2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Learning-Based Nonlinear Model Predictive Control to Improve Vision-Based Mobile Robot Path-Tracking in Challenging Outdoor Environments

Chris J. Ostafew, Angela P. Schoellig, and Timothy D. Barfoot

***Abstract*— This paper presents a Learning-based Nonlinear Model Predictive Control (LB-NMPC) algorithm for an autonomous mobile robot to reduce path-tracking errors over repeated traverses along a reference path. The LB-NMPC algorithm uses a simple *a priori* vehicle model and a learned disturbance model. Disturbances are modelled as a Gaussian Process (GP) based on experience collected during previous traversals as a function of system state, input and other relevant variables. Modelling the disturbance as a GP enables interpolation and extrapolation of learned disturbances, a key feature of this algorithm. Localization for the controller is provided by an on-board, vision-based mapping and navigation system enabling operation in large-scale, GPS-denied environments. The paper presents experimental results including over 1.8 km of travel by a four-wheeled, 50 kg robot travelling through challenging terrain (including steep, uneven hills) and by a six-wheeled, 160 kg robot learning disturbances caused by unmodelled dynamics at speeds ranging from 0.35 m/s to 1.0 m/s. The speed is scheduled to balance trial time, path-tracking errors, and localization reliability based on previous experience. The results show that the system can start from a generic *a priori* vehicle model and subsequently learn to reduce vehicle- and trajectory-specific path-tracking errors based on experience.**

I. Introduction

In many mobile robot applications, it is adequate if not necessary, to explore and navigate the environment by creating and maintaining a network of paths analogous to migration routes or automobile roads [1]. The use and reuse of paths reduces the need for repeated application of exploratory and terrain-assessing software and also provides an opportunity for learning behavior. Learning control algorithms offer tools to acquire knowledge about the robot dynamics and environment in situ, acausally incorporate unknown effects, and improve performance over time. These advantages reduce the requirement for significant modelling *a priori* [2].

In this paper, we investigate a Learning-based Nonlinear Model Predictive Control (LB-NMPC) algorithm for a nonholonomic, mobile robot within the context of an on-board, real-time, Visual Teach and Repeat (VT&R) mapping and navigation system [3]. NMPC is a control framework in which the current control action is obtained by solving, at each sampling instant, a finite-horizon optimal control problem using the current state of the plant as the initial state [4, 5]. We apply NMPC to a system with a process model comprised of two components: (i) a unicycle model representing the kinematics of the robot, and (ii) a learned disturbance model representing both unmodelled robot dynamics and systematic environmental disturbances. We model disturbances as a Gaussian Process (GP) [7] based on observations gathered during previous path traversals as a function of system state, input and other relevant system variables. By modelling the disturbances as a GP, the algorithm is able to interpolate and extrapolate from previous experiences. This is a key enabling feature for practical, large-scale applications of mobile robots that may be required to travel at various speeds on many paths while managing unmodelled terrain and robot dynamics.

The authors are with the University of Toronto Institute for Aerospace Studies, Toronto, Ontario, M3H 5T6, Canada. Email: chris.ostafew@mail.utoronto.ca, schoellig@utias.utoronto.ca, and tim.barfoot@utoronto.ca.

Fig. 1. The proposed learning-based, path-tracking controller compensates for unmodelled environmental disturbances, such as the terrain slope and wheel slip affecting our 50 kg Husky A200 robot, enabling operation in challenging environments.

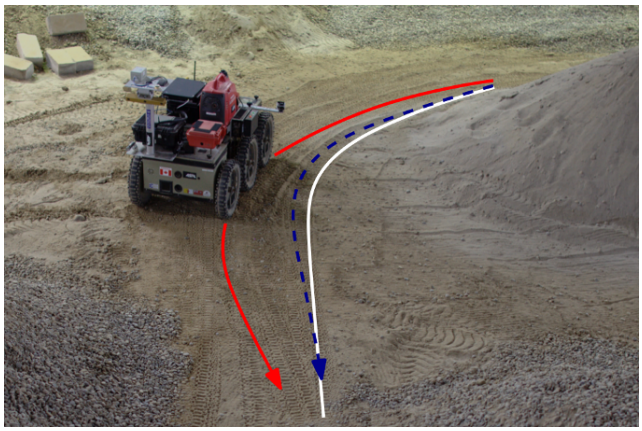

Fig. 2. The proposed algorithm learns unmodelled robot dynamics. Here we show two trajectories of a 150 kg ROC6 robot travelling at 1.0 m/s. The solid white line shows the desired trajectory (tire tracks), the red line shows a trajectory with learning disabled, while the dashed blue line shows a trajectory with learning enabled and reduced path-tracking errors.

We also investigate a speed scheduler that automatically increases speeds along the path based on previous tracking performance (i.e., tracking errors and localization quality) to achieve faster overall path completion. The scheduler balances exploration and exploitation: with the ability to learn a multivariate disturbance model, the system must choose between exploring states that may be beneficial, in our case driving faster, and exploiting states that recall previously learned disturbances and therefore have lower path-tracking errors. Effectively, the proposed speed scheduler increases the speed where the path-tracking errors can be kept small.

The key characteristics of this paper are: (i) a path-tracking, LB-NMPC algorithm based on a fixed, *a priori* known kinematic process model and a learned GP disturbance model, (ii) navigation based on vision only, (iii) an automated speed scheduler, which adapts speeds based on previous experience addressing the exploration vs. exploitation trade-off, and (iv) extensive outdoor experiments including over 1.8 km of travel by two significantly different robots in unmodelled terrain and at intentionally increasing velocities; the results show the algorithm's ability to interpolate and extrapolate from learned experiences. To our knowledge, this paper is the first application of LB-NMPC capable of interpolating and extrapolating from learned experiences on a self-contained autonomous mobile robot travelling in challenging outdoor environments.

II. Related Work

In the literature, there are several examples of MPC applied to mobile robots. Kühne et al. [8], Klančar and Škrjanc [9], and Xie and Fierro [10] demonstrate MPC on robots with nonholonomic constraints operating in indoor lab environments. All of these systems employ a simple kinematic robot model. Peters and Iagnemma [11] present simulation results for a fixed model of wheel slip on a car-like robot, while Burke [12] presents simulated MPC results with an adaptive value for wheel slip on a track-type robot. Howard et al. [13] demonstrate MPC on a large-scale, outdoor robot navigating intricate paths. Unlike these prior works, our LB-NMPC algorithm learns a general, nonlinear disturbance function representing both unmodelled environmental disturbances and robot dynamics.

