Learning Probabilistic Models for Safe Predictive Control in Unknown Environments

Christopher D. McKinnon and Angela P. Schoellig

Abstract— **Researchers rely increasingly on tools from machine learning to improve the performance of control algorithms on real world tasks and enable robots to operate for long periods of time without intervention. Many of these algorithms require a model for the dynamics of the robot. In particular, researchers designing methods for safe learning control often rely on an upper bound on model error to make guarantees about the worst-case closed-loop performance of their algorithm. There are different options for how to learn such a model of the robot dynamics. We study probabilistic models for use in the context of stochastic model predictive control. Two popular choices for learning the robot dynamics are Gaussian Process (GP) regression and various forms of local linear regression. In this paper, we present a study comparing GPs with a particular form of local linear regression for learning robot dynamics with the aim of guaranteeing safety when a robot operates in novel conditions. We show results based on experimental data from a 900 kg ground robot using vision for localisation.**

I. INTRODUCTION

This paper presents a study comparing two probabilistic methods for modelling robot dynamics in the context of stochastic Model Predictive Control (MPC): Gaussian Process (GP) regression, and local, weighted Bayesian Linear Regression (BLR). In particular, we are interested in how these methods can enable a robot to operate in challenging and changing environments with minimal expert input and prior knowledge of the operating conditions. Our study is motivated by the growing popularity of GPs [1]–[4] and local linear regression [5]–[7] to model robot dynamics, and the interest in deploying robots in a wide range of operating conditions.

Safe control methods have emerged as a way to guarantee that safety constraints (e.g. a bound on maximum path tracking error) are kept in the face of model errors. The accuracy of the bound on model error is of critical importance to the validity of these safety guarantees. In order to derive such models for complex systems or systems operating in challenging operating conditions, researchers increasingly rely on tools from machine learning. In particular, probabilistic models are used since they provide a measure of model uncertainty which can naturally be used to derive an upper bound on model error. Two common methods for doing this are GP regression [1]–[4] and various forms of local linear regression [5]–[7].

GPs have recently gained a lot of popularity. They have been applied to problems such as modelling the dynamics of a robot performing repetitive path following [2], model car racing [1], and for a manipulator robot performing a tracking task [6]. While a naive implementation of a GP can be computationally intractable since it scales poorly in the number of training points, several solutions have been proposed including local GPs [8], [9], sparse GPs [1], and GPs using a special set of basis functions [7]. These alternatives make GPs computationally tractable in a control loop, even for controllers such as stochastic MPC which require evaluating the model for the robot dynamics many times at each timestep. In addition to being computationally feasible for stochastic MPC, GP-based models have been extended to learn the dynamics of a robot subjected to changes not seen in the training data [2]. Comparisons have shown that GPs can achieve higher accuracy and good prediction speed compared to local linear regression when GP approximations are used. These comparisons and the work using GPs for modelling robot dynamics typically consider fairly controlled environments such as a manipulator performing a tracking task [10] or a ground robot in a parking lot [2]. While there are notable exceptions, e.g. [3], where a ground robot was driven over challenging outdoor terrain, there has been no comparison to local linear regression in this case.

Local linear regression can also predict model uncertainty which makes it a good alternative for use in learning-based control [7]. While local linear regression can be used as a general nonlinear function approximator [5], it can also be used in combination with parametrized physical models which makes it very computationally efficient [11], [12]. In our previous work [12], we demonstrated that this approach has the potential to learn robot dynamics in a wide range of operating conditions and demonstrated that it is capable of enabling the robot to drive at high speed on challenging offroad terrain. In this paper, we compare the performance of the same safe controller with a GP-based model and a model based on local linear regression and show how the two models perform as the operating conditions are made increasingly challenging.

The rest of this paper is organized as follows. In Sec. II, we define our problem statement including the class of dynamics models being considered. In Sec. III, we provide a brief overview of GP regression and introduce local BLR, which are the two methods for model learning compared in this paper, and show a simple example illustrating a key difference that is relevant for stochastic control in changing environments. In Sec. IV, we describe how we construct local models for control, manage past data, and do uncertainty propagation, which is important for stochastic control. In Sec. V, we show how we apply both methods to a ground

The authors are with the Dynamic Systems Lab (www.dynsyslab.org) at the University of Toronto Institute for Aerospace Studies (UTIAS), Canada. email: chris.mckinnon@mail.utoronto.ca, schoellig@utias.utoronto.ca

robot, and, in Sec. VI, we present our experimental results.

Notation: We denote matrices with boldface uppercase letters, column vectors with boldface lowercase letters, and scalars with non-boldface letters. The identity matrix of appropriate size will be denoted by $\mathbf{I}$. The matrix trace operator is $tr(\cdot)$, and $||\mathbf{v}||^2_{\mathbf{M}} = \mathbf{v}^T\mathbf{M}\mathbf{v}$. The mean of a Gaussian random variable $\mathbf{v}$ is $\bar{\mathbf{v}}$. Functions are followed by round brackets, e.g. $\mathbf{v}(\cdot)$, if the output is a vector and $v(\cdot)$ if the output is a scalar. A probability density function (pdf) of v given parameters $*$ will be denoted by $p(v \mid *)$. A Gaussian pdf of v with mean μ and variance σ^2 will be denoted by $\mathcal{N}(v \mid \mu, \sigma^2)$, and $v \sim \mathcal{N}(\mu, \sigma^2)$ means that samples of v are distributed according to a Gaussian distribution with mean μ and variance σ^2. An Inverse Gamma distribution for variable v with shape parameter a and scale parameter b will be denoted $IG(v \mid a, b)$. A Student t distribution for variable v with mean μ, scale σ^2, and degrees of freedom ν will be denoted as $\mathcal{T}(v \mid \mu, \sigma^2, \nu)$.

II. Problem Statement

The goal of this work is to study probabilistic model learning for stochastic MPC for robots performing repetitive path following tasks in changing conditions. The key requirements for the model are: *(i)* high accuracy multi-step prediction to achieve high control performance; *(ii)* realistic bounds on model error to maintain safety; and *(iii)* computational efficiency to allow for a long prediction horizon in MPC.

We consider systems with dynamics of the form

$$\mathbf{s}_{k+1} = \mathbf{s}_k + dt \overbrace{\mathbf{f}(\mathbf{s}_k, \boldsymbol{\xi}_k)}^{known}, \tag{1}$$

$$\boldsymbol{\xi}_{k+1} = \underbrace{\mathbf{g}^0(\boldsymbol{\xi}_k, \mathbf{u}_k)}_{known} + dt \underbrace{\mathbf{g}_k(\mathbf{x}_k)}_{unknown}, \tag{2}$$

where the state of the system $\mathbf{s}$ evolves according to known dynamics $\mathbf{f}(\cdot)$ that depend on $\mathbf{s}$ and the state of the actuators $\boldsymbol{\xi}$. We assume that our control input $\mathbf{u}$ affects the actuator dynamics which consist of a known part $\mathbf{g}^0(\cdot)$ and an unknown and potentially changing part $\mathbf{g}_k(\cdot)$ that we wish to learn. The unknown dynamics depend on a feature vector $\mathbf{x}$ that may be, for example, composed of $\boldsymbol{\xi}$ and $\mathbf{u}$. The subscript refers to the timestep and dt is the duration of a timestep.

