, and ii) evolving sand terrain conditions of the path made the newly collected data in the following trials was used to identify the changing wheel-terrain dynamics. In practice, we can use GP models trained by a reasonable small amount of data without updating them if there is no major change in the operating terrain.

D. Experiment 3: GP-FBLMPC and GP-NMPC comparisons

In this experiment, the GP-FBLMPC and GP-NMPC performance were compared with respect to path following errors and computational time. The robot followed the infinite path in sand terrain shown in Fig. 5 for 10 tests in trial 1 and 2. Fig. 10 plotted the maximum and RMS of the lateral and heading errors vs. the test number in trial 1 and 2 respectively. In trial 1, compared to the path following errors of the FBLMPC shown in Fig. 6, the NMPC behaved relatively better. The mean of the RMS lateral errors of the NMPC is 15.1% lower than the FBLMPC, while the FBLMPC had a 5.3% lower heading error mean than the NMPC. After integrating estimated GP models in trial 2, as shown in Fig. 6 and 10, the GP-FBLMPC performed slightly better than the GP-NMPC. The GP-FBLMPC had a 0.6% lower lateral error mean and 49.6% lower heading error mean than the GP-NMPC.

The average time of GP-FBLMPC and GP-NMPC to calculate a single control input equals approximately the sum of the MPC computation time and GP models prediction time in each

Fig. 11. In the Experiment 4, the Husky A200 robot drove autonomously to follow the track path in a sand terrain. The desired track path (black-dot line) was blended to the field test scene image took by a DJI Mini.

time step. The FBLMPC algorithm is more computationally efficient than the NMPC because it does not require iterative optimization [3]. In trial 2, The GP model prediction time of GP-FBLMPC and GP-NMPC was 0.0016 and 0.0022 respectively. In the following trials, the GP prediction time grew as more observation disturbance data was used for the GP models estimation. In trial 7, For example, the averaged GP models prediction time of the GP-NMPC was approximately 0.0346 s, while the averaged prediction time of the GP-FBLMPC was approximately 0.0233 s. The averaged GP models prediction time in GP-FBLMPC was approximately $2/3$ of the time in GP-NMPC in each trial. This is because at each time step, there were two GP models to make predictions in the GP-FBLMPC, while three in the GP-NMPC.

In summary, the GP-FBLMPC behaved equally well as the GP-NMPC with respect to the path following errors. The GP-FBLMPC is more computationally efficient than the GP-NMPC because it does not conduct iterative optimization and requires fewer numbers of GP models to make predictions.

E. Experiment 4: GP-FBLMPC is generalizable to reduce path following errors for different paths

The Husky A200 robot was tested on the track path in a sand terrain (Fig. 11) to show the generalizability of the GP-FBLMPC, meaning GP models of the GP-FBLMPC estimated on one path are able to reduce the path following errors for other paths. Instead of estimating new GP models with observation data of the track path in trial 1, we integrated the GP models trained by the infinite path data. In trial 1, the maximum and RMS lateral and heading errors of FBLMPC and NMPC vs. test number were plotted in Fig. 12 and Fig. 13 respectively. The mean of the NMPC lateral errors was 4.31% lower than the FBLMPC, while the FBLMPC had a 9.7 % lower heading error mean than the NMPC.

In trial 2, the GP-FBLMPC integrated GP models estimated in Section V-B. Fig. 12 showed the GP-FBLMPC gained 86.14% lower lateral error mean and 17.29% lower heading error mean compared to trial 1. On the contrary, after integrating GP models estimated in Section V-D, the GP-NMPC had a worse performance than trial 1 shown in Fig. 13. Fig. 14 showed GP predictions and actual lateral and heading errors of the GP-FBLMPC in trial 2 . The GP predictions correspond the actual errors reasonably well. The GP-FBLMPC improved

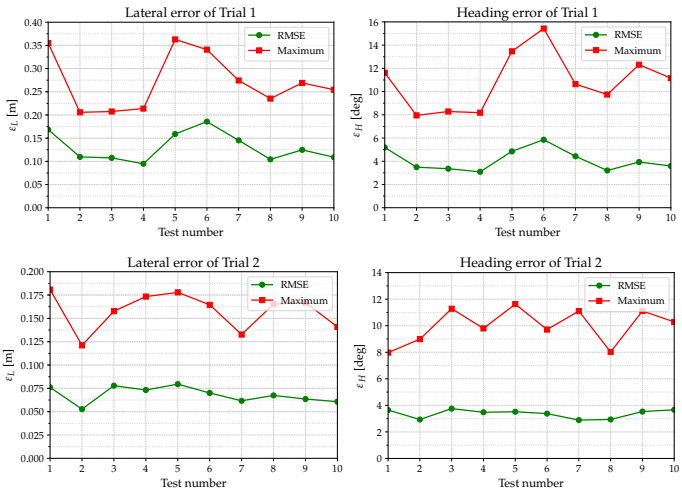

Fig. 12. The GP-FBLMPC algorithm was tested on the track path in sand terrains. The maximum and RMS of the lateral and heading errors vs. test number in trial 1 and trial 2 were plotted.

