Learning-based Nonlinear Model Predictive Control to Improve Vision-based Mobile Robot Path Tracking

Chris J. Ostafew, Angela P. Schoellig, and Timothy D. Barfoot
Institute for Aerospace Studies, University of Toronto, Toronto, Ontario, Canada
e-mail: chris.ostafew@mail.utoronto.ca, schoellig@utias.utoronto.ca, tim.barfoot@utoronto.ca
Jack Collier
Defence Research and Development Canada, Suffield, Alberta, Canada
e-mail: jack.collier@drdc-rddc.gc.ca

Received 6 July 2014; accepted 19 February 2015

This paper presents a Learning-based Nonlinear Model Predictive Control (LB-NMPC) algorithm to achieve high-performance path tracking in challenging off-road terrain through learning. The LB-NMPC algorithm uses a simple *a priori* vehicle model and a learned disturbance model. Disturbances are modeled as a Gaussian process (GP) as a function of system state, input, and other relevant variables. The GP is updated based on experience collected during previous trials. Localization for the controller is provided by an onboard, vision-based mapping and navigation system enabling operation in large-scale, GPS-denied environments. The paper presents experimental results including over 3 km of travel by three significantly different robot platforms with masses ranging from 50 to 600 kg and at speeds ranging from 0.35 to 1.2 m/s (associated video at http://tiny.cc/RoverLearnsDisturbances). Planned speeds are generated by a novel experience-based speed scheduler that balances overall travel time, path-tracking errors, and localization reliability. The results show that the controller can start from a generic *a priori* vehicle model and subsequently learn to reduce vehicle- and trajectory-specific path-tracking errors based on experience.

1. INTRODUCTION

It is well recognized that autonomous guidance, navigation, and control of mobile robots in unstructured, off-road terrain is one of the highest goals in field robotics. For example, robots capable of autonomous off-road operation would be useful in law enforcement, disaster search and rescue, military, forestry, and mining applications. However, operation in off-road terrain requires advanced control techniques to mitigate the effects of unmodeled surface materials (e.g., snow, sand, grass), terrain topography (e.g., side-slopes, inclines), and complex robot dynamics (Figure 1). Finding representative *a priori* models for such effects is challenging since (i) the terrain is often not known ahead of time, (ii) robot-terrain interaction models often do not exist, and (iii) even if such models did exist, finding model parameters is cumbersome. In the past decade, there has been significant work on learning-based controllers for robotics (Nguyen-Tuong & Peters, 2011; Schaal & Atkeson, 2010). Learning-based algorithms alleviate the need for significant engineering work to identify and model all disturbances that a model-based controller may be required to mitigate.

In paper, we investigate a Learning-based Nonlinear Model Predictive Control (LB-NMPC) algorithm for a path-repeating mobile robot operating in challenging outdoor terrain. The algorithm uses a fixed, simple robot model and a learned, nonparametric disturbance model. The goal is to reduce path-tracking errors using real-world experience and a disturbance model instead of preprogramming accurate analytical models that are generally difficult to derive. Disturbances represent measured discrepancies between the *a priori* model and the observed system behavior. They are modeled as a Gaussian process (GP) based on previous experience as a function of state, input, and other relevant variables. Modeling the disturbances as a GP enables the algorithm to learn complex nonlinear model discrepancies and generalize to novel situations.

We also investigate a novel experience-based speed scheduler. During the first trial, when the controller is based solely on the *a priori* model, the speed is set such that path tracking is achieved with tolerable errors and reliable vision-based localization. After this first pass, the scheduler adjusts the planned speed based on previous experience (i.e., tracking errors and localization quality). The new speed schedule achieves faster overall path completion while guaranteeing low path-tracking errors and reliable localization. In this system, the LB-NMPC algorithm is also shown to interpolate and extrapolate from experience. Preliminary results for the scheduler operating with a fixed

	DMRV	ROC6	Clearpath Husky
Mass	600 kg	150 kg	50 kg
Size (LW)	2 m x 1.5 m	1.5 m x 0.5 m	0.9 m x 0.6 m
Steering	Ackermann Steering	Skid Steering	Skid Steering

Figure 1. Robots used to demonstrate the effectiveness of the learning controller. Despite significant differences in robot mass, wheel base, kinematics, and actuator designs, the algorithm uses the same nominal model for all three robots and learns disturbances over trials in order to accurately track desired paths.

feedback controller have been published in Ostafew, Collier, Schoellig, & Barfoot (2014a).

Localization for the controller is provided by an onboard, Visual Teach & Repeat (VT&R) mapping and navigation algorithm enabling operation in large-scale, GPS-denied environments (Furgale and Barfoot, 2010; Stenning, McManus, & Barfoot, 2013). In the first operational phase, namely the teach phase, the robot is piloted along the desired path. Localization in this phase is obtained relative to the robot's starting position by Visual Odometry (VO), computing pose changes over sequential images based on extracted feature maps (three-dimensional positions and associated descriptors). Then at discrete points along the desired path, the algorithm stores the currently viewed feature map. During the repeat phase, the algorithm relocalizes against stored feature maps given the current robot view, thus generating feedback for a path-tracking controller (Figure 2). As long as a sufficient number of feature matches are made between the live robot view and the stored feature maps, the system generates consistent localization over trials and is able to support a learning control algorithm.

The key contributions of this paper are as follows: (i) a path-tracking, LB-NMPC algorithm based on a simple *a priori* process model and learned disturbance model; (ii) an experience-based speed scheduler balancing overall travel time, path-tracking errors, and vision-based localization reliability; and (iii) extensive outdoor experiments on three different robot platforms ranging from 50 to 600 kg with both skid and Ackermann steering (Figure 1). This paper is an extension of previous work (Ostafew, Schoellig, & Barfoot, 2014b). Significant additions include a detailed description and discussion of the LB-NMPC algorithm, an illustrative example, and an additional experiment showing successful learning on an Ackermann-steered robot; all previous test robots were skid-steered. The structure of this paper is as follows. Section 2 relates our work to other research in this field. Sections 3 and 4 describe the proposed LB-NMPC algorithm and implementation details for our experiments, respectively. Section 3.4 presents simulation results, giving an intuition of the benefits of the LB-NMPC algorithm, while Section 5 presents experimental results, demonstrating the successful operation of the algorithm in practice. Finally, Sections 6 and 7 present a discussion and conclusion.

2. RELATED WORK

In this section, we present related work aimed at achieving high-performance path-tracking in spite of unknown disturbances. First we provide a brief review of approaches involving model-based control with (in some cases) online parameter identification. Then we provide a background on learning control approaches.

2.1. Model-based and Adaptive Controllers

In our testing, the two largest sources of disturbances were the unmodeled dynamics of the robot (including actuators)

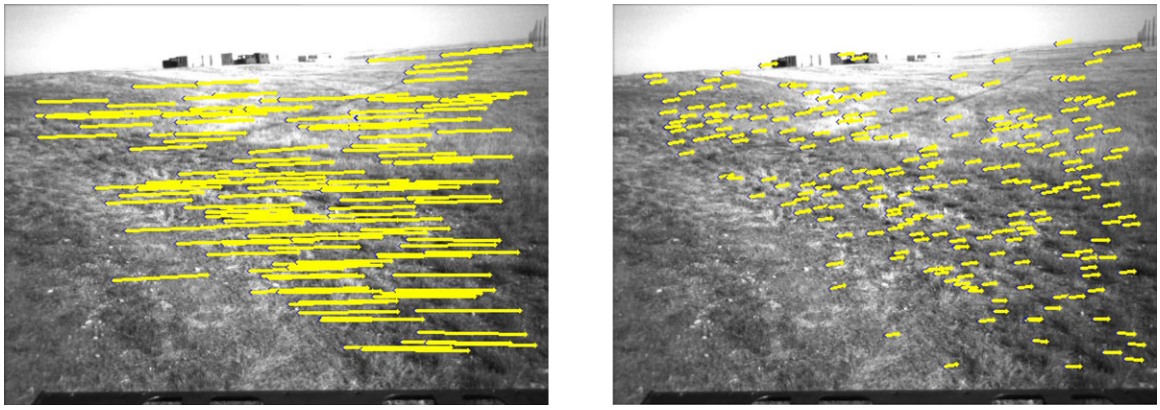

Figure 2. Visual representations of relocalization by our Visual Teach & Repeat (VT&R) navigation algorithm with high (left image) and low (right image) path-tracking errors. Each feature track represents the translation between a feature identified during the VT&R teach phase and the current repeat phase. Reducing the path-tracking errors leads to improved reliability of our VT&R algorithm since it is sensitive to perspective changes.

and the wheel-terrain interactions. One method for mitigating the effects of unknown wheel-terrain interaction is to design a robot with all-wheel drive and steering such that lateral and angular vehicle slip can be compensated directly. For example, Ishigami, Nagatani, and Yoshida (2009) estimate the vehicle slip angle and path-tracking errors using visual and wheel odometry. Then, using two separate proportional feedback controllers, they command the front wheels so as to reduce path-tracking errors, and the rear wheels so as to compensate for the vehicle slip angle. Similarly, Helmick et al. (2006), Helmick, Angelova, and Matthies (2009), and Angelova, Matthies, Helmick, and Perona (2007) estimate lateral and angular vehicle slip rates using visual and wheel odometry. Then they use proportional feedback control to generate desired lateral and angular velocities to compensate for vehicle slip rates. Finally, they use the robot's inverse dynamics to generate desired individual wheel speeds and orientations. However, these approaches can only *react* to path-tracking errors and vehicle slip. On the other hand, our approach is based on Model Predictive Control (MPC), including a learned model representing wheel-terrain interactions, robot dynamics, and other systematic disturbances, and it can therefore act in anticipation of tracking errors.