III. Model Learning Techniques

In this paper, we compare two common methods for probabilistic model learning: *(i)* Gaussian Process (GP) regression, which can learn nonlinear functions with additive Gaussian noise; and *(ii)* a particular form of Bayesian linear regression (BLR), which can learn functions that have additive Gaussian noise and are linear in a set of model parameters. In this section, we describe both methods and give a simple example illustrating a key difference between these methods that is relevant for stochastic control in changing conditions.

For both methods, our goal is to model $\mathbf{g}_k(\mathbf{x}_k)$ given measurements of $\mathbf{x}_k$, $\boldsymbol{\xi}_k$, and $\mathbf{u}_k$. We consider each dimension of $\mathbf{g}_k(\mathbf{x}_k)$ separately. For this section, we will refer to a single dimension of $\mathbf{g}_k(\cdot)$ as $g(\cdot)$. For a given pair of $\mathbf{x}_k$ and $\boldsymbol{\xi}_k$, the corresponding sample from $g(\mathbf{x}_k)$, denoted as g_k, may be calculated as $g_k = (\xi_{k+1} - g^0(\boldsymbol{\xi}_k, \mathbf{u}_k))/dt$, where ξ_{k+1} and $g^0(\cdot)$ are the relevant dimensions of $\boldsymbol{\xi}_{k+1}$ and $\mathbf{g}^0(\cdot)$ respectively.

A. Gaussian Process Regression

GPs are a common choice for modelling unknown functions such as $\mathbf{g}_k(\cdot)$ [1], [3], [13]. They provide an estimate of model uncertainty, can approximate nonlinear functions, and the parameters can be determined from data. Since there are many good references on GPs, e.g. [14], here we provide only a high-level summary. A GP models functions of the form

$$g(\mathbf{x}) = \mu(\mathbf{x}) + \eta, \tag{3}$$

where $\eta \sim \mathcal{N}(0, \sigma^2)$ and $\mu(\cdot)$ is a potentially nonlinear mean function. A GP is a distribution over functions given past data $\mathcal{D} = \{\mathbf{x}_i, g_i\}_{i=1}^n$, a kernel, and kernel hyperparameters. For this formulation, all points are weighted equally. The posterior distribution is characterized by a mean and variance, which can be queried at any point $\mathbf{x}_*$ using

$$\mu(\mathbf{x}_*) = \mathbf{k}^T(\mathbf{x}_*)\mathbf{K}^{-1}\mathbf{g}, \tag{4}$$

$$\sigma^2(\mathbf{x}_*) = \kappa(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}^T(\mathbf{x}_*)\mathbf{K}^{-1}\mathbf{k}(\mathbf{x}_*), \tag{5}$$

where $\mathbf{g}$ is a column vector of g_i, the covariance matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ has entries $\mathbf{K}_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$, where $\kappa(\cdot)$ is the kernel function. The column vector $\mathbf{k}(\mathbf{x}_*) = [\kappa(\mathbf{x}_*, \mathbf{x}_1), ..., \kappa(\mathbf{x}_*, \mathbf{x}_n)]^T$ contains the covariances between the new test point $\mathbf{x}_*$ and the observed data points in $\mathcal{D}$. For this work, we use the squared exponential kernel,

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2}||\mathbf{x}_i - \mathbf{x}_j||^2_{\mathbf{M}^{-2}}\right) + \sigma_\eta^2 \delta_{ij} \tag{6}$$

with δ_{ij} being the Kronecker delta, because of its success in modelling robot dynamics [3], [15]–[17]. The hyperparameters are the diagonal matrix $\mathbf{M}$ with so-called length-scales on the diagonal, which are inversely related to the importance of each element of $\mathbf{x}$, the process noise variance σ_f^2, which is the variance of the prior family of functions represented by the GP, and the measurement noise variance σ_η^2.

Although it is assumed that the underlying function is deterministic and that we receive noisy observations with a constant noise variance σ_η^2, the predicted variance (5) depends on $\mathbf{x}_*$ because uncertainty about the underlying function depends on the distribution of training data.

As training data is added to a particular GP, uncertainty is reduced and the posterior distribution of the GP specializes to a particular family of functions that represents the system dynamics in a particular operating condition.

B. Bayesian Linear Regression

Alternatively to GPs, if the function has additive Gaussian noise and can be factored such that it is linear in a set of model parameters, we can use Bayesian Linear Regression (BLR). In this section, we present weighted BLR (wBLR) where each data point can be assigned a weight indicating its importance. The weight can be used to vary its contribution

to the current regression. These weights are determined in a separate step and are useful for tasks such as long-term learning where some past experiences may be more relevant than others [12]. We will discuss how to compute these weights in Sec. IV-B.

Suppose we are given a *weighted* dataset $\mathcal{D}^l = \{\mathbf{x}_i,\ g_i,\ l_i\}_{i=1}^n$ with scalar weights $l_i \in [0, 1]$ that determine the importance of each data point. If $l_i = 0$, the point has no influence on the regression, and if $l_i = 1$, the point is fully included. In a simple scenario, all weights can be set to 1, in which case we recover regular BLR. We assume that the dynamics of interest depend on a vector of model parameters $\mathbf{w}$ and are of the form

$$g(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + \eta, \tag{7}$$

where $\eta \sim \mathcal{N}(0,\ \sigma^2)$. The goal of wBLR is to determine the distribution of both $\mathbf{w}$ and σ^2 given $\mathcal{D}^l$.