LB-MPC has previously been proposed for other applications. Kocijan et al. [14] combine a GP model and NMPC for the control of a simulated pH neutralization process. They represent the full dynamics of the system by a GP model trained on 400 observations of the chemical system. NMPC is applied to control the system based on the offline-identified GP model (i.e., no online learning). The disturbance model in our system is constructed online, improving from trial to trial as more observations are collected. Ko et al. [15] use a GP-enhanced model and offline reinforcement learning to identify open-loop control sequences for indoor blimp maneuvers. In our system, the learned model is used by an online NMPC algorithm in determining optimal control actions at each control time. Nguyen-Tuong et al. [16] and Park et al. [17] focus on achieving online operation and use local GP (LGP) models to approximate the inverse dynamics of 7-DoF manipulator arms and the forward dynamics of an indoor mobile robot, respectively. In both applications, real-time operation is achieved by partitioning training data into regions and training an LGP model for each region. Predictions are generated by weighted estimation using nearby local models. In this paper, the GP hyperparameters are trained on all data. However, only a local subset of the training data is used when computing a disturbance. Aswani et al. [18] present a general framework for a safe and robust linear LB-MPC algorithm. Disturbance observations are collected online and predictions are made using a Nadaraya−Watson estimator. They present simulated results from the control of a nonlinear jet engine compression system. In this paper, we use NMPC combined with a GP disturbance model, and demonstrate our system operating online in challenging environments. Lehnert and Wyeth [19] investigate Locally Weighted Learning MPC. Initially, the system is controlled by a simple feedback controller while the system learns the parameters for multiple local linear models. Then the set of linear models is incorporated into an MPC algorithm using Receptive Field Weighted Regression. In online operation, the set of models is updated either by adding new models or updating existing parameters. In this paper, our NMPC algorithm is based on a known vehicle kinematic model and a learned general GP disturbance model. This enables continuous operation from the first trial and the representation of complex disturbance characteristics.

Finally, we employ a speed scheduler that addresses the classic exploration versus exploitation trade-off, balancing speed, path-tracking errors, and localization reliability. In practice, it is more common for the speed to be scheduled in reaction to the detection of an obstacle [20]. However, here we assume the desired path is obstacle-free and design the speed scheduling to greedily maximize speed while minimizing path-tracking errors and localization failures.

III. Visual Teach and Repeat

Localization for the mobile robots we use in this paper is provided by an on-board VT&R algorithm developed by Furgale and Barfoot [3] where the sole sensor is an on-board stereo camera. In the first operational phase, the teach phase, the robot is driven along the desired path manually by an operator. Localization in this initial operation is obtained relative to the robot's starting position by visual odometry (VO). In addition to the VO pipeline, path vertices are defined along the path by storing keyframes composed of local feature descriptors and their 3D positions. During the second operational phase, the repeat phase, the robot re-localizes against the stored keyframes thus generating feedback for a path-tracking controller. Re-localization is achieved by matching feature descriptors to generate feature tracks between the current robot view and the teach-pass robot view. As long as sufficient correct feature matches are made, the system generates consistent localization over trials and is able to support a learning-based control algorithm.

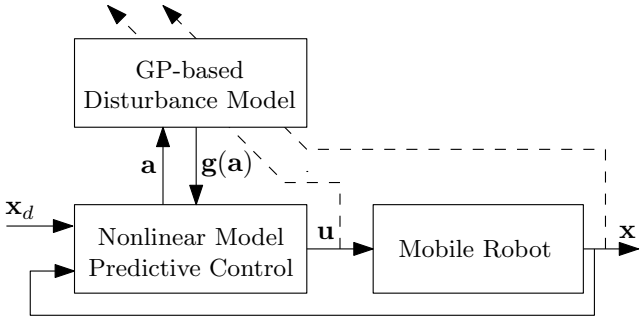

Fig. 3. The controller is composed of two main components: 1) the path-tracking NMPC controller, and 2) the GP-based Disturbance Model.

IV. Mathematical Formulation

A. Nonlinear Model Predictive Control

At a given sample time, the NMPC algorithm finds a sequence of control inputs based on the current state that optimizes the plant behaviour over a prediction horizon. The first control input in the optimal sequence is then applied to the system, resulting in a new system state. The entire process is then repeated at the next sample time for the new system state. In traditional NMPC implementations [6], the process model is specified *a priori* and remains unchanged during operation. In this paper, we augment the process model with a disturbance model generated from experience in order to compensate for effects not captured by the *a priori* process model, such as environmental disturbances and unknown dynamics (Figs. 1, 2, and 3).

Consider the following nonlinear, state-space system:

$$\mathbf{x}_{k+1} = \overbrace{\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}^{\textit{a priori}\text{ model}} + \overbrace{\mathbf{d}(\mathbf{x}_k, \mathbf{v}_k, \mathbf{u}_k)}^{\text{unknown}}, \tag{1}$$

$$\mathbf{v}_{k+1} = \underbrace{\mathbf{h}(\mathbf{v}_k, \mathbf{u}_k)}_{\text{unknown}} \tag{2}$$

with system state, $(\mathbf{x}_k, \mathbf{v}_k)$, and control input, $\mathbf{u}_k$, both at time k. We separate the state into two parts, $\mathbf{x}_k$ and $\mathbf{v}_k$, as we wish only to affect $\mathbf{x}_k$ through our path-tracking control design. The models $\mathbf{f}(\cdot)$, $\mathbf{d}(\cdot)$, and $\mathbf{h}(\cdot)$ are nonlinear process models; $\mathbf{f}(\cdot)$ is an *a priori*, simple vehicle model, $\mathbf{d}(\cdot)$ is an (unknown) general disturbance, and $\mathbf{h}(\cdot)$ represents the (unknown) dynamics of the system. As written, Eqs. 1 and 2 are entirely general, except that we make the assumption that Eq. 2 does not depend on $\mathbf{x}_k$. We can think of $\mathbf{x}_k$ roughly as 'position' and $\mathbf{v}_k$ as 'velocity'. Thus, the chosen model form covers most robotic situations where the dynamics cascade into the kinematics. The specific definitions for our robot experiments are given in Sec. V-A.