Cariou, Lenain, Thuilot, and Berducat (2009) and Guillet, Lenain, and Thuilot (2013) propose online adaptive controllers mitigating wheel slip and robot dynamics. They demonstrate feedback-linearized controllers based on kinematic models extended with wheel slip angles. The slip angles are estimated online using observers. They address robot dynamics using a predictive controller including future path curvatures and offline tuned values representing actuator delay and robot inertias. Unlike these controllers, which partially identify and model disturbances *a priori*, our learning approach treats disturbances, including both wheel slip and robot dynamics, as a GP, enabling the representation of complex disturbance characteristics not known prior to operation. Furthermore, since disturbances are learned and stored in memory, our algorithm can anticipate disturbances and act accordingly.

MPC is a control framework that uses a process model directly. 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 (Rawlings and Mayne, 2009). Kühne, Lages, and Silva (2005), Klančar and Škrjanc (2007), and Xie and Fierro (2008) present MPC-based mobile robot controllers based on kinematic models and show results for robots traveling on smooth, flat surfaces. Howard, Green, and Kelly (2009) demonstrate MPC on a large-scale, outdoor robot navigating intricate paths. Finally, Peters and Iagnemma (2008) demonstrate MPC for a mobile robot where the process model includes effects such as tire deformation, wheel-terrain interaction, and suspension compliance. However, in each of these examples, the controllers are based on *a priori* models and, in some cases, rely on parameters whose determination in practice is challenging. For example, Iagnemma, Kang, Shibly, and Dubowsky (2004) demonstrate an online method of estimating terramechanics parameters where the rover speed is restricted to 10 cm/s in order to assume a quasi-static analysis. At higher speeds, Seegmiller, Rogers-Marcovitz, Miller, and Kelly (2013) determine vehicle model parameters, including wheel slip, by integrated prediction error minimization. In this paper, our NMPC algorithm is based on a fixed nominal model and a learned, nonparametric disturbance model. This reduces the need for accurate *a priori* process models and parameter-specific observers while maintaining the benefits of MPC such as predictive behavior and constraint handling.

2.2. Learning Controllers

Unlike controllers based on fixed models, controllers using learned models gather data over time, incrementally constructing accurate approximations of the true system model. In this paper, we model disturbances as a GP based on input-output data from previous trials. This approach enables both model flexibility and consistent uncertainty estimates (Rasmussen, 2006). For example, Kocijan, Murray-Smith, Rasmussen, and Girard (2004) combine a GP model and MPC 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. MPC is applied to control the system based on the offline-identified GP model (i.e., no online learning). While their work was restricted to offline simulation, our algorithm is used for real-time path-tracking and learns from trial to trial. Sparse GP approximations are one approach to enable fast, online GP evaluation, and they do so by discarding some training points and keeping only "inducing inputs," also known as "support points" (Quiñonero-Candela and Rasmussen, 2005). An alternative are local GP (LGP) methods, as implemented in this work, which enable online operation by dividing the GP input space into smaller subspaces and generating an LGP for each subspace (Rasmussen and Ghahramani, 2002; Snelson and Ghahramani, 2007). For example, Nguyen-Tuong, Peters, and Seeger (2009) and Meier, Hennig, and Schaal (2014) focus on achieving online operation and use LGP models to approximate the inverse dynamics of seven degrees of freedom (7-DoF) manipulator arms. Unlike these two examples, where many LGP models are generated for operation, we rapidly compute a single LGP model online based on a sliding window of learned data and use NMPC to enable predictive control. Finally, robustness of learning controllers is a large unanswered question. Aswani, Gonzalez, Shankar Sastry, and Tomlin (2013) focus on developing a safe and robust LB-MPC approach using Tube MPC (Langson, Chrysochoos, Raković, and Mayne, 2004). The approach produces optimal inputs based on the learned system dynamics. However, they ensure safety and robustness by checking whether these inputs keep the nominal model stable when it is subject to uncertainty. In this paper, we do not explicitly consider the robustness of the controller but focus on the practical application of LB-NMPC to mobile robots. This requires continuous operation from the first trial and representation of complex disturbances by the learned model.

Additionally, Iterative Learning Control (ILC) and Reinforcement Learning (RL) are two other common approaches to learning from experience. Schoellig, Mueller, and D'Andrea (2012) and Ostafew, Schoellig, and Barfoot (2013) present ILC algorithms for quadrotors and mobile robots, respectively, that learn a feedforward control signal over sequential trials. Unlike ILC, our LB-NMPC algorithm learns a flexible, general disturbance model that allows interpolation and extrapolation of learned experience, and thus covers multiple paths and speed schedules simultaneously. RL, on the other hand, learns a control policy that maximizes a cumulative expected reward. For example, Abbeel, Quigley, and Ng (2006) and Ko, Klein, Fox, and Haehnel (2007) present RL algorithms for the control of a mobile robot and an autonomous blimp, respectively. However, unlike our algorithm, which provides continuous operation from the first trial, RL is known to require a prohibitively large number of training examples before operation, an issue for RL that is the focus of much current work (Deisenroth, Fox, and Rasmussen, 2014).

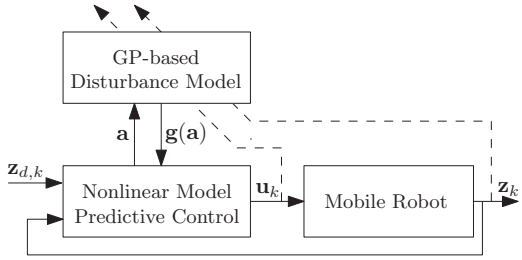

Figure 3. The LB-NMPC algorithm is composed of two parts: 1) the path-tracking NMPC algorithm that includes a nominal process model, and 2) the GP-based disturbance model. During the first trial, the algorithm relies solely on the nominal process model to guide the vehicle along the desired path, $\mathbf{z}_d$. In subsequent trials, the NMPC algorithm uses the disturbance model as a correction to the nominal model at states, $\mathbf{a}$, to be defined in Section 3.1. Dashed lines indicate that the signals $\mathbf{z}_k$ and $\mathbf{u}_k$ update the model.

3. MATHEMATICAL FORMULATION

3.1. Nonlinear Model Predictive Control

At a given sample time, the NMPC algorithm finds a sequence of control inputs that optimizes the plant behavior over a prediction horizon based on the current state. The first input in the optimal sequence is then applied to the system, resulting in a new system state. The entire process is repeated at the next sample time for the new system state. In traditional NMPC implementations (Rawlings and Mayne, 2009), 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 fixed process model, such as environmental disturbances and unknown dynamics (Figure 3).

3.1.1. Full-state Feedback Control

Consider the following nonlinear, state-space system:

$$\mathbf{z}_{k+1} = \mathbf{f}_{\text{true}}(\mathbf{z}_k, \mathbf{u}_k), \tag{1}$$

with an observable system state, $\mathbf{z}_k \in \mathbb{R}^n$, and control input, $\mathbf{u}_k \in \mathbb{R}^m$, both at time k. In this work, the true system is not known exactly and is represented by the sum of an *a priori* model and an experience-based, learned model,

$$\mathbf{z}_{k+1} = \overbrace{\mathbf{f}(\mathbf{z}_k, \mathbf{u}_k)}^{\textit{a priori}\text{ model}} + \overbrace{\mathbf{g}(\mathbf{z}_k, \mathbf{u}_k)}^{\text{learned disturbance model}}. \tag{2}$$

The models $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$ are nonlinear process models: $\mathbf{f}(\cdot)$ is a known nominal process model representing our knowledge of $\mathbf{f}_{\text{true}}(\cdot)$, while $\mathbf{g}(\cdot)$ is an (initially unknown) disturbance model representing discrepancies between the nominal model and the actual system behavior. The system is further assumed to be Markovian, thus the processes $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$ involve only states from the current time.