The likelihood of the dataset $\mathcal{D}^l$ assuming all measurements are independent is

$$p(\mathbf{g} \mid \mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{i=1}^{n} \mathcal{N}(g_i \mid \mathbf{x}_i^T\mathbf{w},\ \sigma^2)^{l_i}, \tag{8}$$

where $\mathbf{g}$ is a column vector of g_i, and rows of $\mathbf{X}$ are $\mathbf{x}_i$. Expanding the above, we get

$$\begin{aligned} &p(\mathbf{g} \mid \mathbf{X}, \mathbf{w}, \sigma^2) \\ &= \prod_{i=1}^{n} \left[(2\pi)^{-\frac{1}{2}} (\sigma^2)^{-\frac{1}{2}} \exp\left(-\frac{1}{2\sigma^2}(g_i - \mathbf{x}_i^T\mathbf{w})^2 \right) \right]^{l_i} \quad (9) \\ &= (2\pi)^{-\frac{tr(\mathbf{L})}{2}} (\sigma^2)^{-\frac{tr(\mathbf{L})}{2}} \exp\left(-\frac{1}{2\sigma^2} ||\mathbf{g} - \mathbf{X}\mathbf{w}||^2_{\mathbf{L}} \right), \quad (10) \end{aligned}$$

where $\mathbf{L}$ is a diagonal matrix with diagonal elements l_i. Now we factor the likelihood to get it in terms of the parameters of the distribution rather than the data. Ignoring constants (which do not depend on $\mathbf{w}$ and σ),

$$\begin{aligned} &p(\mathbf{g} \mid \mathbf{X}, \mathbf{w}, \sigma^2) \\ &\propto (\sigma^2)^{\frac{-tr(\mathbf{L})}{2}} \exp\left(\frac{-1}{2\sigma^2} (||\mathbf{g}||^2_{\mathbf{L}} - 2\mathbf{g}^T\mathbf{L}\mathbf{X}\mathbf{w} + ||\mathbf{X}\mathbf{w}||^2_{\mathbf{L}}) \right) \\ &\propto (\sigma^2)^{\frac{-tr(\mathbf{L})}{2}} \exp\left(\frac{-1}{2\sigma^2} ||\mathbf{w} - \bar{\mathbf{w}}||^2_{\mathbf{\Sigma}^{-1}} \right) \\ &\qquad \exp\left(\frac{-1}{2\sigma^2} (||\mathbf{g}||^2_{\mathbf{L}} - ||\bar{\mathbf{w}}||^2_{\mathbf{\Sigma}^{-1}}) \right), \quad (11) \end{aligned}$$

where $\bar{\mathbf{w}}^T = \mathbf{g}^T\mathbf{L}\mathbf{X}(\mathbf{X}^T\mathbf{L}\mathbf{X})^{-1}$ and $\mathbf{\Sigma}^{-1} = (\mathbf{X}^T\mathbf{L}\mathbf{X})$. This is of the form of a Normal Inverse Gamma (NIG) distribution,

$$\begin{aligned} p(\mathbf{g} \mid \mathbf{X}, \mathbf{w}, \sigma^2) = &\mathcal{N}(\mathbf{w} \mid \bar{\mathbf{w}},\ \sigma^2\mathbf{\Sigma}) \\ &IG(\sigma^2 \mid\ tr(\mathbf{L})/2,\ (||\mathbf{g}||^2_{\mathbf{L}} - ||\bar{\mathbf{w}}||^2_{\mathbf{\Sigma}^{-1}})/2), \end{aligned} \tag{12}$$

where $IG(\cdot)$ is the Inverse Gamma distribution. We can then choose an NIG distribution as a conjugate prior [18]. The prior represents our belief about the distribution of $\mathbf{w}$ and σ^2 before observing the data $\mathcal{D}^l$. The posterior joint distribution for $\mathbf{w}$ and σ^2 is then

$$\begin{aligned} p(\mathbf{w}, \sigma^2 \mid \mathcal{D}^l) &= NIG(\mathbf{w}, \sigma^2 \mid \mathbf{w}_N, \mathbf{V}_N, a_N, b_N) \quad (13) \\ &\triangleq \mathcal{N}(\mathbf{w} \mid \mathbf{w}_N, \sigma^2\mathbf{V}_N) IG(\sigma^2 \mid a_N, b_N), \quad (14) \end{aligned}$$

where

$$\mathbf{V}_N = (\mathbf{V}_0^{-1} + \mathbf{X}^T\mathbf{L}\mathbf{X})^{-1}, \tag{15}$$

$$\mathbf{w}_N = \mathbf{V}_N(\mathbf{V}_0^{-1}\mathbf{w}_0 + \mathbf{X}^T\mathbf{L}\mathbf{g}), \tag{16}$$

$$a_N = a_0 + tr(\mathbf{L})/2, \tag{17}$$

$$b_N = b_0 + \frac{1}{2}(\mathbf{w}_0^T\mathbf{V}_0^{-1}\mathbf{w}_0 + \mathbf{g}^T\mathbf{L}\mathbf{g} - \mathbf{w}_N^T\mathbf{V}_N^{-1}\mathbf{w}_N). \tag{18}$$

The subscripts 0 and N indicate parameters of the prior and posterior, respectively. The prior parameters (and their posterior equivalents) have the following interpretation: $\mathbf{w}_0$ is the prior mean for $\mathbf{w}$; a_0 proportional to the strength of the prior; $b_0 = \sigma_0^2 a_0$, where σ_0^2 is the prior estimate of σ^2; and $\mathbf{V}_0 = 1/\sigma_0^2\mathbf{\Sigma}_0^{\mathbf{ww}}$ where $\mathbf{\Sigma}_0^{\mathbf{ww}}$ is the covariance matrix of $\mathbf{w}_0$. For example, if we have prior knowledge suggesting that each element of $\mathbf{w}$ is independent and has a mean value of zero and variance τ_0^2, then $\mathbf{V}_0 = \tau_0^2/\sigma_0^2\mathbf{I}$. These parameters can also be identified from data using (15)-(18) and a non-informative prior, i.e. $\mathbf{w}_0 = \mathbf{0}$, $\tau_0^2 = \infty$, $b_0 = 0$, and $a_0 = 0$, and σ_0^2 large. The resulting posterior parameters can then be used to set a *new* prior. The prior is initialized once this way at the beginning of each run.

Following [18], the posterior marginals are then

$$p(\sigma^2 \mid \mathcal{D}^l) = IG(\sigma^2 \mid a_N,\ b_N), \tag{19}$$

$$p(\mathbf{w} \mid \mathcal{D}^l) = \mathcal{T}(\mathbf{w} \mid \mathbf{w}_N,\ \frac{b_N}{a_N}\mathbf{V}_N,\ 2a_N), \tag{20}$$

where $\mathcal{T}$ is a Student t distribution. The Student t distribution arises because of marginalizing out σ^2 and rapidly converges to a Gaussian distribution for $2a_N \gg 5$ [18], which is useful for using this method with common tools for robotics which often assume Gaussian probability distributions.