By substituting $\mathbf{v}_k = \mathbf{h}(\mathbf{v}_{k-1}, \mathbf{u}_{k-1})$ into Eq. 1, we can write a new version of the state equation as

$$\mathbf{x}_{k+1} = \overbrace{\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}^{\textit{a priori}\text{ model}} + \overbrace{\mathbf{g}(\mathbf{a}_k)}^{\text{learned using GP}}, \tag{3}$$

with

$$\mathbf{a}_k = (\mathbf{x}_k, \mathbf{v}_{k-1}, \mathbf{u}_k, \mathbf{u}_{k-1}), \tag{4}$$

assumed to be observable. The disturbance, $\mathbf{g}(\cdot)$, now includes both unknown disturbances and unmodelled dynamics; we will learn $\mathbf{g}(\cdot)$ from experience over time and represent it using a Gaussian process (more on this in the next section).

As previously mentioned, the objective of the NMPC algorithm is to find a set of controls that optimizes the plant behaviour over a given prediction horizon. To this end, we define the cost function to be minimized over the next K timesteps to be

$$J(\mathbf{u}) = (\mathbf{x}_d - \mathbf{x})^T \mathbf{Q} (\mathbf{x}_d - \mathbf{x}) + \mathbf{u}^T \mathbf{R} \mathbf{u}, \tag{5}$$

where $\mathbf{Q}$ is positive semi-definite, $\mathbf{R}$ is positive definite, $\mathbf{u}$ is a sequence of control inputs,

$$\mathbf{u} = (\mathbf{u}_k, \ldots, \mathbf{u}_{k+K}),$$

$\mathbf{x}_d$ is a sequence of desired states,

$$\mathbf{x}_d = (\mathbf{x}_{d,k+1}, \ldots, \mathbf{x}_{d,k+K+1}),$$

and $\mathbf{x}$ is a sequence of predicted states,

$$\mathbf{x} = (\mathbf{x}_{k+1}, \ldots, \mathbf{x}_{k+K+1}).$$

Since both our process model and disturbance model are nonlinear, the minimum of $J(\mathbf{u})$ must be found iteratively using a nonlinear optimization technique. In this paper, we use unconstrained Gauss-Newton minimization [21] to solve the nonlinear least-squares problem. However, there are other nonlinear optimization algorithms, such as the constrained Gauss-Newton algorithm [6], that could be used to incorporate constraints on the path-tracking errors and control inputs.

Because this optimization is fairly standard, we avoid the minutiae and provide only a high-level sketch. We linearize around an initial guess for the optimal control input sequence, $\bar{\mathbf{u}}$, with $\mathbf{u} = \bar{\mathbf{u}} + \delta\mathbf{u}$. A good initial guess for $\bar{\mathbf{u}}$ is the sequence of optimal inputs calculated in the previous timestep. For the first timestep, we use $\bar{\mathbf{u}} = \mathbf{0}$. With $\bar{\mathbf{x}}$ representing the predicted sequence of states resulting from $\bar{\mathbf{u}}$ and $\mathbf{x} = \bar{\mathbf{x}} + \delta\mathbf{x}$, we can write a linearized equation for the state in lifted form,

$$\delta\mathbf{x} = \mathbf{H}\,\delta\mathbf{u}, \tag{6}$$

where $\mathbf{H}$ is the block-Jacobian of Eq. 3 with respect to $\mathbf{u}$; evaluting this involves computing partials of $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$. In the case of $\mathbf{f}(\cdot)$, we have an analytical model and in the case of $\mathbf{g}(\cdot)$, the derivatives are tractable so long as an appropriate kernel function is chosen for use in the Gaussian process model (see next section). Substituting Eq. 6, $\mathbf{x} = \bar{\mathbf{x}} + \delta\mathbf{x}$, and $\mathbf{u} = \bar{\mathbf{u}} + \delta\mathbf{u}$ into Eq. 5 results in $J(\cdot)$ being exactly quadratic in $\delta\mathbf{u}$. We can can easily find the value of $\delta\mathbf{u}$ that minimizes $J(\cdot)$, update our control input,

$$\bar{\mathbf{u}} \leftarrow \bar{\mathbf{u}} + \delta\mathbf{u}, \tag{7}$$

and iterate to convergence. In accordance with NMPC, we apply the resulting control input for one timestep and start all over at the next timestep.

B. Gaussian Processes

We model the disturbance, $\mathbf{g}(\cdot)$, as a GP based on past observations given a disturbance dependency, $\mathbf{a}$. The model depends on observations of the disturbances collected during previous trials. At time k, we use the estimated poses, $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{x}}_{k-1}$, from the VT&R system, and the control input, $\mathbf{u}_{k-1}$, to solve Eq. 3 for $\hat{\mathbf{g}}(\mathbf{a}_{k-1})$,

$$\hat{\mathbf{g}}(\mathbf{a}_{k-1}) = \hat{\mathbf{x}}_k - \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}). \tag{8}$$

We collect an observation for all sample times in a trial and organize the data from trial j into a set of data pairs, $\mathcal{D}_j = [\{\mathbf{a}_0, \hat{\mathbf{g}}(\mathbf{a}_0)\}, \ldots, \{\mathbf{a}_k, \hat{\mathbf{g}}(\mathbf{a}_k)\}, \ldots, \{\mathbf{a}_{N_j-1}, \hat{\mathbf{g}}(\mathbf{a}_{N_j-1})\}]$, where N_j is the number of timesteps it took to travel the length of the path during trial j, and $\mathbf{a}_k$ is as defined in Eq. 4. After j trials we have datasets $\mathcal{D}_1, \ldots, \mathcal{D}_j$, that we combine into a single database, $\mathcal{D}$, with $N = N_1 + \cdots + N_j$ observations. We also drop the timestep index, k, on each datapair in $\mathcal{D}$, so that when referring to $\mathbf{a}_{\mathcal{D},i}$ or $\hat{\mathbf{g}}_{\mathcal{D},i}$, we mean the ith pair of data in the superset $\mathcal{D}$. Note that there is no requirement that $N_j = N_{j-1}$ as the system simply collects observations as they occur for the length of time that it takes to complete a trial. Moreover, all experiences are treated equally as observations of the underlying unmodelled disturbance. Therefore, the system does not require identical initial conditions, termination conditions, or speed schedules.