As previously mentioned, the objective of the NMPC algorithm is to find a set of controls that optimizes the plant behavior over a given prediction horizon. Toward that end, we define the cost function to be minimized over the next K time-steps to be

$$J(\mathbf{u}) = (\mathbf{z}_d - \mathbf{z})^T \mathbf{Q} (\mathbf{z}_d - \mathbf{z}) + \mathbf{u}^T \mathbf{R}\, \mathbf{u}, \tag{3}$$

where $\mathbf{Q} \in \mathbb{R}^{Kn \times Kn}$ is positive semidefinite, $\mathbf{R} \in \mathbb{R}^{Km \times Km}$ is positive-definite, $\mathbf{u}$ is a sequence of control inputs $\mathbf{u} = (\mathbf{u}_k, \ldots, \mathbf{u}_{k+K-1})$, $\mathbf{z}_d$ is a sequence of desired states $\mathbf{z}_d = (\mathbf{z}_{d,k+1}, \ldots, \mathbf{z}_{d,k+K})$, $\mathbf{z}$ is a sequence of predicted states $\mathbf{z} = (\mathbf{z}_{k+1}, \ldots, \mathbf{z}_{k+K})$, obtained from Eq. (2) when applying $\mathbf{u}$, and K is a given prediction horizon length. Weighting on the state begins at time $k+1$ since the state at time k can no longer be affected by the control input. Also, by requiring $\mathbf{R}$ to be positive-definite, inputs are guaranteed to be finite. Further restrictions on control inputs or states are commonly imposed using constraints when solving for the optimal control input (Diehl, Ferreau, and Haverbeke, 2009).

Since both our process model and our 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 (Nocedal and Wright, 1999) to solve the nonlinear least-squares problem. We begin by linearizing 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 time-step. For the first time-step, we use $\bar{\mathbf{u}} = \mathbf{0}$. With $\bar{\mathbf{z}}$ representing a sequence of states obtained from Eq. (2) when applying $\bar{\mathbf{u}}$ and with $\mathbf{z} = \bar{\mathbf{z}} + \delta\mathbf{z}$, and $\bar{\mathbf{z}}_k = \mathbf{z}_k$, we find

$$\bar{\mathbf{z}}_{k+b+1} = \mathbf{f}(\bar{\mathbf{z}}_{k+b}, \bar{\mathbf{u}}_{k+b}) + \mathbf{g}(\bar{\mathbf{z}}_{k+b}, \bar{\mathbf{u}}_{k+b}) \tag{4}$$

and

$$\delta\mathbf{z}_{k+b+1} \approx \mathbf{H}_{\mathbf{z},k+b}\, \delta\mathbf{z}_{k+b} + \mathbf{H}_{\mathbf{u},k+b}\, \delta\mathbf{u}_{k+b}, \tag{5}$$

where

$$\mathbf{H}_{\mathbf{z},k+b} = \left.\frac{\partial \mathbf{f}(\cdot)}{\partial \mathbf{z}}\right|_{\bar{\mathbf{z}}_{k+b}, \bar{\mathbf{u}}_{k+b}} + \left.\frac{\partial \mathbf{g}(\cdot)}{\partial \mathbf{z}}\right|_{\bar{\mathbf{z}}_{k+b}, \bar{\mathbf{u}}_{k+b}},$$
$$\mathbf{H}_{\mathbf{u},k+b} = \left.\frac{\partial \mathbf{f}(\cdot)}{\partial \mathbf{u}}\right|_{\bar{\mathbf{z}}_{k+b}, \bar{\mathbf{u}}_{k+b}} + \left.\frac{\partial \mathbf{g}(\cdot)}{\partial \mathbf{u}}\right|_{\bar{\mathbf{z}}_{k+b}, \bar{\mathbf{u}}_{k+b}} \tag{6}$$

for $b \in \{0, \ldots, K-1\}$. 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 a continuously differentiable kernel function is chosen for use in the Gaussian process model (see Section 3.2). As $\mathbf{z}_k$ is the current state as measured, $\delta\mathbf{z}_k = \mathbf{0}$. Given Eqs. (5) and (6), we have

$$\delta\mathbf{z} = \mathbf{H}_{\mathbf{z}} \delta\mathbf{z} + \mathbf{H}_{\mathbf{u}} \delta\mathbf{u} \tag{7}$$

$$= (\mathbf{1} - \mathbf{H}_{\mathbf{z}})^{-1} \mathbf{H}_{\mathbf{u}}\, \delta\mathbf{u} \tag{8}$$

$$= \mathbf{H}' \delta\mathbf{u}, \tag{9}$$

where $\mathbf{1}$ represents an identity matrix, $\mathbf{H}_{\mathbf{u}} = \text{diag}(\mathbf{H}_{\mathbf{u},k}, \ldots, \mathbf{H}_{\mathbf{u},k+K-1})$, and

$$\mathbf{H}_{\mathbf{z}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{\Lambda}_{\mathbf{z}} & \mathbf{0} \end{bmatrix}, \quad \mathbf{\Lambda}_{\mathbf{z}} = \text{diag}(\mathbf{H}_{\mathbf{z},k+1}, \ldots, \mathbf{H}_{\mathbf{z},k+K-1}). \tag{10}$$

Substituting $\mathbf{z} = \bar{\mathbf{z}} + \delta\mathbf{z}$, Eq. (9), and $\mathbf{u} = \bar{\mathbf{u}} + \delta\mathbf{u}$ into Eq. (3) results in $J(\cdot)$ being quadratic in $\delta\mathbf{u}$,

$$J(\mathbf{u}) = (\mathbf{z}_d - \bar{\mathbf{z}} - \delta\mathbf{z})^T \mathbf{Q} (\mathbf{z}_d - \bar{\mathbf{z}} - \delta\mathbf{z}) + (\bar{\mathbf{u}} + \delta\mathbf{u})^T \mathbf{R} (\bar{\mathbf{u}} + \delta\mathbf{u}) \tag{11}$$

$$\approx (\mathbf{z}_d - \bar{\mathbf{z}} - \mathbf{H}'\delta\mathbf{u})^T \mathbf{Q} (\mathbf{z}_d - \bar{\mathbf{z}} - \mathbf{H}'\delta\mathbf{u}) + (\bar{\mathbf{u}} + \delta\mathbf{u})^T \mathbf{R} (\bar{\mathbf{u}} + \delta\mathbf{u}). \tag{12}$$

We can find the value of $\delta\mathbf{u}$ that minimizes $J(\cdot)$ by solving

$$\frac{\partial J(\mathbf{u})}{\partial \delta\mathbf{u}} = \mathbf{0} \tag{13}$$

for $\delta\mathbf{u}$, and compute the control input about which (3) is linearized in the next iteration,

$$\bar{\mathbf{u}} \leftarrow \bar{\mathbf{u}} + \delta\mathbf{u}. \tag{14}$$

After iterating to convergence, we apply the first element of the resulting optimal control input sequence for one time-step, and start all over at the next time-step.

3.1.2. Partial-state Feedback Control

Consider the following system, covering most robotic systems, where the dynamics cascade into the kinematics:

$$\text{kinematics:} \quad \mathbf{x}_{k+1} = \mathbf{f}_{\mathbf{x},\text{true}}(\mathbf{x}_k, \mathbf{v}_k), \tag{15}$$

$$\text{dynamics:} \quad \mathbf{v}_{k+1} = \mathbf{f}_{\mathbf{v},\text{true}}(\mathbf{v}_k, \mathbf{u}_k), \tag{16}$$

with the system state, $\mathbf{z}_k = (\mathbf{x}_k, \mathbf{v}_k)$, representing pose, $\mathbf{x}_k \in \mathbb{R}^{n_\mathbf{x}}$, and velocity, $\mathbf{v}_k \in \mathbb{R}^{n_\mathbf{v}}$, separately, and control input, $\mathbf{u}_k$, all at time k. By substituting $\mathbf{v}_k = \mathbf{f}_{\mathbf{v},\text{true}}(\mathbf{v}_{k-1}, \mathbf{u}_{k-1})$ into Eq. (15), we can write

$$\mathbf{x}_{k+1} = \mathbf{f}'_{\text{true}}(\mathbf{x}_k, \mathbf{v}_{k-1}, \mathbf{u}_{k-1}). \tag{17}$$

Now, if we assume that our *a priori* model represents the robot kinematics with $\mathbf{v}_k = \mathbf{u}_k$ (i.e., the *a priori* model assumes robot dynamics are negligible), and that the true process, $\mathbf{f}'_{\text{true}}(\cdot)$, can be represented by the sum of our *a priori* and learned models, we find

$$\mathbf{x}_{k+1} = \overbrace{\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}^{\textit{a priori}\text{ model}} + \overbrace{\mathbf{g}(\underbrace{\mathbf{x}_k, \mathbf{v}_{k-1}, \mathbf{u}_k, \mathbf{u}_{k-1}}_{\mathbf{a}_k})}^{\text{learned model}}, \tag{18}$$

with disturbance query state, $\mathbf{a}_k \in \mathbb{R}^p$,

$$\mathbf{a}_k = (\mathbf{x}_k, \mathbf{v}_{k-1}, \mathbf{u}_k, \mathbf{u}_{k-1}). \tag{19}$$