Recursive Updates: When dealing with streaming data such as the data generated by a robot driving, it is useful to continually update the model in order to adapt quickly to new scenarios. To do this while ensuring the model stays flexible enough to adapt to sudden changes, we recursively update the prior parameters while keeping the strength of the prior fixed at a pre-determined value n_0. The value of n_0 determines how many effective data points we attribute to the prior. A large value for n_0 results in smoother estimates for the $\mathbf{w}$ and σ^2 while a smaller value for n_0 allows them to vary more quickly. If we start with fewer than n_0 points in the prior, e.g. $a_0 < n_0/2$, we update the prior using (15)-(18) with the weight for the new point set to one, and set the posterior parameters to the prior for the next timestep. Once a_0 reaches $n_0/2$, we use (15)-(18) with the weight for the new point set to one and then use the following re-weighting

to keep n_0 constant:

$$\mathbf{V}_{0^*} = \frac{n_0+1}{n_0}\mathbf{V}_N, \qquad \mathbf{w}_{0^*} = \mathbf{w}_N, \tag{21}$$

$$a_{0^*} = \frac{n_0}{n_0+1}a_N, \qquad b_{0^*} = \frac{n_0}{n_0+1}b_N. \tag{22}$$

The parameters $(\cdot)_{0^*}$ are the re-weighted parameters which become the new prior. This is equivalent to assigning the prior *and* the new point a weight of $n_0/(n_0+1)$ and carrying out a weighted update using (15)-(18).

Compared to GPs, this gives us more control on how fast the model adapts. For a GP, a new point must either displace an existing one if the model has fixed size or increase the model size, which increases the computational cost of the model and will make it less flexible over time as more points are added. For wBLR, the influence of old data decreases after each re-weighting. The rate at which this happens depends on n_0, which is a parameter of our choosing and does not affect the computational cost of the model.

C. Computational Efficiency

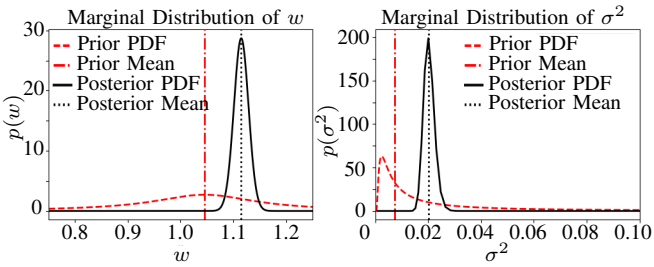
There are three important operations for each model: *(i)* fitting the model to the data, *(ii)* evaluating the mean at a test point and *(iii)* evaluating the variance at a test point. Fitting a model can be done in parallel to the control loop to mitigate some of the computational cost of this step [2]. However, querying the mean and variance are often part of computing the control, which makes these operations time-critical. This is especially true for MPC, which queries the model several times at each timestep.

The time it takes to fit a GP to N data points, query the variance, and query the mean scales with $O(N^3)$, $O(N^2)$, and $O(N)$, respectively. This is because of the computation of the matrix inverse $\mathbf{K}^{-1}$, the matrix product in (5), and the dot product in (4). For wBLR, the same operations scale with $O(N)$, $O(1)$, and $O(1)$ because of the dot products in (15)-(18) and because the output equations are parametric.

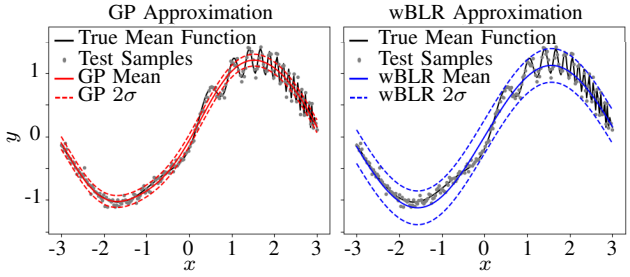
D. Simple Example

Safe control algorithms require an accurate estimate of the model uncertainty to guarantee constraint satisfaction even if the mean prediction from the model is inaccurate. One key difference between wBLR and GP regression (for fixed GP hyperparameters) is that the uncertainty estimate from a GP depends only on the similarity between the query point and training points in the *input* space as measured by a kernel (see (5)). In contrast, the wBLR parameters can be updated efficiently online using (15)-(18) which allows the posterior estimate to reflect variations in noise more readily.

Figure 1b shows a simple example where the underlying function has a component that increases in frequency as the input variable increases. Training samples are gathered from $x \in [-3, 0]$ and used to determine the GP hyperparameters and the wBLR prior. For the GP, we use a squared exponential kernel (6) and determine the hyperparameters using maximum likelihood. For wBLR, we use a non-informative prior and set $l_i = 1/n$ so that $a_0 = 1/2$ (the effective number of data points in the prior is 1). For a robot, this represents gathering training data from a set of test scenarios. The test data is gathered from $x \in [-3, 3]$, which represents deploying the robot in a partially novel scenario. Once both models are fit to this new data, which involves computing $\mathbf{K}^{-1}$ and $\mathbf{K}^{-1}\mathbf{g}$ based on the new points for the GP and computing the posterior parameters for wBLR using (15)-(18), Fig. 1b shows how the GP uncertainty estimate is no longer valid while the wBLR model is able to adapt the uncertainty estimate (see Fig. 1a) making it more appropriate for this new data. This property is important for safe learning-based controllers deployed in novel scenarios.



(a) wBLR estimates for the distribution of the weights and the output variance. The prior is set using 100 samples from the function depicted in Fig. 1b from $x \in [-3, 0]$ and the posterior is set using 200 samples from $x \in [-3, 3]$. The wBLR posterior estimate of the variance increases to reflect the high frequency component of the function that is more pronounced for $x > 0$.



(b) Predictive distribution for the GP and wBLR models after fitting the posterior to $x \in [-3, 3]$. The fact that the GP hyperparameters are fixed, which is a common assumption due to computational constrains, means it cannot adapt to the high frequency component for $x > 0$, while the wBLR model increases the output variance resulting in more accurate uncertainty bounds. While the mean estimates for the two methods are very similar, the model uncertainty estimate for wBLR is more realistic which is important for safe learning algorithms.

Fig. 1: Simple example illustrating one difference between wBLR and GPs for the function $y = 1.2\sin(x) + 0.2\sin(5\exp(x)) + \eta$, where $\eta \sim \mathcal{N}(0, 0.05^2)$. For wBLR, it is assumed that the function is of the form $y = w\sin(x) + \eta$ where w is the model parameter and $\sin(x)$ is the feature.

IV. Model Learning for Repetitive Path Following and Predictive Control

We consider a repetitive path following task with a stochastic MPC (presented in [12]) where data from each run is stored and indexed by location along the path as depicted in Fig. 2. Stochastic MPC computes a sequence of controls that minimises a given cost function over a prediction horizon subject to state and input constraints given a model for the robot dynamics and the current state of the robot. At each timestep, the first input value of the computed, optimal sequence is applied to the system. This is repeated at the each time-step.