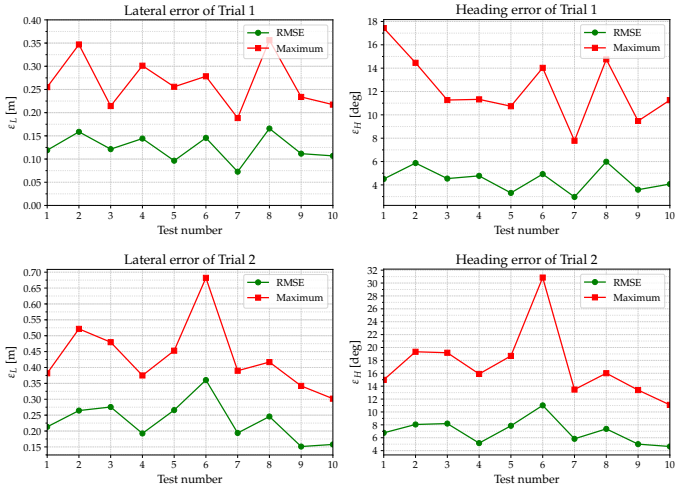

Fig. 13. The GP-NMPC algorithm was tested on the track path in sand terrains. The maximum and RMS of the lateral and heading errors vs. test number in trial 1 and trial 2 were plotted.

performance after integrating GP models estimated on the infinite path, while the GP-NMPC showed a worse performance by integrating GP models estimated on the infinite path. As explained in Section IV, GP models of the GP-FBLMPC and GP-NMPC were estimated by the observation data with the FBL states $\mathbf{z}$ and robot poses (x, y, θ) as inputs respectively. Thus, the GP-FBLMPC algorithm is generally capable of reducing path following errors for different paths, the GP-NMPC only works on the same path it was trained.

F. Experiment 5: GP-FBLMPC follows a long complex path

We further demonstrated the generalizability of the GP-FBLMPC algorithm on a more complex infinite3 path (Fig. 4) in a grass terrain shown in Fig. 15. The robot first drove autonomously on the infinite path to collect disturbance observation data for training GP models. In trial 1, the robot was commanded to follow the infinite3 path by the FBLMPC. The estimated GP models were integrated to the GP-FBLMPC in trial 2. Three tests were repeated in trial 1 and 2, the

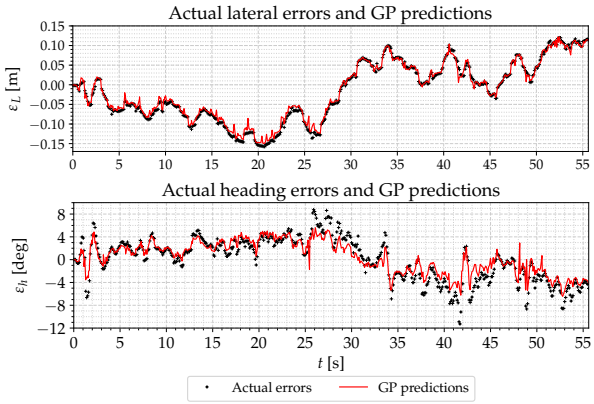

Fig. 14. The lateral and heading errors of the GP model predictions and the actual errors vs. travel time in trial 2 for the GP-FBLMPC on the track path.

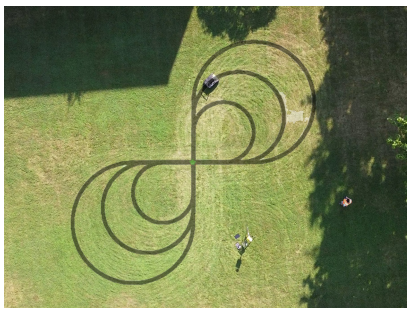

Fig. 15. The Husky A200 robot drove autonomously to follow the infinite3 path in a grass terrain. The desired infinite3 path (black-dot line) was blended to the field test scene image took by a DJI Mini.

maximum and RMS lateral and heading errors of trial 1 and 2 vs. test number were plotted shown in Fig. 16. In trial 2, there were 90.94 % lower lateral error mean and 39.63 % lower heading error mean than in trial 1. Fig. 17 showed the GP model predictions together with the actual lateral and heading errors of the one with the maximum RMS lateral error mean among the three tests. It can be seen that the predictions of GP disturbance models were close to the actual path following errors. The generalizability of the GP-FBLMPC algorithm to reduce path following errors was validated on the challenging infinite3 path.

VI. Conclusion

This paper presents a Gaussian Processes Learning-based Feedback Linearized Model Predictive Control (GP-FBLMPC)

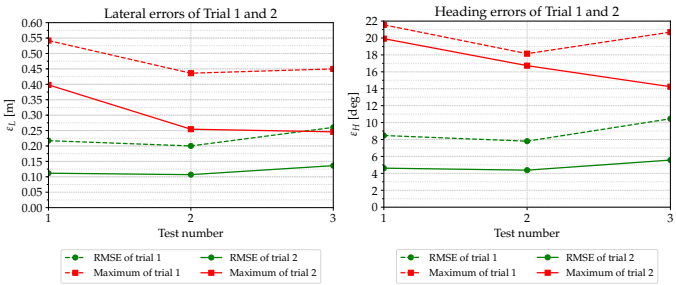

Fig. 16. The GP-FBLMPC algorithm was tested on the infinite3 path in grass terrains. The maximum and RMS of the path following errors vs. test number in trial 1 and trial 2 were plotted.