In other words, in order to capture the dynamics of the system, the disturbance query state, $\mathbf{a}_k$, is now required to include historic states. We can further define the corresponding cost function to be

$$J(\mathbf{u}) = (\mathbf{x}_d - \mathbf{x})^T \mathbf{Q}_\mathbf{x} (\mathbf{x}_d - \mathbf{x}) + \mathbf{u}^T \mathbf{R}\, \mathbf{u}, \tag{20}$$

where $\mathbf{Q}_\mathbf{x} \in \mathbb{R}^{Kn_\mathbf{x} \times Kn_\mathbf{x}}$ is positive-semidefinite, $\mathbf{R}$ and $\mathbf{u}$ are as in Eq. (3), $\mathbf{x}_d$ is a sequence of desired states, $\mathbf{x}_d = (\mathbf{x}_{d,k+1}, \ldots, \mathbf{x}_{d,k+K})$, $\mathbf{x}$ is a sequence of predicted states, $\mathbf{x} = (\mathbf{x}_{k+1}, \ldots, \mathbf{x}_{k+K})$, and K is the given prediction horizon length. The state, $\mathbf{x}_k$, and learned model, $\mathbf{g}(\cdot)$, are now of reduced dimension, $n_\mathbf{x} \leq n$, while still capturing both unknown disturbances and unmodeled dynamics. This approach enables a user to provide a simple *a priori* model with few parameters, if any. Further, the derivation suggests that the approach is applicable to processes with even higher-order dynamics by continuing to add historic states to the disturbance dependency.

3.2. Gaussian Process Disturbance Model

We model the disturbance, $\mathbf{g}(\cdot)$, as a GP, which is a function of a disturbance dependency, $\mathbf{a}$. The model depends on observations of the disturbances collected during previous trials, representing attempts to achieve a control objective, such as tracking a path from start to finish. 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 isolate Eq. (18) for $\hat{\mathbf{g}}(\mathbf{a}_{k-1})$,

$$\hat{\mathbf{g}}(\mathbf{a}_{k-1}) = \hat{\mathbf{x}}_k - \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}). \tag{21}$$

We collect observations 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 time-steps it took to travel the length of the path during trial j, and $\mathbf{a}_k$ is as defined in Eq. (19). After j trials, we have multiple 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 time-step index, k, on each data pair 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 unmodeled disturbance. In fact, the system collects experience data whenever it moves while repeating the desired path. As a result, 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. This approach makes the assumption that disturbances are uncorrelated. 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 learned model assumes a measured disturbance originates from a Gaussian process model,

$$\hat{g}(\mathbf{a}_{\mathcal{D},i}) \sim \mathcal{GP}(\mathbf{0}, k(\mathbf{a}_{\mathcal{D},i}, \mathbf{a}_{\mathcal{D},i})), \tag{22}$$

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 modeled 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,

$$\begin{bmatrix} \hat{\mathbf{g}} \\ g(\mathbf{a}_k) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{k}(\mathbf{a}_k)^T \\ \mathbf{k}(\mathbf{a}_k) & k(\mathbf{a}_k, \mathbf{a}_k) \end{bmatrix}\right), \tag{23}$$

where $\mathbf{K} \in \mathbb{R}^{N \times N}$ with $(\mathbf{K})_{i,j} = k(\mathbf{a}_{\mathcal{D},i}, \mathbf{a}_{\mathcal{D},j})$, and $\mathbf{k}(\mathbf{a}_k) = [k(\mathbf{a}_k, \mathbf{a}_{\mathcal{D},1}),\ k(\mathbf{a}_k, \mathbf{a}_{\mathcal{D},2}), \ldots,\ k(\mathbf{a}_k, \mathbf{a}_{\mathcal{D},N})]$. In our case, we use the Squared-Exponential (SE) kernel function (Rasmussen, 2006),

$$k(\mathbf{a}_i, \mathbf{a}_j) = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{a}_i - \mathbf{a}_j)^T \mathbf{M}^{-2} (\mathbf{a}_i - \mathbf{a}_j)\right) + \sigma_n^2\, \delta_{ij}, \tag{24}$$

where δ_{ij} is the Kronecker delta, which is 1 if and only if $i = j$ and 0 otherwise, and the constants $\mathbf{M}$, σ_f, and σ_n are hyperparameters. The SE kernel function is an example of a radial basis function (Rasmussen, 2006) and is commonly used to represent continuous functions based on dense data. Further, the SE kernel is continuously and analytically differentiable, enabling the rapid computation of derivatives for the Gauss-Newton optimization algorithm. In our implementation with $\mathbf{a}_k \in \mathbb{R}^p$, the constant $\mathbf{M}$ is a diagonal

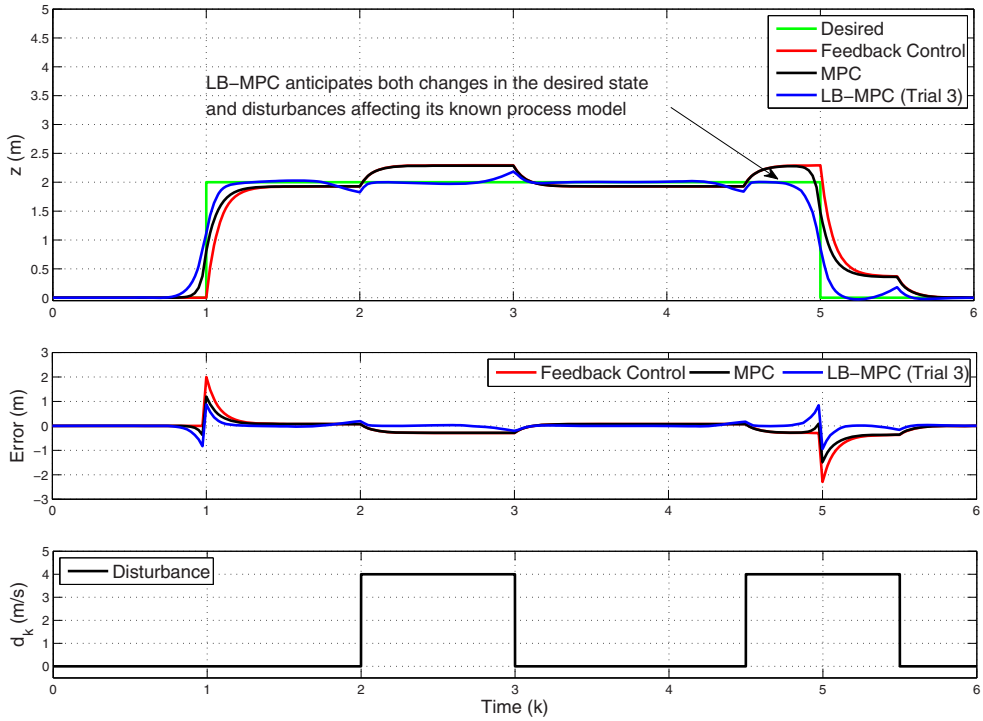

Figure 4. Compared to both simple feedback control (red) and MPC (black), LB-NMPC (blue) is able to anticipate and reduce errors caused by changes in the desired state and unmodeled disturbances.

matrix, $\mathbf{M} = \text{diag}(\mathbf{m})$, $\mathbf{m} \in \mathbb{R}^p$, representing 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,

$$g(\mathbf{a}_k)|\hat{\mathbf{g}} \sim \mathcal{N}\left(\mathbf{k}(\mathbf{a}_k)\mathbf{K}^{-1}\hat{\mathbf{g}},\ k(\mathbf{a}_k,\mathbf{a}_k) - \mathbf{k}(\mathbf{a}_k)\mathbf{K}^{-1}\mathbf{k}(\mathbf{a}_k)^T\right). \tag{25}$$

In this work, we only make use of the predicted mean value of disturbances. However, in future work, the predicted variance could be used as an indication of the uncertainty in the learned 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 Section 4.3.

3.3. Gaussian Process Hyperparameter Selection

Having defined the NMPC algorithm and disturbance model, $\mathbf{g}(\mathbf{a}_k)$, it remains to define the source of the hyperparameters, $\mathbf{M}$, σ_f^2, and σ_n^2. Solving for optimal hyperparameters is not currently a real-time process in our experiments. As such, we assume that a suitable set of hyperparameters has been determined prior to each trial based on previous experience (i.e., from previous trials). For the first trial, when the robot has no experience, the predicted disturbance is zero. Given a set of experiences, we find the optimal hyperparameters offline by maximizing the log marginal likelihood of collected experiences using a gradient ascent algorithm (Rasmussen, 2006). To avoid local maxima, the algorithm is repeated several times, initialized with different initial values, and the set of hyperparameters resulting in the greatest likelihood is selected.