In this work, we train a separate GP for each dimension in $\mathbf{g}(\cdot) \in \mathbb{R}^n$ to model disturbances as the robot travels along a path. For simplicity of discussion, we will assume for now that $n = 1$ and denote $\hat{\mathbf{g}}_{\mathcal{D},i}$ by $\hat{g}_{\mathcal{D},i}$. The GP model assumes a measured disturbance originates from a process model,

$$\hat{g}(\mathbf{a}_{\mathcal{D},i}) \sim \mathcal{GP}\left(\mathbf{0}, k(\mathbf{a}_{\mathcal{D},i}, \mathbf{a}_{\mathcal{D},i})\right), \tag{9}$$

with zero mean and kernel function, $k(\mathbf{a}_{\mathcal{D},i}, \mathbf{a}_{\mathcal{D},i})$, to be defined. We assume that each disturbance measurement is corrupted by zero-mean additive noise with variance, σ_n^2, so that $\hat{g}_{\mathcal{D},i} = g_{\mathcal{D},i} + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma_n^2)$. Then a modelled disturbance, $g(\mathbf{a}_k)$, and the N observed disturbances, $\hat{\mathbf{g}} = (\hat{g}_{\mathcal{D},1}, \ldots, \hat{g}_{\mathcal{D},N})$, are jointly Gaussian,

$$\begin{bmatrix} \hat{\mathbf{g}} \\ g(\mathbf{a}_k) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{k}(\mathbf{a}_k)^T \\ \mathbf{k}(\mathbf{a}_k) & k(\mathbf{a}_k, \mathbf{a}_k) \end{bmatrix}\right), \tag{10}$$

where

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{a}_{\mathcal{D},1}, \mathbf{a}_{\mathcal{D},1}) & k(\mathbf{a}_{\mathcal{D},1}, \mathbf{a}_{\mathcal{D},2}) & \ldots & k(\mathbf{a}_{\mathcal{D},1}, \mathbf{a}_{\mathcal{D},N}) \\ k(\mathbf{a}_{\mathcal{D},2}, \mathbf{a}_{\mathcal{D},1}) & k(\mathbf{a}_{\mathcal{D},2}, \mathbf{a}_{\mathcal{D},2}) & \ldots & k(\mathbf{a}_{\mathcal{D},2}, \mathbf{a}_{\mathcal{D},N}) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{a}_{\mathcal{D},N}, \mathbf{a}_{\mathcal{D},1}) & k(\mathbf{a}_{\mathcal{D},N}, \mathbf{a}_{\mathcal{D},2}) & \ldots & k(\mathbf{a}_{\mathcal{D},N}, \mathbf{a}_{\mathcal{D},N}) \end{bmatrix},$$

and

$$\mathbf{k}(\mathbf{a}_k) = \begin{bmatrix} k(\mathbf{a}_k, \mathbf{a}_{\mathcal{D},1}) & k(\mathbf{a}_k, \mathbf{a}_{\mathcal{D},2}) & \ldots & k(\mathbf{a}_k, \mathbf{a}_{\mathcal{D},N}) \end{bmatrix}.$$

In our case, we use the squared-exponential kernel function [7],

$$k(\mathbf{a}_i, \mathbf{a}_j) = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{a}_i - \mathbf{a}_j)^T \mathbf{M}^{-2} (\mathbf{a}_i - \mathbf{a}_j)\right) + \sigma_n^2 \delta_{ij},$$

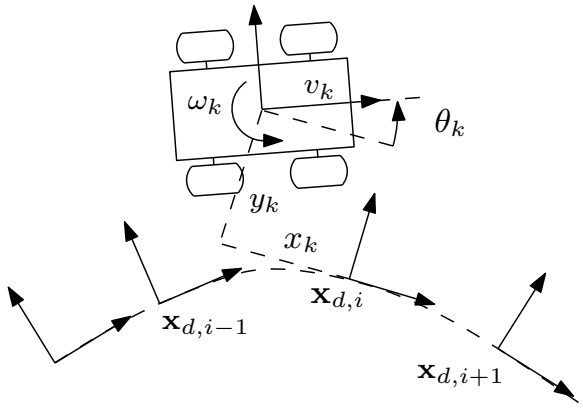

Fig. 4. Definition of the robot velocities, v_k and ω_k, and three pose variables, x_k, y_k and θ_k, calculated relative to the nearest path vertex by Euclidean distance.

where δ_{ij} is the Kronecker delta, that is 1 iff $i = j$ and 0 otherwise, and the constants $\mathbf{M}$, σ_f, and σ_n are hyperparameters. In our implementation with $\mathbf{a}_k \in \mathbb{R}^p$, the constant $\mathbf{M}$ is a diagonal matrix, $\mathbf{M} = \mathrm{diag}(\mathbf{m})$, $\mathbf{m} \in \mathbb{R}^p$, representating the relevance of each component in $\mathbf{a}_k$, while the constants σ_f^2 and σ_n^2, represent the process variation and measurement noise, respectively. Finally, we have that the prediction, $g(\mathbf{a}_k)$, of the disturbance at an arbitrary state, $\mathbf{a}_k$, is also Gaussian distributed,

$$g(\mathbf{a}_k)|\mathbf{g} \sim \mathcal{N}\Big(\mathbf{k}(\mathbf{a}_k)\mathbf{K}^{-1}\hat{\mathbf{g}},\ k(\mathbf{a}_k, \mathbf{a}_k) - \mathbf{k}(\mathbf{a}_k)\mathbf{K}^{-1}\mathbf{k}(\mathbf{a}_k)^T\Big).$$

While we do not make use of the variance information in our controller, in future implementations it could be used as an indication of the uncertainty in the model and used appropriately in deciding the resulting control command. Finally, we include further detail on the storage and retrieval of observations for online operation in Sec. V-C.

C. Gaussian Process Hyperparameter Selection

Having defined the NMPC algorithm and the disturbance model, $\mathbf{g}(\mathbf{a}_k)$, it remains to define the source of the hyperparameter selection. With the squared-exponential kernel function and a disturbance dependency input of size p, our GP has $(2 + p)n$ hyperparameters to be determined. After collecting a set of training data, we find the optimal hyperparameters offline by maximizing the log marginal likelihood using a gradient ascent algorithm, effectively maximizing the probability of the data by adjusting the hyperparameters [7]. In order to avoid local maxima, the algorithm is repeated many times, initialized with different initial values, and the set of hyperparameters resulting in the greatest log marginal likelihood is selected. After training, the hyperparameters are held fixed during online operations.