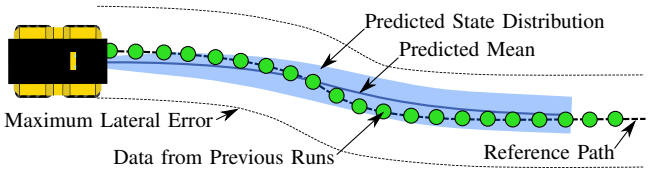


Fig. 2: This figure shows the robot and the predicted state distribution given the sequence of inputs from MPC overlaid on the reference path with maximum path tracking constraints (state constraints). Data from previous runs is indexed by location along the path and indicated by the green circles. We are interested in constructing a model for the vehicles dynamics over the section of the path which is relevant for the MPC based on the chosen prediction horizon (shaded blue region). To construct this model, we consider data from previous runs over that section of the path.

A. Local Models

Since the controller only makes use of the dynamics of the vehicle over the upcoming section of the path, we only have to model the dynamics of the vehicle relevant for the upcoming manoeuvre. The advantage for the GP is that we need fewer points to model the dynamics over this restricted region of the state space which is more computationally efficient. The advantage for wBLR is that the relatively simple parametric model is more likely to be a good approximation for the dynamics over this restricted region.

B. Data Management

For the GP, we only use data from the upcoming section of the path (and not the current run) because the number of points in the GP is limited and we only need the dynamics for the upcoming manoeuvre. Following [2], to update the points in the GP, we randomly take n_p points from the previous run over the upcoming section of the path, combine this with the n_{gp} points in the GP, and randomly select n_{gp} points from this combined set to construct the updated model. In this way, the number of points in the GP remains constant and the GP 'forgets' old experiences over time.

For wBLR, we recursively update the prior using data from the current run at each timestep (Sec. III-B) and compute the posterior using data from the upcoming section of the path weighted according to how similar the dynamics in each previous run were to the current run (15)-(18). See [12] for details on determining the weights. We use data from all previous runs because the update is efficient and the time to evaluate the model does not depend on the number of data points used to construct it. The posterior update (based on data from previous runs) is considered to be location specific and therefore discarded after computing the control; that is, the recursively updated prior becomes the prior for the next timestep.

We use these local models to predict trajectories and their associated uncertainty at each timestep given a sequence of inputs from MPC. Both models are updated at every timestep

C. Uncertainty Propagation

We assume a Gaussian belief over the state at each time step and a nonlinear model for the robot dynamics. This allows us to use the Extended Kalman Filter (EKF) prediction equations to propagate our belief over the full state into the future given a series of inputs [19].

For a wBLR-based model, we include uncertainty in the full state $\mathbf{z} = [\mathbf{s}^T, \boldsymbol{\xi}^T]^T$, the actuator model parameters $\mathbf{w}$, and the actuator model output $\boldsymbol{\eta}$. Let $\mathbf{h}(\cdot)$ be the combined dynamics model (1) and (2) and $\mathbf{A}$ be the Jacobian of $\mathbf{h}(\cdot)$ with respect to the stacked full state and parameters evaluated at the mean state at time k, $\mathbf{A} = [\mathbf{A_z}, \mathbf{A_w}]$. The mean $\bar{\mathbf{z}}_k$ and covariance $\boldsymbol{\Sigma}_k^{\mathbf{zz}}$ can be updated using:

$$\bar{\mathbf{z}}_{k+1} = \mathbf{h}(\bar{\mathbf{z}}_k, \mathbf{u}_k), \tag{23}$$

$$\boldsymbol{\Sigma}_{k+1}^{\mathbf{zz}} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}_k, \tag{24}$$

$$\mathbf{P}_k = \begin{bmatrix} \boldsymbol{\Sigma}_k^{\mathbf{zz}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_k^{\mathbf{ww}} \end{bmatrix}, \tag{25}$$

where $\boldsymbol{\Sigma}_k^{\mathbf{ww}}$ is a block-diagonal matrix containing the model weight covariance matrix from (20) for each dimension of $\mathbf{g}_k(\cdot)$, $\mathbf{Q}_k$ is the process noise covariance, and $\mathbf{u}_k$ comes from MPC. The only non-zero components in $\mathbf{Q}_k$ are the diagonal elements corresponding to uncertainty in the output of the actuators for which we use the posterior mean of the variance from (19). In this framework, we can include uncertainty in the evolution of the model parameters $\mathbf{w}$ by modelling their dynamics as a random walk. In this work, we consider them to be fixed at the posterior estimate over the lookahead horizon.

The prediction equations for the GP-based model are the same but without the additional elements for $\mathbf{w}$ and using (5) for the non-zero elements of $\mathbf{Q}_k$.

D. Ancillary Controller Design

If we use the dynamics model $\mathbf{h}(\cdot)$ without modification, the method presented in the previous section estimates the uncertainty in the predicted trajectories as if the control sequence was applied open loop. However, this is conservative since it neglects the fact that the controller can take corrective actions at each timestep. One common approach to account for this is to include an ancillary controller in the predictive model that is designed to steer the system towards the predicted mean, denoted by $\bar{(\cdot)}$. This limits the growth of uncertainty around the mean. A common simplifying assumption is that the prior model is linear (or linearised about an operating point) and that the unknown dynamics are bounded [19]–[22]. Since we assume the dynamics of $\mathbf{s}$ are known and nonlinear, we use an ancillary controller to control the growth of the uncertain, learned dynamics of $\boldsymbol{\xi}$. Not only does this simplify the design of the ancillary controllers, but it allows us to reduce the uncertainty in $\boldsymbol{\xi}$ which in turn keeps the uncertainty in $\mathbf{s}$ low over the length of a typical MPC prediction horizon. The dynamics of the actuator state with the added ancillary controller $\boldsymbol{\pi}(\cdot)$ become

$$\boldsymbol{\xi}_{k+1} = \mathbf{g}^0(\boldsymbol{\xi}_k, \mathbf{u}_k) + dt\big(\mathbf{g}_k(\mathbf{x}_k) + \boldsymbol{\pi}(\boldsymbol{\xi}_k, \mathbf{s}_k, \bar{\boldsymbol{\xi}}_k, \bar{\mathbf{s}}_k)\big). \tag{26}$$

The ancillary controller is never used directly and is a design parameter used to control the growth of predicted uncertainty. There are guidelines on how to choose it [19]–[22]. In general, the more control action is allocated to reduce uncertainty in the predicted state, the less control action is

Fig. 3: This figure shows the 900 kg Clearpath Grizzly on the muddy test track used for the experiments in the results section. The controller receives estimates of its pose and velocity from a vision system which relies solely on a stereo camera (c.f. [23]).

available for MPC to steer the mean of the system along the path. In the following section, we will show one possible design for $\boldsymbol{\pi}(\cdot)$ for a ground vehicle.