3.4. Illustrative Example

In this section, we highlight the benefits of LB-NMPC and present an illustrative example comparing (i) fixed feedback control, (ii) nonlearning NMPC, and (iii) LB-NMPC. Consider the following process model:

$$z_{k+1} = \alpha\, z_k + \Delta t\, \beta\, u_k + \Delta t\, d_k, \tag{26}$$

with system state, $z_k \in \mathbb{R}$, control input, $u_k \in \mathbb{R}$, and time-dependent disturbance, $d_k \in \mathbb{R}$, shown in Figure 4. Further, α, $\beta \in \mathbb{R}$ are unknown constants. In simulation, they are 0.99 and 0.5, respectively. The goal is to track a sequence of desired states, $z_{\text{d},k}$, as shown in green in Figure 4, which

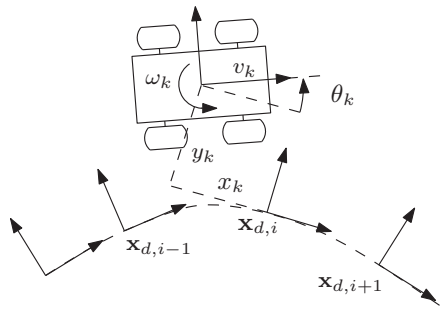

Figure 5. Definition of the robot velocities, v_k and ω_k, and three pose variables, x_k, y_k, and θ_k. At each time-step, the VT&R algorithm provides an estimate of the robot position relative to the nearest desired pose by Euclidean distance.

is known to the example controllers prior to starting. The feedback controller uses a simple feedback law,

$$u_{\text{fb},k} = k_{\text{fb}}\,(z_{d,k} - z_k). \tag{27}$$

Both the NMPC and LB-NMPC controllers assume a nominal process model,

$$z_{k+1} = z_k + \Delta t\, u_k, \tag{28}$$

a prediction horizon, $K = 10$, and a cost function (3), with $\mathbf{Q} = 10\times\mathbf{1}$ and $\mathbf{R} = 0.01\times\mathbf{1}$, where $\mathbf{1}$ is the identity matrix. The LB-NMPC algorithm also includes a learned disturbance model, as described in Section 3.1, such that the complete system model used by the LB-NMPC algorithm is

$$z_{k+1} = z_k + \Delta t\, u_k + g(z_k, u_k, k). \tag{29}$$

In this simple example, the disturbances are a function of time (26) and hence the learned disturbance is a function of time, k. However, in practice, we assume disturbances are time-invariant (18).

As expected, the feedback controller is incapable of anticipating errors caused by either changes in desired state, $z_{d,k}$, or disturbances, d_k (Figure 4). On the other hand, the MPC controller (without a learned model) enables some amount of predictive control to reduce tracking errors due to changes in the desired state. However, tracking errors are not canceled completely because the MPC algorithm does not have the correct process model. Finally, the LB-NMPC algorithm exploits its previous experience to predict and compensate for both changes in the desired state and unknown disturbances not anticipated by the *a priori* process model.

4. IMPLEMENTATION

4.1. Robot Model

In this paper, robots are modeled (Figure 5) as unicycle-type vehicles with "position" state variables (18), $\mathbf{x}_k = (x_k,\ y_k,\ \theta_k)$. At every time-step, the VT&R localization algorithm provides the position of the robot, $\mathbf{x}_k$, relative to the nearest desired pose by Euclidean distance (Figure 6). The robots have two control inputs, their linear and angular velocities, $\mathbf{u}_k = (v_{\text{cmd},k}, \omega_{\text{cmd},k})$. The commanded linear velocity, $v_{\text{cmd},k}$, is constrained to the scheduled speed for the jth trial at the nearest path vertex, $v_{\text{cmd},k} = v^{(j)}_{\text{sched},i}$, leaving only the angular velocity, $\omega_{\text{cmd},k}$, for the NMPC algorithm to optimize considering (20). Prior to each trial, scheduled path speeds are optimized by the experience-based speed scheduler (Section 4.2).

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

$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{x}_k + \begin{bmatrix} \Delta t\,\cos\theta_k & 0 \\ \Delta t\,\sin\theta_k & 0 \\ 0 & \Delta t \end{bmatrix} \mathbf{u}_k, \tag{30}$$

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 all robots in our experiments, despite the fact that they are quite different in scale (Figure 1).

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

To build and query the learned model, $\mathbf{g}(\cdot)$, throughout the prediction horizon, we require all of the quantities in Eq. (19): $\mathbf{a}_{k+b} = (\mathbf{x}_{k+b}, \mathbf{v}_{k-1+b}, \mathbf{u}_{k+b}, \mathbf{u}_{k-1+b})$, $b \in B = \{0, \ldots, K-1\}$. We know $\mathbf{u}_{k+b}$ and $\mathbf{u}_{k-1+b}$, $b \in B$, as these are commanded inputs. We initially obtain the robot position from our vision-based localization system, $\mathbf{x}_k = \hat{\mathbf{x}}_k$, and then from our system model (18), $\mathbf{x}_{k+1+b} = \mathbf{f}(\mathbf{x}_{k+b}, \mathbf{u}_{k+b}) + \mathbf{g}(\mathbf{a}_{k+b})$, $b \in B$. Finally, we compute the velocity state variables, $\mathbf{v}_{k-1+b} = (v_{\text{act},k-1+b}, \omega_{\text{act},k-1+b})$, based on the computed robot positions,

$$v_{\text{act},k-1+b} = \frac{\sqrt{(x_{k+b} - x_{k-1+b})^2 + (y_{k+b} - y_{k-1+b})^2}}{\Delta t}, \quad b \in B$$

and

$$\omega_{\text{act},k-1+b} = \frac{(\theta_{k+b} - \theta_{k-1+b})}{\Delta t}, \quad b \in B,$$

with $\mathbf{x}_{k-1} = \hat{\mathbf{x}}_{k-1}$. Since $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{x}}_{k-1}$ come from our vision-based localization system, we are able to initialize the predictive controller with accurate velocity estimates with respect to the ground. This is preferable to using wheel

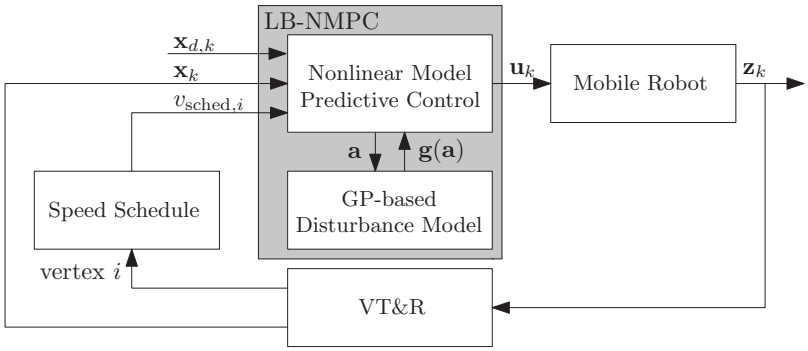

Figure 6. In practice, our overall system combines the LB-NMPC algorithm (Section 3), the experience-based speed scheduler (Section 4.2), and a vision-based VT&R system for localization.

encoders because they are unable to measure wheel slip and other ground-interaction effects.

4.2. Automated Speed Scheduler

We implemented an automated speed scheduler (Ostafew et al., 2014a) to demonstrate the LB-NMPC algorithm's ability to interpolate and extrapolate from learned experiences. The algorithm uses experience from previous trials to schedule speeds that minimize travel time while ensuring reliable localization, low path-tracking errors, and realizable control inputs. Effectively, the scheduler incrementally increases or decreases speeds along the path where possible or necessary, respectively, requiring the LB-NMPC algorithm to interpolate and extrapolate from previous experience.

When using vision-based localization systems, there exists a speed limit above which localization becomes unreliable and the safety of the robot can no longer be assured. This speed limit may come as a result of motion blur, a degraded scene (relative to when the path was taught), or large deviations from the path. As an indicator of the conditions faced by the localization system, we record the number of features matched by the VT&R system, $c^{(j)}_{\text{feature},i}$, when passing the ith vertex during the jth trial for use in the speed scheduler.

We also record the lateral and heading path-tracking errors, $e^{(j-1)}_{L,i}$ and $e^{(j-1)}_{H,i}$, respectively,

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

when passing the ith vertex during the jth trial. Since we have assumed that the desired path is safe and free of obstacles, it is important to maintain low path-tracking errors. Further, the vision system is sensitive to perspective changes between the teach pass and any repeat pass. Perspective changes are the direct result of path-tracking errors and reduce the reliability of the localization system.

Finally, the scheduled linear speed also addresses constraints on angular velocities resulting from actuator limits. The true robot angular velocity differs from the commanded velocity as a result of wheel slip, side slopes, and other model discrepancies. As a result, we record the commanded angular velocity, $\omega^{(j)}_{\text{cmd},i}$, during the jth trial when passing the ith path vertex.