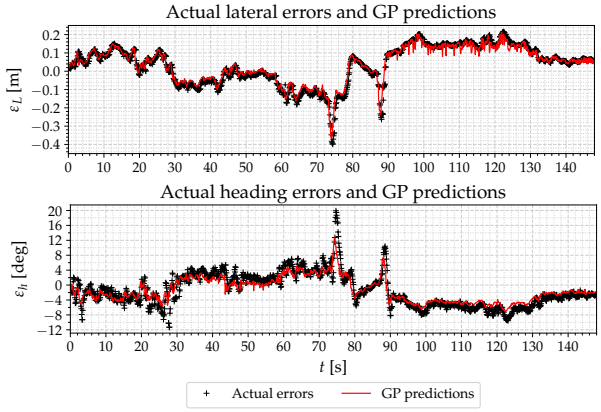

Fig. 17. The GP-FBLMPC lateral and heading errors of the GP model predictions and actual errors vs. travel time in trial 2 on the infinite3 path.

algorithm for path following control of ground mobile robots operating in off-road terrains. The GP-FBLMPC algorithm uses a fixed unicycle kinematic model and Gaussian Processes (GP) models to learn the unmodeled dynamics by the data collected in the field. Extensive outdoor tests of five experiments on Husky A200 mobile robot demonstrate the GP-FBLMPC algorithm is able to reduce the path following errors significantly than our previous FBLMPC algorithm [3]. When compared to a GP learning-based nonlinear MPC (GP-NMPC) algorithm [1], the GP-FBLMPC algorithm behaves equally well to reduce path following errors with a much higher computational efficiency and is generalizable to reduce path following errors for different paths.

In the current work, we didn't consider the uncertainty propagation in the GP model approximations. We only used the predicted mean values of the GP models. A GP model could become over-confident in the multiple-step ahead predictions, which might result in over-optimistic MPC solutions [15]. We focused on the practical application of GP-FBLMPC algorithm to mobile robots without considering the robustness of the controller, which is also an open question for learning-based control [5]. In further work, we plan to use the predicted uncertainty of the GP models explicitly to guarantee some level of the robust stability of the GP-FBLMPC algorithm.

VII. Acknowledgments

This research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) through the NSERC Canadian Robotics Network (NCRN).

References

[1] 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.

[2] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *arXiv preprint arXiv:2108.06266*, 2021.

[3] M. T. Fader, "Autonomous Ground Vehicle Path Following by Combining Feedback Linearization with Model Predictive Control," Master's thesis, Queen's University, 2020.

[4] A. Spitzer and N. Michael, "Feedback linearization for quadrotors with a learned acceleration error model," *arXiv preprint arXiv:2105.13527*, 2021.

[5] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.

[6] J. Umlauft, T. Beckers, M. Kimmel, and S. Hirche, "Feedback linearization using Gaussian processes," in *Proceedings of the Annual Conference on Decision and Control*. IEEE, 2017, pp. 5249–5255.

[7] M. Greeff and A. P. Schoellig, "Exploiting differential flatness for robust learning-based tracking control using Gaussian processes," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1121–1126, 2020.

[8] M. K. Helwa, A. Heins, and A. P. Schoellig, "Provably robust learning-based approach for high-accuracy tracking control of lagrangian systems," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1587–1594, 2019.

[9] L. G. Dekker, J. A. Marshall, and J. Larsson, "Industrial-scale autonomous wheeled-vehicle path following by combining iterative learning control with feedback linearization," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2017, pp. 2643–2648.

[10] J. Wang, "An intuitive tutorial to Gaussian processes regression," *arXiv preprint arXiv:2009.10862*, 2020.

[11] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes in machine learning*. MIT Press, 2006.

[12] K. Polymenakos, L. Laurenti, A. Patane, J.-P. Calliess, L. Cardelli, M. Kwiatkowska, A. Abate, and S. Roberts, "Safety guarantees for iterative predictions with Gaussian processes," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 3187–3193.

[13] J. Vinogradska, B. Bischoff, J. Achterhold, T. Koller, and J. Peters, "Numerical quadrature for probabilistic policy search," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 1, pp. 164–175, 2018.

[14] L. Hewing, A. Liniger, and M. N. Zeilinger, "Cautious NMPC with Gaussian process dynamics for autonomous miniature race cars," in *Proceedings of the European Control Conference*. IEEE, 2018, pp. 1341–1348.

[15] T. X. Nghiem, T.-D. Nguyen, and V.-A. Le, "Fast Gaussian process based model predictive control with uncertainty propagation," in *Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 2019, pp. 1052–1059.