V. Implementation

A. Defining the Process Model and Variables

In this paper, robots are modelled as unicycle-type vehicles with 'position' state variables, $\mathbf{x}_k = (x_k, y_k, \theta_k)$, calculated relative to the nearest path vertex by Euclidean distance (Fig. 4). The robots have two control inputs, their linear and

angular velocities, $\mathbf{u}_k = (v_{\mathrm{cmd},k}, \omega_{\mathrm{cmd},k})$. The commanded linear velocity is set to the desired, scheduled speed at the nearest path vertex, leaving only the angular velocity, $\omega_{\mathrm{cmd},k}$, for the NMPC algorithm to choose (i.e., we do not optimize the commanded linear velocity).

When the time between control signal updates is defined as Δt, the resulting *a priori* process model employed by the NMPC algorithm is

$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{x}_k + \begin{bmatrix} \Delta t \cos\theta_k & 0 \\ \Delta t \sin\theta_k & 0 \\ 0 & \Delta t \end{bmatrix} \mathbf{u}_k, \tag{11}$$

which represents a simple kinematic model for our robot; it does not account for dynamics or environmental disturbances. We use the same *a priori* model for both robots in our experiments, despite them being quite different in scale.

The 'velocity' state variables are $\mathbf{v}_k = (v_{\mathrm{act},k}, \omega_{\mathrm{act},k})$, which represent the actual linear and rotational speeds of the robot. These will differ from the commanded ones, $\mathbf{u}_k$, owing to the fact that the robots we are working with have fixed rate-control loops that attempt to drive the robot at the commanded velocity. However, the combined dynamics of the robot and these rate controllers are not modelled. We allow the LB-NMPC algorithm to learn these dynamics, as well as any other systematic disturbances, based on experience.

In order to build and query the learned model, $\mathbf{g}(\cdot)$, we require all of the quantities in Eq. 4: $\mathbf{a}_k = (\mathbf{x}_k, \mathbf{v}_{k-1}, \mathbf{u}_k, \mathbf{u}_{k-1})$. We know $\mathbf{u}_k$ and $\mathbf{u}_{k-1}$, as these are commanded inputs, and we obtain $\mathbf{x}_k$ from our vision-based localization system. We also require the 'velocity' state variables, $\mathbf{v}_{k-1} = (v_{\mathrm{act},k-1}, \omega_{\mathrm{act},k-1})$; we could potentially measure these directly using a sensor, but instead approximate them according to

$$v_{\mathrm{act},k-1} = \frac{\sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2}}{\Delta t},$$

and

$$\omega_{\mathrm{act},k-1} = \frac{(\theta_k - \theta_{k-1})}{\Delta t}.$$

This is preferable to using, say, wheel encoders because we want the true speeds with respect to the ground and wheel encoders are unable to measure slip. Because $\mathbf{x}_k$ comes from our vision-based localization system, we are able to measure wheel slip in this way.

B. Automated Speed Scheduler

We implemented an automated speed scheduler to demonstrate the LB-NMPC algorithm's ability to interpolate and extrapolate from learned experiences. For the jth trial, the scheduled speed at the ith path vertex, $v^{(j)}_{\mathrm{sched},i}$, is based on variables recorded at the ith vertex during the previous trial such as the lateral and heading path-tracking errors, $e^{(j-1)}_{L,i}$, $e^{(j-1)}_{H,i}$, respectively,

$$\begin{bmatrix} e^{(j-1)}_{L,i} \\ e^{(j-1)}_{H,i} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} (\mathbf{x}^{(j-1)}_{d,i} - \mathbf{x}^{(j-1)}_{i}),$$

the scheduled speed, $v^{(j-1)}_{\mathrm{sched},i}$, the commanded angular velocity, $\omega^{(j-1)}_{\mathrm{cmd},i}$, and the matched feature count, $c^{(j-1)}_{\mathrm{feature},i}$. Then, using gains for increasing and decreasing the scheduled speed, $\gamma_1 > 1$, $\gamma_2 < 1$, respectively, and thresholds, $\lambda_L \geq 0$, $\lambda_H \geq 0$, $\lambda_v > 0$, $\lambda_\omega > 0$, and $\lambda_{\mathrm{feature}} \geq 4$, the scheduler follows rules to adjust the scheduled velocity as necessary:

$$v^{(j)}_{\mathrm{sched},i} = \tag{12}$$
$$\begin{cases} \gamma_1 v^{(j-1)}_{\mathrm{sched},i} & \text{if } (|e^{(j-1)}_{L,i}| < \lambda_L) \wedge (|e^{(j-1)}_{H,i}| < \lambda_H) \wedge \\ & \quad (|v^{(j-1)}_{\mathrm{sched},i}| < \lambda_v) \wedge (|\omega^{(j-1)}_{\mathrm{cmd},i}| < \lambda_\omega) \wedge \\ & \quad (c^{(j-1)}_{\mathrm{feature},i} > \lambda_{\mathrm{feature}}) \\ \gamma_2 v^{(j-1)}_{\mathrm{sched},i} & \text{if } (|e^{(j-1)}_{L,i}| > \lambda_L) \vee (|e^{(j-1)}_{H,i}| > \lambda_H) \vee \\ & \quad (|v^{(j-1)}_{\mathrm{sched},i}| > \lambda_v) \vee (|\omega^{(j-1)}_{\mathrm{cmd},i}| > \lambda_\omega) \vee \\ & \quad (c^{(j-1)}_{\mathrm{feature},i} < \lambda_{\mathrm{feature}}) \\ v^{(j-1)}_{\mathrm{sched},i} & \text{otherwise.} \end{cases}$$

For the first trial, the scheduled speed at all vertices in the path was set to a fixed linear velocity, $v^{(1)}_{\mathrm{sched},i} = v_{\mathrm{init}}$. Effectively, the automated speed scheduler identifies sections of the path where the system can tolerate additional speed and sections where it cannot, thus balancing the trade-off between speed and path-tracking errors. The scheduler accounts for the limits of the robot and the vision system.

C. Managing Experiences

In order to ensure the LB-NMPC algorithm was executed in constant computation time, our implementation required the ability to use a subset of the observed experiences when computing a disturbance. For this purpose, as experiences were learned, they were binned at each path vertex by commanded velocity. When a bin was considered full, the oldest experience in the bin was discarded. Then when computing a disturbance, experiences were drawn from the bins at nearby path vertices based on commanded velocity.