If path-tracking errors are below (above) a threshold, control inputs are below (above) a threshold, and (or) there is a sufficient (insufficient) number of matched features in a certain section, then the speed in that section is increased (decreased). Using tuned values for increasing and decreasing the scheduled speed, $\gamma_1 > 0$ and $\gamma_2 > 0$, respectively, and thresholds, $\lambda_L \geq 0$, $\lambda_H \geq 0$, $\lambda_\omega > 0$, and $\lambda_{\text{feat}} \geq 3$, the scheduler follows rules to generate the suggested speeds for each path vertex:

$$v^{(j+1)}_{\text{sched},i} = \begin{cases} v^{(j)}_{\text{sched},i} + \gamma_1 & \text{if } (|e^{(j)}_{L,i}| < \lambda_L) \wedge (|e^{(j)}_{H,i}| < \lambda_H) \wedge \\ & \quad (|\omega^{(j)}_{\text{cmd},i}| < \lambda_\omega) \wedge (c^{(j)}_{\text{feature},i} > \lambda_{\text{feat}}), \\ v^{(j)}_{\text{sched},i} - \gamma_2 & \text{if } (|e^{(j)}_{L,i}| > \lambda_L \lambda_{\text{db}}) \vee (|e^{(j)}_{H,i}| > \lambda_H \lambda_{\text{db}}) \vee \\ & \quad (|\omega^{(j)}_{\text{cmd},i}| > \lambda_\omega \lambda_{\text{db}}) \vee (c^{(j)}_{\text{feature},i} < \lambda_{\text{feat}}/\lambda_{\text{db}}), \\ v^{(j)}_{\text{sched},i} & \text{otherwise.} \end{cases} \quad (31)$$

Effectively, the automated speed scheduler identifies sections of the path where the system can tolerate higher speeds and sections where it cannot, thus balancing the tradeoff between speed, path-tracking errors, and vision-based localization reliability. We use $\lambda_{\text{db}} > 1$ to produce a deadband where the speed at a vertex is neither increased nor decreased. For the first trial, the scheduled speed at all vertices in the path was set to a fixed speed, $v^{(1)}_{\text{sched},i} = v_{\text{init}}$.

4.3. Managing Experiences

To ensure the LB-NMPC algorithm is executed in constant computation time, our implementation requires the

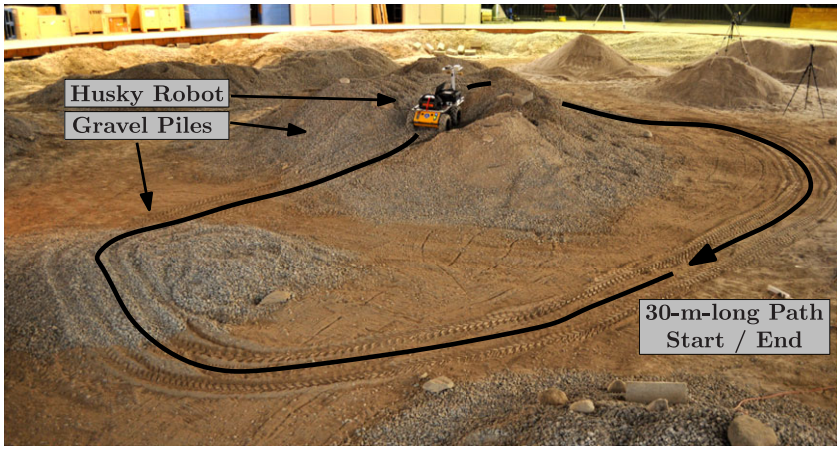

Figure 7. The first and second experiments were conducted inside the University of Toronto Institute for Aerospace Studies (UTIAS) MarsDome on gravel, sand, and loose dirt. The 30-m-long path, shown here, was used for the first experiment. In all experiments, the nominal unicycle model used in our LB-NMPC algorithm included no prior information on wheel-terrain interactions or robot dynamics.

ability to use a subset of the observed experiences when computing a disturbance. Similar to work by Nguyen-Tuong et al. (2009) and Meier et al. (2014), we employ a local model. However, unlike their work, we use a single sliding local model. As experiences are learned, they are stored in bins, $\mathcal{D}_{i,l}$, by path vertex, i, and commanded velocity, $l = \lfloor v_{\text{cmd},k}/v_{\text{bin}} \rfloor$, where v_{bin} represents the velocity discretization and $\lfloor \cdot \rfloor$ represents the floor function. When the number of experiences in a bin exceeds a threshold, c_{bin}, the oldest experience in the bin is discarded. Then, when computing a control input at the ith vertex, a "local" dataset is created, drawing experiences from bins at nearby path vertices and commanded velocities, $\mathcal{D} = \{\mathcal{D}_{a,b} | \, a \in \{i - c_{\text{vertex}}, \ldots, i + c_{\text{vertex}}\}, b \in \{l - c_{\text{velocity}}, \ldots, l + c_{\text{velocity}}\}\}$. Thus, models are effectively assembled on demand rather than precomputing hundreds of local models, enabling a constant-time algorithm independent of path length or deployment time.

5. FIELD TESTING

5.1. Overview

We tested the LB-NMPC algorithm in three different experiments involving three significantly different mobile robots (Figure 1) and paths with dirt, gravel, sand, grass, inclines, and side slopes. This resulted in over 3 km of learning-enabled path-tracking in GPS-denied environments. The three tests demonstrate the algorithm's effectiveness at reducing path-tracking errors with only cursory prior knowledge of the robot's behavior (i.e., that it could be treated as a unicycle robot, Section 4.1). Details on the tuning parameters are presented in Section 5.2.

The first experiment (Section 5.3) demonstrated the algorithm's ability to learn unmodeled environmental disturbances. We tested on a 30-m-long path including slopes, dusty ground, and loose gravel surfaces (Figure 7). The robot was a 50 kg, four-wheeled Clearpath Husky robot traveling at a desired speed of 0.4 m/s (i.e., the automated speed scheduler was disabled for the first experiment). With a 0.5 m wheelbase, Husky robots are relatively small and agile skid-steered mobile robots. As such, the path included slope angles up to 15°, side-slope angles up to 15°, and path curvatures up to 1 m^{-1}.

The second experiment (Section 5.4) demonstrated the algorithm's ability to interpolate and extrapolate from previous experience. We used a 150 kg, six-wheeled ROC6 robot (Figure 1) learning to drive at a range of scheduled speeds over 20 trials on a 60-m-long path. Like the Husky, the ROC6 robot is a skid-steered platform. However, the ROC6 is heavier and longer, with a 1.5 m wheelbase, and it is better suited to operate in more open terrains at higher speeds. Scheduled speeds for each trial, $v_{\text{sched},k}$, were provided by the proposed automated scheduler that used matched features, path-tracking errors, and control inputs from previous trials to determine safe speeds for the next trial (Section 4.2).

Finally, the third experiment (Section 5.5) further demonstrated the algorithm's ability to learn disturbances due to robot design. Whereas the first two experiments involved skid-steered robots, this experiment used a 600 kg, Ackermann-steered DMRV robot (Figure 1). Traditional path-tracking controllers would represent the robot using a bicycle model (Figure 8) with steering angle, $\delta_{\text{cmd},k}$, and linear velocity, $v_{\text{cmd},k}$, as control inputs. However, in this paper, the LB-NMPC algorithm treats the Ackermann-steered robot as a unicycle robot with linear and angular

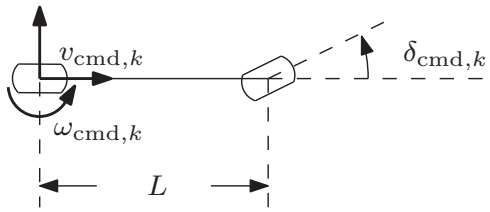

Figure 8. Here we show the relationship between the steering angle of an Ackermann-steered robot, $\delta_{\mathrm{cmd},k}$, and the linear and angular velocities, $v_{\mathrm{cmd},k}$ and $\omega_{\mathrm{cmd},k}$, respectively. In experiment 3, we used the LB-NMPC algorithm for path-tracking on a 600 kg, Ackermann-steered mobile robot.

velocity commands, $v_{\mathrm{cmd},k}$ and $\omega_{\mathrm{cmd},k}$, respectively. The robot then converted these velocity commands to a steering angle, $\delta_{\mathrm{cmd},k}$,

$$\delta_{\mathrm{cmd},k} = \tan^{-1}\left(\frac{L\,\omega_{\mathrm{cmd},k}}{v_{\mathrm{cmd},k}}\right), \tag{32}$$

where L is defined as the wheelbase of the Ackermann-steered robot. The robot learned to drive at a range of scheduled speeds over 10 trials on a 100-m-long path. The scheduled speeds for each trial, $v_{\mathrm{sched},k}$, were generated using the same automated speed scheduler (Section 4.2) as was used in the second experiment.