V. APPLICATION TO A GROUND ROBOT

In this section, we show an example of how to apply the two probabilistic modelling techniques, GPs and wBLR, in combination with stochastic MPC to a ground robot, the Clearpath Grizzly depicted in Fig. 3.

A. Robot Model

Let $\mathbf{s} = [x, y, \theta]^T$, the 2D position and heading of the robot, $\boldsymbol{\xi} = [v, \omega]^T$, the speed and turn rate of the robot, and $\mathbf{u} = [v^{cmd}, \omega^{cmd}]^T$, the commanded speed and turn rate of the robot. We assume that the dynamics of $\mathbf{s}$ are well approximated by a unicycle

$$\underbrace{\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix}}_{\mathbf{s}_{k+1}} = \underbrace{\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}}_{\mathbf{s}_k} + dt \underbrace{\begin{bmatrix} v_k \cos\theta_k \\ v_k \sin\theta_k \\ \omega_k \end{bmatrix}}_{\mathbf{f}(\cdot)}, \tag{27}$$

which is of the form (1). For wBLR, we will model the dynamics of $\boldsymbol{\xi}$ as

$$\underbrace{\begin{bmatrix} v_{k+1} \\ \omega_{k+1} \end{bmatrix}}_{\boldsymbol{\xi}_{k+1}} = \underbrace{\begin{bmatrix} v_k \\ \omega_k \end{bmatrix}}_{\mathbf{g}^0(\cdot)} + dt \underbrace{\begin{bmatrix} [v_k^{cmd}, v_k]\mathbf{w}_k^v + \eta_k^v \\ [\omega_k^{cmd}, \omega_k]\mathbf{w}_k^\omega + \eta_k^\omega \end{bmatrix}}_{\mathbf{g}_k(\cdot)}, \tag{28}$$

which is of the form (2). For the model using GPs, we model the speed and turn rate dynamics as

$$\underbrace{\begin{bmatrix} v_{k+1} \\ \omega_{k+1} \end{bmatrix}}_{\boldsymbol{\xi}_{k+1}} = \underbrace{\begin{bmatrix} v_k^{cmd} \\ \omega_k^{cmd} \end{bmatrix}}_{\mathbf{g}^0(\cdot)} + dt \underbrace{\begin{bmatrix} \mu_k^v(v_k, v_k^{cmd}) + \eta_k^v \\ \mu_k^\omega(\omega_k, \omega_k^{cmd}) + \eta_k^\omega \end{bmatrix}}_{\mathbf{g}_k(\cdot)}. \tag{29}$$

We use different models for the GP and wBLR so that the speed and turn rate for the GP-based model depends on the control inputs before data is added to the GP. If it did not, MPC would not be able to drive the vehicle for the first run. For wBLR, we keep $\mathbf{g}^0(\cdot) = \boldsymbol{\xi}_k$ which is a more natural choice for modelling dynamic systems. It should be noted that the wBLR prior is equivalent to the GP prior for a certain choice of $\mathbf{w}$. In both cases, $\mathbf{x}$ is composed of $\boldsymbol{\xi}$ and $\mathbf{u}$.

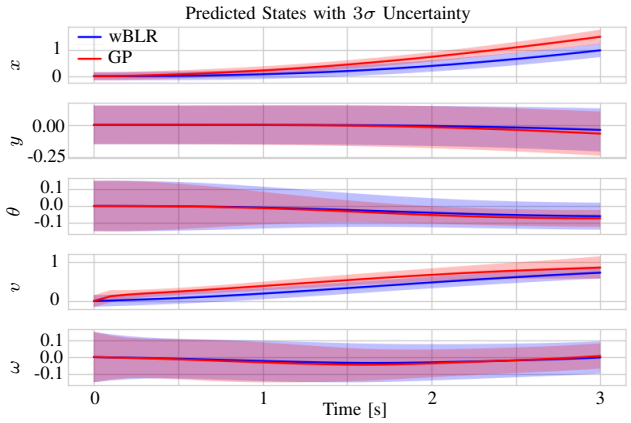


Fig. 4: This figure shows the predicted state distribution from both the GP-based (red) and wBLR-based (blue) models using the proposed ancillary controller. The shaded regions are the predicted 3σ bounds. The proposed ancillary controller is able to keep the 3σ uncertainty in y, which is roughly equivalent to lateral uncertainty, at about 25 cm at the end of the prediction horizon for both models, which is well below the maximum lateral uncertainty used in our experiments.

B. Ancillary Controller Design

We use a linear state feedback controller as an ancillary controller for the learned dynamics. The main benefit of this is that predicted uncertainty in the error the controller reacts to can be straightforwardly used to tighten the constraints on the control inputs using the method described in [19]. Here, we present an ancillary controller design for both wBLR and GP-based models and show simulation results demonstrating the effectiveness of this approach.

We use the following ancillary controllers to reduce the predicted speed and heading uncertainty and, as a consequence, position uncertainty, which is linked to the maximum path tracking error constraint. The turn rate and speed dynamics with the ancillary controllers are

$$\begin{bmatrix} v_{k+1} \\ \omega_{k+1} \end{bmatrix} = \mathbf{g}_0(\cdot) + dt \underbrace{\begin{bmatrix} g_k^v(v_k, v_k^{cmd}) \\ g_k^\omega(\omega_k, \omega_k^{cmd}) \end{bmatrix}}_{\mathbf{g}_k(\cdot)} + \underbrace{\begin{bmatrix} K_v(v_k - \bar{v}_k) \\ K_\theta(\theta_k - \bar{\theta}_k) \end{bmatrix}}_{\boldsymbol{\pi}(\cdot)}, \tag{30}$$

where $g_k^v(v_k, v_k^{cmd})$ and $g_k^\omega(\omega_k, \omega_k^{cmd})$ are the respective wBLR or GP-based models and $\mathbf{g}_0(\cdot)$ is the corresponding prior, either (28) or (29). The ancillary controller gains were chosen manually because it was easy to find values that worked well in practice.

Figure 4 shows the predicted uncertainty using the GP and wBLR models with the proposed ancillary controller. Each learning model was trained using data from the same experiment and K_v and K_θ were set to -5. Here, we can see that the proposed method is able to keep the uncertainty at a reasonable level for the duration of a typical MPC prediction horizon for both GP- and wBLR-based models.

VI. RESULTS

The purpose of this section is to compare the performance of the stochastic MPC when using the GP- and wBLR-based

models. To test both methods in a challenging environment where the robot dynamics change, we drove the vehicle around a 43 m muddy path (depicted in Fig. 3) with varied target speed. As the target speed was increased, the vehicle incurred more slip which noticeably changed the dynamics. The speed was increased gradually to ensure that the most relevant data was from the last run for the GP.