VI. Experiments

A. Overview

We tested the LB-NMPC algorithm in two separate experiments, including over 1.8 km of learning-enabled path-tracking in GPS-denied environments. In the first experiment, we tested the algorithm's ability to learn unmodelled environmental disturbances, such as the terrain slope depicted in Fig. 1. We tested on a 30-m-long path including unmodelled slopes, sandy surfaces, and gravel surfaces using a 50 kg, four-wheeled Husky A200 robot travelling at a desired speed of 0.4 m/s. The test path for the first experiment included slope angles up to 15°, side-slope angles up to 15°, and path curvatures up to 1 m^{-1} (Fig. 5).

In the second experiment, we tested the algorithm's ability to interpolate and extrapolate from previous experience by having a 150 kg, six-wheeled ROC6 robot learn to drive at a range of scheduled speeds over 20 trials on a 60-m-long path. The speeds were provided by an automated speed scheduler that used path-tracking errors, control inputs, and matched features from previous trials to determine safe speeds for the next trial (Sec. V-B).

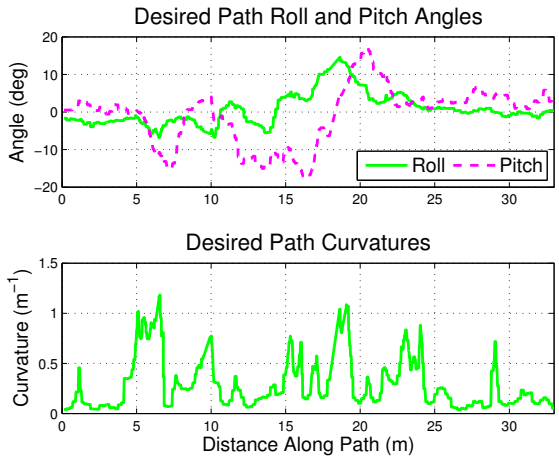

Fig. 5. Experiment 1 Conditions: The test path for the first experiment included slope angles up to 15°, side-slope angles up to 15°, and path curvatures up to 1 m^{-1}. At 17 m along the path, the gravel covered path pitched forward while rolling and turning to the right.

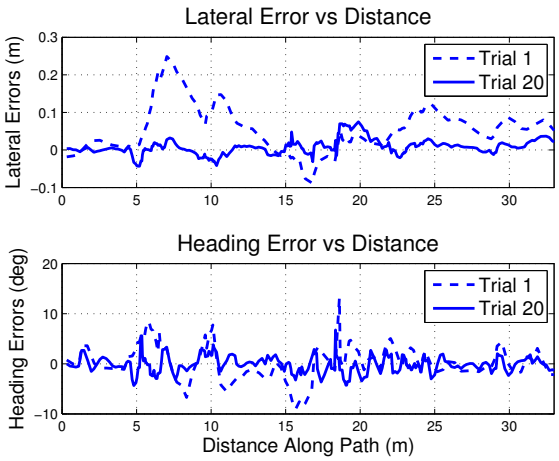

Fig. 6. Experiment 1 Results: Lateral and error versus distance along the path. The maximum heading and lateral errors in trial 20 occurred during the most challenging part of the path at 17 m along the path.

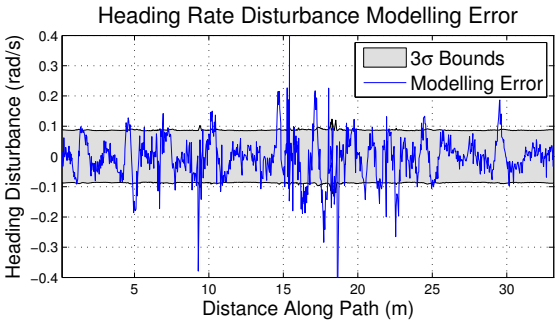

Fig. 7. Experiment 1 Results: Trial 20 heading rate disturbance modelling error versus distance along the path. The disturbance modelling error was largely contained within the 3σ bounds estimated by the GP.

Both experiments were performed in the University of Toronto Institute for Aerospace Studies (UTIAS) MarsDome. In both cases, the controller described in Sec. IV was

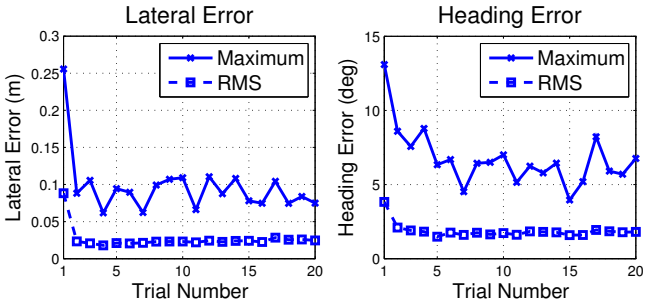

Fig. 8. Experiment 1 Results: Lateral and heading error versus trial number. The maximum and Root-Mean-Square (RMS) errors are reduced significantly within the first few trials.

implemented and run in addition to the VT&R software on a Lenovo W530 laptop with an Intel 2.6 Ghz Core i7 processor with 16 GB of RAM. The camera in both experiments was a Point Grey Bumblebee XB3 stereo camera. The resulting real-time localization and path-tracking control signals were generated at approximately 10 Hz. Since GPS was not available, the improvement due to the LB-NMPC algorithm was quantified by the localization of the VT&R algorithm.

B. Tuning Parameters

The performance of the system was adjusted using the NMPC weighting matrices **Q** and **R**, and the speed scheduler gains and thresholds. The weighting matrices were selected in advance with roughly 2:1:1 ratios weighting heading errors, position errors, and control inputs, and kept the same for all experiments. Otherwise, the speed scheduler gains were set to adjust the scheduled speeds by no more than 0.05 m/s between trials, while maintaining heading errors less than 10°, lateral errors less than 15 cm, matched feature counts greater than 30, linear speeds less than 1.0 m/s, and angular velocities less than 1.0 rad/s.

C. Results

In the first experiment, the robot autonomously travelled the length of the 30-m-long path for 20 trials at a fixed velocity of 0.4 m/s resulting in 600 m of travel. The LB-NMPC algorithm successfully reduced the maximum lateral and heading errors by roughly 50% in the first five trials, then maintained these errors for the next 15 trials (Figs. 6 and 8), demonstrating the system's ability to handle unmodelled terrain. The maximum heading and lateral errors in trial 20 occurred around 17 m along the path where the path pitched forward, rolled to the right, and turned to the right (Figs. 5 and 6). By trial 20, the heading rate disturbance modelling error was largely contained within the 3σ bounds estimated by the GP (Fig. 7).