The first and second experiments were performed in the University of Toronto Institute for Aerospace Studies (UTIAS) MarsDome in Toronto, Ontario, Canada (Figure 7). The third experiment was performed at the Defence Research and Development Canada (DRDC) Experimental Proving Grounds in Suffield, Alberta, Canada. In all experiments, the controller described in Section 3 was 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 all tests was a Point Grey Bumblebee XB3 stereo camera. The resulting real-time localization and path-tracking control signals were generated at approximately 10 Hz. As previously mentioned, hyperparameter selection is currently an offline process, taking up to 5 min in the later trials of an experiment when the system had accumulated approximately 5,000 experiences. Since GPS was not available, the improvement due to the LB-NMPC algorithm was quantified by the localization of the VT&R algorithm. The VT&R algorithm is based on visual odometry and provides localization with errors less than 4 cm/m when compared against GPS ground-truth (Stenning, McManus, & Barfoot, 2013).

5.2. Tuning Parameters

The performance of the system was adjusted using the NMPC weighting matrices $\mathbf{Q}_x$ and $\mathbf{R}$, the experience management parameters, and the speed scheduler gains and thresholds. The weighting matrices for each test were selected in advance ranging from roughly a 3:1 ratio weighting path-tracking errors and control inputs for the 50 kg Husky to a 1:1 ratio for the 600 kg DMRV robot. The increased weighting on the control inputs for the heavier robots was selected to ensure controller stability at higher speeds. Local GP models were generated based on a sliding window of size, $c_{\mathrm{vertex}} = 5$ and $c_{\mathrm{velocity}} = 1$, where velocities were discretized by $v_{\mathrm{bin}} = 0.25\,\mathrm{m/s}$. The maximum number of experiences per bin, c_{bin}, was set to 4, resulting in local models based on up to 180 experiences. Finally, the speed scheduler parameters we used are shown in Table I.

5.3. Experiment 1: Learning to Follow a Path with a Fixed Speed Schedule

In the first experiment, the 50 kg Husky robot autonomously traveled the length of a 30-m-long path for 20 trials at a fixed speed of 0.4 m/s resulting in 600 m of travel (Figure 7). Figure 9 shows plots of path characteristics, path-tracking errors, and angular velocity control input vs. distance along the path. The plots include comparisons between the first trial (red), when the learned model has no experience from which to draw, and the 20$^{\mathrm{th}}$ trial (blue), when the learned model has a significant amount of experience from which to draw. The heading and lateral errors, e_H and e_L, respectively, reached their peaks in trial 20 (blue) around 14-22 m along the path where the path pitched forward, rolled to the right, and turned to the right. This section also corresponded to the largest changes in control input between the first and last trial. In Figure 10, we show plots of the maximum and RMS path-tracking errors vs. trial number. By disabling the speed scheduler for experiment 1, the learned model was allowed to converge. As a result, the LB-NMPC algorithm successfully reduced the maximum lateral and heading errors by roughly 75% in the first few trials, then maintained these errors for the next 15 trials. However, even after many trials, the maximum and RMS errors continued to vary. We suspect that these changes were due mainly to evolving path conditions (e.g., ruts, dirt piles, etc.) and our experience management scheme, which handles computational complexity and changing disturbances by forgetting experiences over time.

5.4. Experiment 2: Learning to Follow a Path at Increasing Speeds

In the second experiment, the 150 kg ROC6 robot autonomously traveled the length of a 60-m-long path at a range of scheduled speeds over 20 trials to demonstrate the ability of the algorithm to interpolate and extrapolate from learned experiences (Figure 11). The path for the second experiment was mainly on level ground, but it included path curvatures up to 0.5 m^{-1}, suiting the capabilities of the ROC6 robot. Figure 12 shows plots of path characteristics, scheduled speeds, and VT&R matched features vs.

Table I. Speed scheduler gains and thresholds. The scheduler was not used in experiment 1.

	γ_1	γ_2	λ_L	λ_H	λ_{feature}	λ_ω	λ_{db}
Experiment 1	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Experiment 2	0.15	0.1	0.15 m	10°	30	1.0 rad/s	1.1
Experiment 3	0.2	0.15	0.15 m	10°	30	1.0 rad/s	1.1

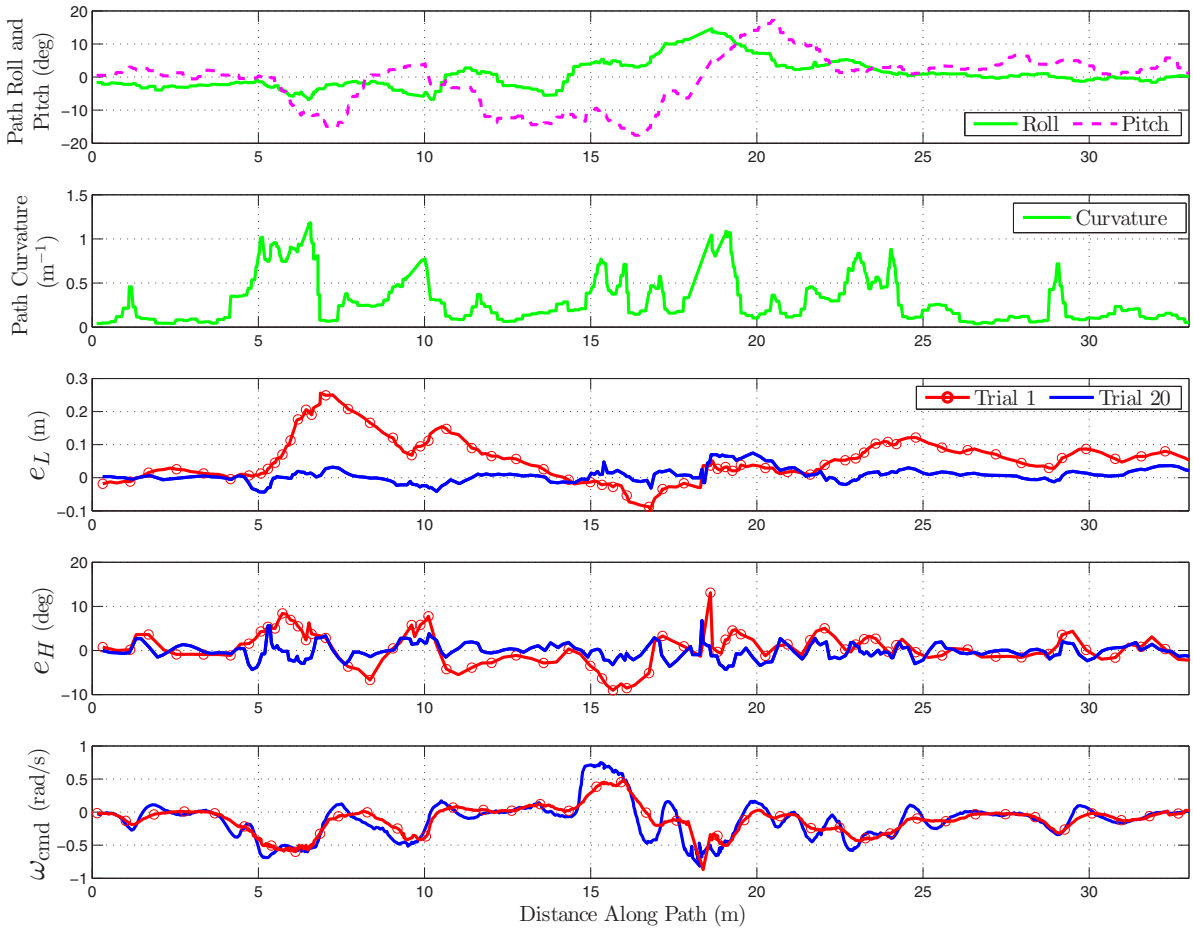

Figure 9. The desired path for experiment 1 included slope angles up to 15°, side-slope angles up to 15°, and path curvatures up to 1 m^{-1} as estimated relative to the start of the path by the VT&R algorithm. We also show the lateral and heading path-tracking errors, $e_{L,i}^{(j)}$ and $e_{H,i}^{(j)}$, and the commanded angular velocity, $\omega_{\text{cmd},i}^{(j)}$, for $j = \{1, 20\}$.

distance along the path. The speed scheduler (Section 4.2) determined where along the path the system could tolerate higher speeds using experience from previous traversals, thus minimizing the travel time in sequential trials. In some sections of the path (e.g., at ∼22 m), the system took up to three trials before safely increasing the scheduled speed. This does not necessarily mean the learned model in these sections had converged, but only that the path-tracking errors, the matched feature counts, and the control inputs were within the specified limits for the speed scheduler (Section 5.2). In general, the speed schedules resulted in the robot learning to drive the path faster, increasing speeds from 0.35 to 1.0 m/s. Sections of the path with poor lighting and high curvature, such as at 10, 20, and 40 m along the path, had relatively low VT&R matched features. In these sections, the speed scheduler suggested increased speeds, though not as high as sections with good lighting and low curvature, such as at 15 or 30 m along the path. Further, with