First, we compare the tracking performance at various target speeds. Next, we compare the model performance to link the mean and uncertainty estimates from each model to the resulting closed-loop performance.

A. Experimental Setup

Our experimental platform was the Clearpath Grizzly depicted in Fig. 3. The controller was the stochastic MPC described in [12]. The controller parameters were kept constant and only the learning model and target speed were changed during the experiment. Two sets of GP hyperparameters were determined from a dataset where the wBLR-based controller drove the vehicle on the muddy test track: one with the vehicle driving with a target speed of 1.0 m/s (GP 1); and the other from the vehicle driving with a target speed varied between 1.0 and 2.5 m/s in increments of 0.5 m/s (GP 2). The hyperparameters were chosen to maximize the likelihood of a randomly chosen validation set. A constant prior was used to initialize wBLR that was identified from another dataset. This prior only matters for the first few seconds because it is constantly updated using data from the current run.

Our algorithm was implemented in C++ on an Intel i7 2.70 GHz 8 core processor with 16 GB of RAM. The solver used in MPC was CPLEX [24]. The controller runs at 10 Hz with a look-ahead of 2 seconds, which is discretised with 20 points. The local GP was constructed using $n_{gp} = 50$ past data points and $n_p = 5$. For wBLR, n_0 was set to 200. The maximum lateral error was set to 1.0 m and K_v and K_θ were set to -5. We rely on a vision-based system for localization [23], which runs on the same laptop.

B. Closed-Loop Performance Comparison

Figure 5 shows that both models enabled the controller to perform well at low speed. However, for the GP-based models, the tracking error increased quickly with the target speed. GP 1 failed during run 6 at a target speed of 1.5 m/s and GP 2, trained on data from higher speeds, failed during run 7 with a target speed of 2.0 m/s. When the controller used the wBLR-based model, the tracking error increased with the desired speed, but the vehicle was able to traverse the path safely and reliably for all target speeds. In the next section, we analyse the prediction accuracy of the two models and show how it is linked to the difference in performance.

C. Multi-Step Prediction Performance

In this section, we compare the prediction performance of the best GP-based model (GP 2) and the wBLR-based model offline. Our performance metrics are the Multi-step RMSE (M-RMSE), which is the RMS prediction error over the MPC look-ahead horizon, and the Multi-step RMSZ (M-RMSZ),

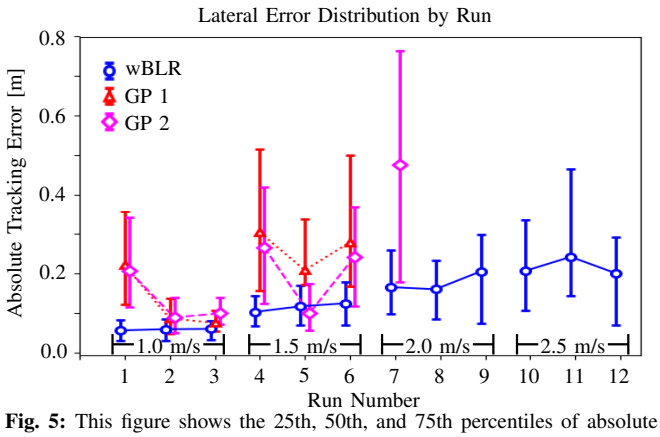


Fig. 5: This figure shows the 25th, 50th, and 75th percentiles of absolute lateral tracking error when the controller used each model as the target speed was increased. When the controller used the wBLR-based model, it was able to drive the vehicle at a significantly higher target speed while maintaining low path tracking error. While the controller performed well at low speed using the GP-based models, the performance quickly degraded as the target speed was increased leading to failures on runs 6 for GP 1 and 7 for GP 2.

which is the RMS Z-score over the MPC look-ahead horizon [2]. Ideally, the M-RMSE should be low and the M-RMSZ should be close to 1.

Figure 6 shows that the predictive performance of both models is comparable when the vehicle drives up to 2.0 m/s. The GP takes one run to adapt because it only uses data from previous runs. This slow adaptation is likely what caused the failure for GP 2 at 2.0 m/s. When the target speed is increased to 2.5 m/s, the wBLR-based model clearly outperforms the GP-based models. The M-RMSE is higher than previous runs for wBLR as well, but the M-RMSZ is around 1.0 indicating that the model uncertainty estimate is still realistic. This indicates that while the wBLR-based model appears to make stronger assumptions on the form of the model, being able to adapt all model parameters including the uncertainty online and use data from the current run enables this model to adapt to novel scenarios better than the GP-based models.

D. Speed Comparison

Figure 7 shows the *actual* speed of the vehicle when the controller used the GP 2- and wBLR-based models. The average speed achieved by the better of the GP-based controllers (GP 2) during runs 4-6 was 1.00 m/s. The wBLR-based controller was slightly slower during runs 4-6 at 0.92 m/s, but increased this to 1.08 and 1.09 m/s during runs 7-9 and 10-12 respectively (9 % faster than the GP). The fact that the speed did not increase much between runs 7-9 and 10-12 indicates that the controller is making use of the uncertainty estimate to limit the maximum speed of the vehicle in order to maintain safety. This shows that the wBLR-based model enabled the controller to drive the vehicle *safely* and *reliably* at equal or higher speeds than the GP-based models in these challenging conditions.

VII. CONCLUSION

There are many choices for probabilistic models for robot dynamics. Two common choices are GP regression and various forms of local linear regression. There is a large body

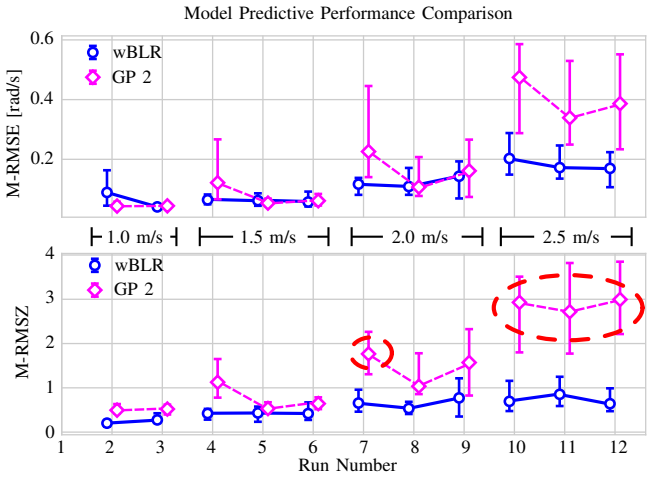


Fig. 6: This figure shows the 25th, 50th, and 75th percentiles of M-RMSE and M-RMSZ for the GP and wBLR-based models at varied target speed. These values are calculated offline based on the data from the wBLR-based controller driving the vehicle. This shows that while both methods perform well at low speed, the wBLR-based model continues to perform well at high speed indicating better generalization to new conditions. The GP-based model produces higher error and more overconfident error estimates (higher M-RMSZ indicated by the red circles) when the speed changes and at higher speeds because it does not have a fast adaptation term or adjust the estimate of σ^2 like wBLR. This likely contributed to the controller using the GP-based model to fail during run 7 (see Fig. 5).