In the second experiment, the robot autonomously travelled the length of a 60-m-long path for 20 trials (Figs. 2 and 9) at a range of scheduled speeds (Fig. 10) to demonstrate the ability of the disturbance model to interpolate and extrapolate from learned experiences (Fig. 12).

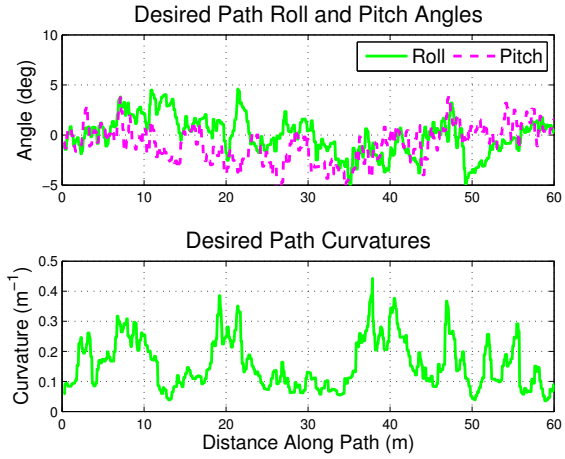

Fig. 9. Experiment 2 Conditions: Slope, side-slope, and curvature versus distance along the path. The test path for the second experiment formed a large 60-m-long loop. In total, the ROC6 repeated the 60-m-long path in 20 trials, resulting in over 1.2 km of testing on this path.

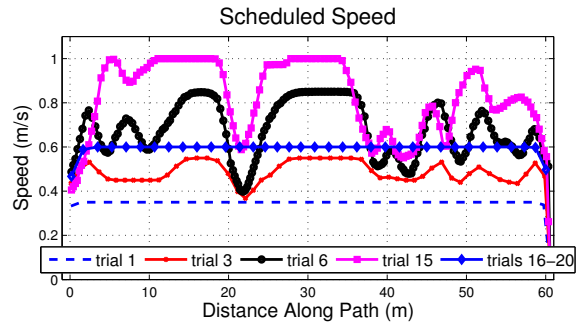

Fig. 10. Experiment 2 Results: Commanded linear speed versus distance along the path. The speed scheduler greedily increased the scheduled speed, thus tending to expand the learned disturbance model rather than exploit it.

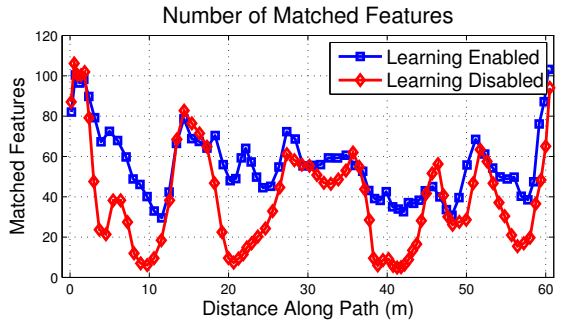

Fig. 11. Experiment 2 Results: Trial 15 VT&R matched features versus distance along the path. In trial 15, the learning algorithm brought the average number of matched features up from 38.33 to 55.77 features resulting in a reduction in temporary localization failures.

For the first 15 trials, the linear speed was incrementally increased on a trial-by-trial basis by the speed scheduler, demonstrating the ability of the disturbance model to extrapolate disturbances from past observations (Fig. 12). The speed scheduler determined where along the path the system could tolerate higher speeds using experience from previous traversals (Fig. 10). In the last five trials, the speed was fixed at 0.6 m/s, a new speed for the system, demonstrating the

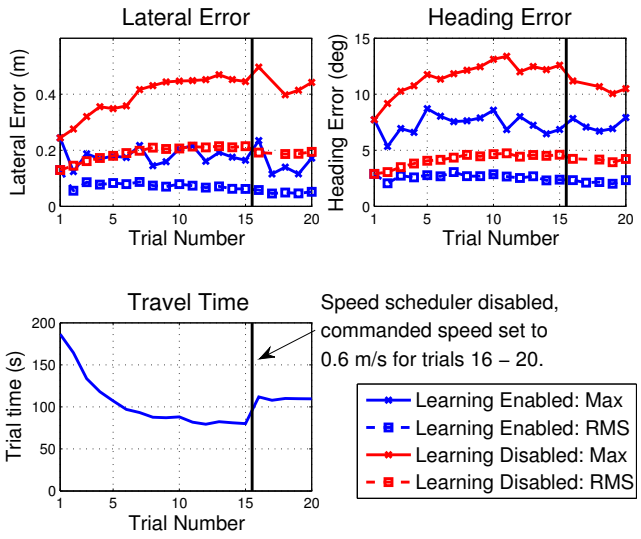

Fig. 12. Experiment 2 Results: In trials 1-15, the scheduled speed is iteratively increased to test the algorithm's ability to extrapolate from learned data. Then in trials 16-20, the scheduled speed is fixed to 0.6 m/s (Fig. 10) to test the algorithm's ability to interpolate from learned data.

ability of the disturbance model to interpolate disturbances from past observations (Fig. 13). Over the course of the 20 trials, the LB-NMPC algorithm was shown to reduce the lateral and heading errors by roughly 50% (Fig. 12) while learning disturbances at speeds ranging from 0.35 m/s to 1.0 m/s (Fig. 10).

In total, the system had learned a model based on roughly 5000 observations (Fig. 13). Of note, the system was unable to travel faster than 0.7 m/s at 40 m along the path due to the path's curvature (Fig. 9). As a result, the system was not able to collect experience above 0.7 m/s for the section of the path around 40 m (Figs. 13 and 14). However, when operating with the learning disabled, the system consistently failed to localize at higher speeds when passing through the turns at 20, 40, and 50 m along the path (Fig. 11). On the other hand, localization was consistent when learning was enabled due to better path-tracking and the resulting increase in the number of matched features.

VII. Conclusion

In summary, this paper presents a Learning-based Nonlinear Model Predictive Control algorithm for a path-repeating, mobile robot negotiating large-scale, GPS-denied, outdoor environments. The disturbance is modelled as a Gaussian Process based on observed disturbances as a function of relevant variables such as the system state and input. Localization for the controller is provided by an on-board, Visual Teach & Repeat mapping and navigation system.

Two experiments on two significantly different robots, including over 1.8 km of travel through challenging GPS-denied outdoor environments, demonstrated the system's ability to handle unmodelled terrain and robot dynamics, and also to interpolate and extrapolate from learned disturbances. In the second experiment, the system used an automated

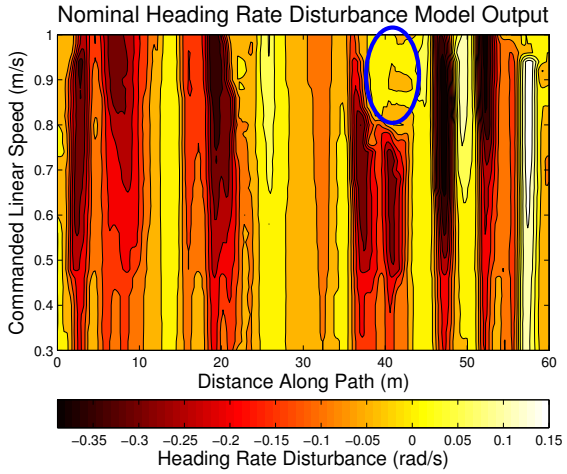

Fig. 13. Experiment 2: The disturbance model maintains a database of observations. We show the learned values for the heading rate disturbance as a function of commanded speed and distance along the path. At 0.9 m/s, 40 m along the path (blue ellipse), there is very little data and the model is untrustworthy (see Fig. 14).

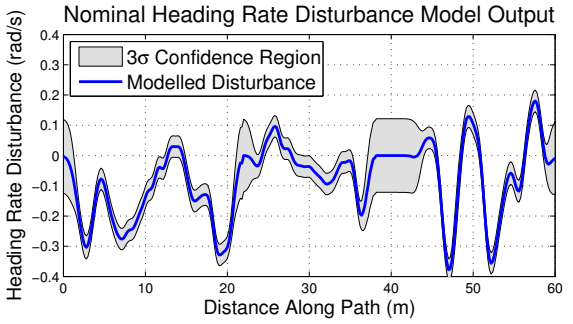

Fig. 14. Experiment 2: Here we show the nominal values for the heading rate disturbance at a commanded speed of 0.9 m/s. The system has no relevant experience when travelling 0.9 m/s at 40 m along the path, therefore the modelled disturbance is zero and 3σ confidence region is relatively large.

speed scheduler based on previous experience to address the classic exploration vs. exploitation trade-off balancing speed and path-tracking errors. The LB-NMPC approach proved to be flexible and effective at reducing path-tracking errors and increasing the reliability of the localization system.

VIII. Acknowledgements

The authors would like to thank the Ontario Ministry of Research and Innovation's Early Research Award Program for funding our research, the Canada Foundation for Innovation for funding the ROC6 robot, and Clearpath Robotics for funding the Husky A200 robot.

References

[1] T. Barfoot, B. Stenning, P. Furgale, and C. McManus. Exploiting reusable paths in mobile robotics: Benefits and challenges for long-term autonomy. *In Proc. of the 9th Canadian Conf. on Computer and Robot Vision (CRV)*, pages 388–395, 2012.

[2] D. Nguyen-Tuong and J. Peters. Model learning for robot control: a survey. *Cognitive Processing*, 12:319–340, 2011.

[3] P. Furgale and T. Barfoot. Visual teach and repeat for long-range rover localization. *Jour. of Field Robotics*, 27(5):534–560, 2010.

[4] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.

[5] M. Morari and J. H. Lee. Model predictive control: past, present and future. *Computers and Chemical Engineering*, 23 (4):667–682, 1999.

[6] M. Diehl, H. J. Ferreau, and N. Haverbeke. Efficient numerical methods for nonlinear mpc and moving horizon estimation. In *Nonlinear Model Predictive Control*, pages 391–417. Springer, 2009.

[7] C.E. Rasmussen and C.K.I. Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, MA, 2006.

[8] F. Kühne, W.F. Lages, and J.M.G. Silva. Mobile robot trajectory tracking using model predictive control. *Proc. of the Latin-American Robotics Symp.*, Sept. 2005.

[9] G. Klančar and I. Škrjanc. Tracking-error model-based predictive control for mobile robots in real time. *Robotics and Autonomous Systems*, 55(6):460–469, 2007.

[10] F. Xie and R. Fierro. First-state contractive model predictive control of nonholonomic mobile robots. *Proc. of the American Control Conf.*, pages 3494–3499, 2008.

[11] S. Peters and K. Iagnemma. Mobile robot path tracking of aggressive maneuvers on sloped terrain. *Proc. of the Int. Conf. on Intelligent Robots and Systems*, pages 242–247, 2008.

[12] M. Burke. Path-following control of a velocity constrained tracked vehicle incorporating adaptive slip estimation. *Proc. of the Int. Conf. on Robotics and Automation*, pages 97–102, 2012.

[13] T. M. Howard, C. J. Green, and A. Kelly. Receding horizon model-predictive control for mobile robot navigation of intricate paths. *Field and Service Robotics*, pages 69–78, 2009.

[14] J. Kocijan, R. Murray-Smith, C.E. Rasmussen, and A. Girard. Gaussian process model based predictive control. In *Proc. of the American Control Conf.*, volume 3, pages 2214–2219, 2004.

[15] J. Ko, D. Klein., D. Fox, and D. Haehnel. Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 742-747, 2006.

[16] D. Nguyen-Tuong, J. Peters, and M. Seeger. Local gaussian process regression for real time online model learning. Advances in Neural Information Processing Systems, pages 1193-1200, 2008.

[17] S. Park, S.K. Mustafa., and K. Shimada. Learning-based robot control with localized sparse online gaussian process. In *Proc. of the Int. Conf. on Intelligent Robots and Systems*, pages 1202-1207, 2013.

[18] A. Aswani, H. Gonzalez, S. Shankar Sastry, and C. Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49:1216–1226, 2013.

[19] C. Lehnert and G. Wyeth. Locally weighted learning model predictive control for nonlinear and time varying dynamics. *Proc. of the Int. Conf. on Robotics and Automation*, pages 2604–2610, 2013.

[20] L. Lapierre, R. Zapata, and P. Lepinay. Combined path-following and obstacle avoidance control of a wheeled robot. *Int. Jour. of Robotics Research*, 26(4):361–375, 2007.

[21] C. F. Gauss. *Méthode des Moindres Carrés*. Mallet-Bachelier, Impreimeur-Libraire de L'École Polytechnique, Quai des Augustins no. 55, Paris, 1855.