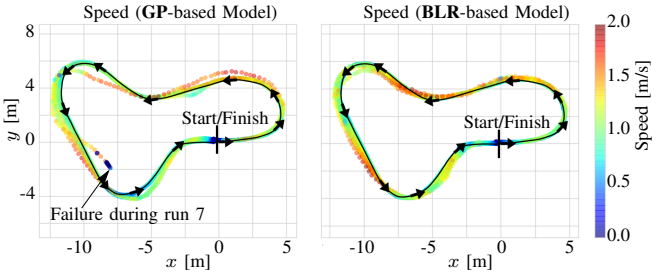


Fig. 7: A top-down view of the path taken by the vehicle coloured by speed for each run over our 43 m test track when using GP 2 (left) and wBLR (right) to model the robot dynamics. This shows that the wBLR-based method was able to drive the vehicle faster than the GP-based method consistently as well as maintaining low path-tracking error. The black line indicates the reference and the arrows along the path indicate the direction of travel.

of work that shows that GP regression is useful for modelling robot dynamics in relatively controlled environments. In this paper, we have shown that a wBLR-based approach can generalize better than GP regression in challenging and changing conditions. This allows the stochastic MPC to continue operating as conditions become more challenging and increasingly different from previously seen conditions. This enables safe control in such environments. We hope the reader will consider methods such as wBLR, which enables fast adaptation and online parameter updates, good candidates for modelling robot dynamics, especially when the robot is deployed in a wide range of operating conditions.

References

[1] L. Hewing, A. Liniger, and M. N. Zeilinger. Cautious NMPC with Gaussian Process Dynamics for Miniature Race Cars. In *Proc. of the European Control Conf.*, pages 1341–1348, 2018.

[2] C. D. McKinnon and A. P. Schoellig. Experience-Based Model Selection to Enable Long-Term, Safe Control for Repetitive Tasks Under Changing Conditions. In *Proc. of the Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2977–2984, 2018.

[3] C. Ostafew, A. P. Schoellig, and T. Barfoot. Robust Constrained Learning-based NMPC Enabling Reliable Mobile Robot Path Tracking. *Intl. Journal of Robotics Research (IJRR)*, 35(13):1547–1563, 2016.

[4] A. Akametalu, J. Fisac, J. Gillula, S. Kaynama, M. Zeilinger, and C. Tomlin. Reachability-based Safe Learning with Gaussian Processes. In *Proc. of the Conf. on Decision and Control (CDC)*, pages 1424–1431, 2014.

[5] L. Jamone, B. Damas, and J. Santos-Victor. Incremental Learning of Context-dependent Dynamic Internal Models for Robot Control. In *Intl. Symp. on Intelligent Control (ISIC)*, pages 1336–1341, 2014.

[6] J. Ting, A. D'Souza, S. Vijayakumar, and S. Schaal. A Bayesian Approach to Empirical Local Linearization for Robotics. In *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, pages 2860–2865, 2008.

[7] V. Desaraju, A. Spitzer, and N. Michael. Experience-driven Predictive Control with Robust Constraint Satisfaction under Time-Varying State Uncertainty. In *Proc. of the Robotics: Science and Systems Conf. (RSS)*, 2017.

[8] F. Meier, D. Kappler, N. Ratliff, and S. Schaal. Towards Robust Online Inverse Dynamics Learning. In *Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4034–4039, 2016.

[9] F. Meier and S. Schaal. Drifting Gaussian processes with varying neighborhood sizes for online model learning. In *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, pages 264–269, 2016.

[10] D. Nguyen-Tuong, M. Seeger, and J. Peters. Model Learning with Local Gaussian Process Regression. *Advanced Robotics*, 23(15):2015–2034, 2009.

[11] C. Xie, S. Patil, T. Moldovan, S. Levine, and P. Abbeel. Model-based Reinforcement Learning with Parametrized Physical Models and Optimism-Driven Exploration. In *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, pages 504–511, 2016.

[12] C. D. McKinnon and A. P. Schoellig. Learn Fast, Forget Slow: Safe Predictive Learning Control for Systems with Unknown and Changing Dynamics Performing Repetitive Tasks. *Robotics and Automation Letters*, 2019.

[13] C. D. McKinnon and A. P. Schoellig. Learning Multi-Modal Models for Robot Dynamics with a Mixture of Gaussian Process Experts. In *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, pages 322–328, 2017.

[14] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[15] C. Ostafew, A. P. Schoellig, and T. Barfoot. Learning-based Nonlinear Model Predictive Control to Improve Vision-Based Mobile Robot Path-tracking in Challenging Outdoor Environments. In *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, pages 4029–4036, 2014.

[16] C. Ostafew, A. P. Schoellig, and T. Barfoot. Conservative to Confident: Treating Uncertainty Robustly Within Learning-Based Control. In *Proc of the Intl. Conf. on Robotics and Automation (ICRA)*, pages 421–427, 2015.

[17] P. Bouffard, A. Aswani, and C. Tomlin. Learning-based Model Predictive Control on a Quadrotor: Onboard Implementation and Experimental Results. In *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, pages 279–284, 2012.

[18] K. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[19] L. Hewing and M. N. Zeilinger. Cautious Model Predictive Control Using Gaussian Process Regression. In *arXiv:1705.10702*, 2017.

[20] A. Aswani, H. Gonzalez, S. Sastry, and C. Tomlin. Provably Safe and Robust Learning-based Model Predictive Control. *Automatica*, 49(5):1216–1226, 2013.

[21] Y. Gao, A. Gray, H. Tseng, and F. Borrelli. A Tube-based Robust Nonlinear Predictive Control Approach to Semiautonomous Ground Vehicles. *Vehicle System Dynamics*, 52(6):802–823, 2014.

[22] A. Carvalho, Y. Gao, S. Lefevre, and F. Borrelli. Stochastic Predictive Control of Autonomous Vehicles in Uncertain Environments. In *Proc. of the Intl. Symp. on Advanced Vehicle Control*, pages 712–719, 2014.

[23] M. Paton, F. Pomerleau, K. MacTavish, C. Ostafew, and T. Barfoot. Expanding the Limits of Vision-based Localization for Long-term Route-following Autonomy. *Journal of Field Robotics (JFR)*, 34(1):98–122, 2017.

[24] IBM. IBM ILOG CPLEX Optimization Studio 12.7.1.