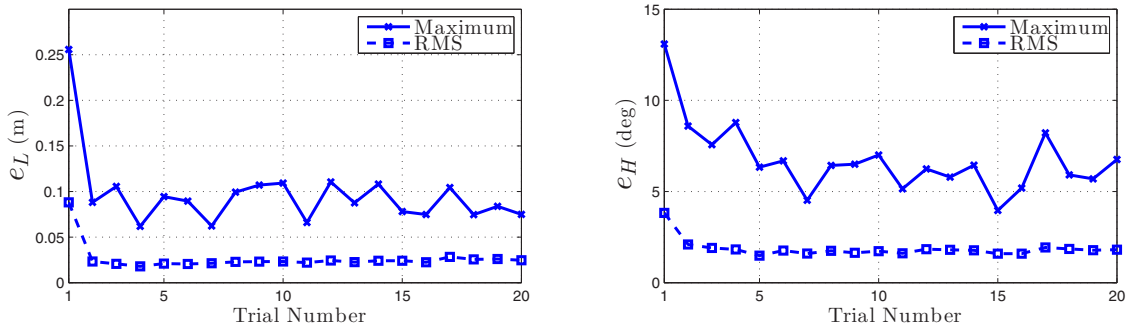

Figure 10. The maximum and root mean square (RMS) path-tracking errors in experiment 1 were reduced significantly within the first few trials. Since MPC is an optimal controller balancing path-tracking errors and control input, we do not expect the path-tracking errors to be eliminated completely.

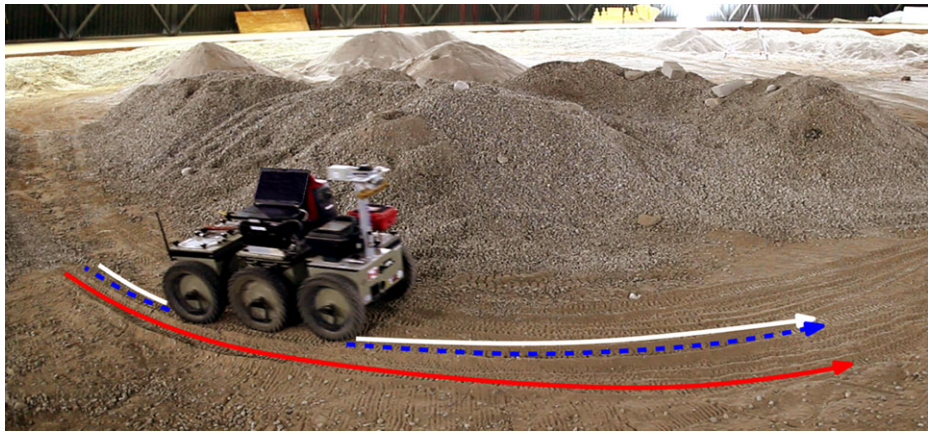

Figure 11. The second and third experiments focused on the algorithm's ability to learn unmodeled robot dynamics. Here we show the skid-steered ROC6 robot driving at 0.6 m/s with learning enabled. The white line shows the desired trajectory (tire tracks), the red line shows a trajectory with learning disabled, while the dashed blue line shows a trajectory with learning enabled and reduced path-tracking errors.

learning enabled and reduced path-tracking errors, the average number of matched features was increased from 38.33 to 55.77. Since the VT&R localization algorithm depends on matching features between the live-view and teach-pass view, an increase in matches tends to result in an increase in the localization reliability for the vision-based mapping and localization system. Figure 13 shows plots of the maximum path-tracking errors, root mean square (RMS) path-tracking errors, and overall travel time vs. trial number. The LB-NMPC algorithm reduced the lateral and heading errors by roughly 50% over the course of the 20 trials, enabling a significant reduction in travel time.

Figure 14 shows the learned model output vs. distance along the path and commanded speed. Even though our system collects discrete measurements of the underlying disturbance function, it is able to continuously interpolate and extrapolate from the data. In the second experiment, the system had collected roughly 20,000 observations for the learned model, retaining only 5,000 observations after 20 trials based on our experience management scheme. Note that the system was unable to travel faster than 0.8 m/s at 40 m along the path due to the path's curvature. As a result, the system was not able to collect experience above 0.8 m/s for this section of the path, and the resulting modeled disturbance is close to zero with relatively high uncertainty (Figure 15). Nonetheless, Figures 13 and 14 show that our LB-NMPC algorithm is capable of effectively maintaining a learned model for many operating conditions simultaneously, learning new disturbances as required while maintaining a wealth of knowledge from previous experience.

5.5. Experiment 3: Learning to Follow a Path at Increasing Speeds with an Ackermann-steered Robot

In the third experiment, the 600 kg, Ackermann-steered robot autonomously traveled the length of a 100-m-long

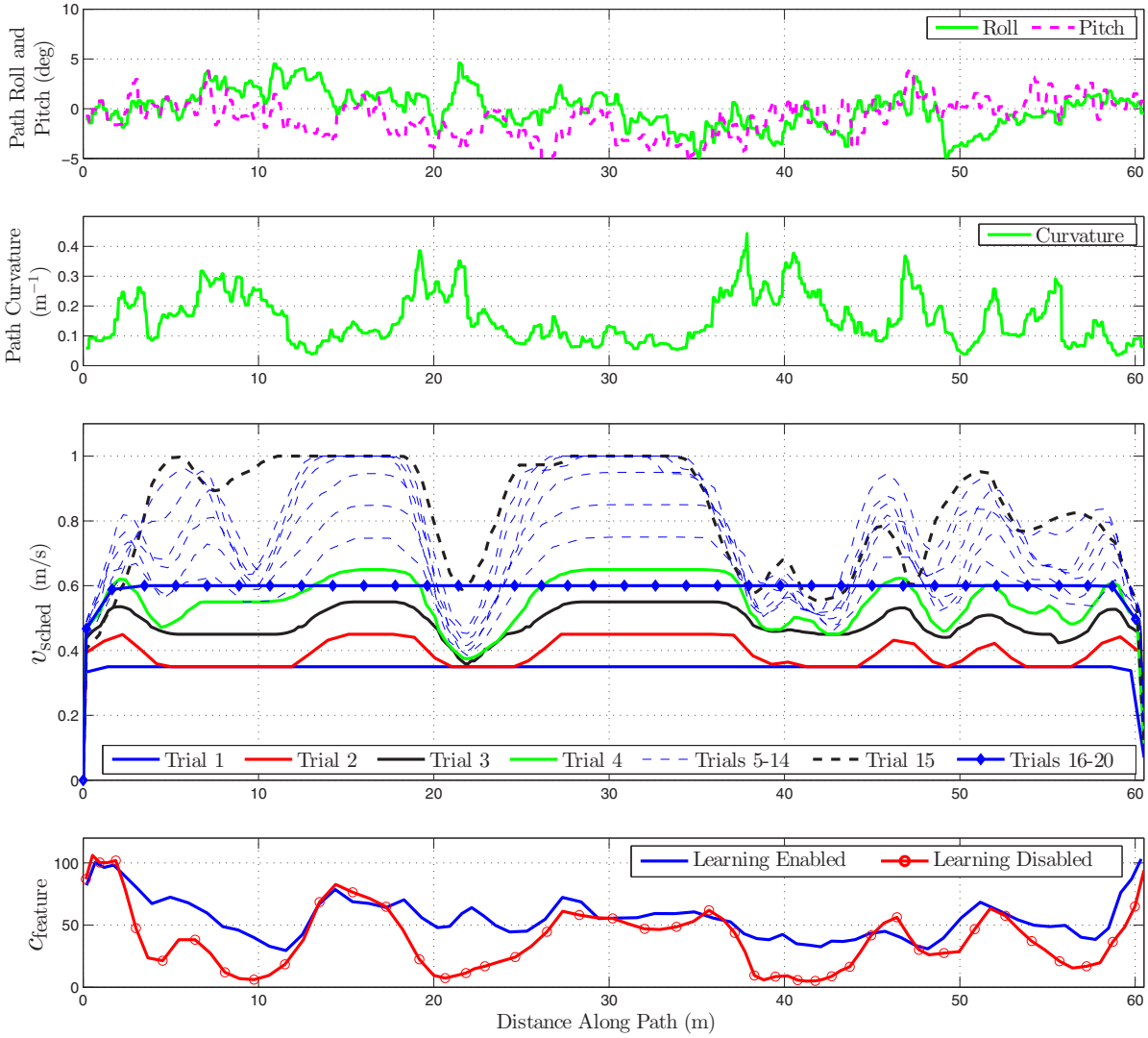

Figure 12. The test path for experiment 2 was mainly on level ground, but it included path curvatures up to 0.5 m^{-1}. Here we also show the scheduled speeds, $v^{(j)}_{\text{sched},i}$, for trials 1 through 20, and the VT&R matched feature counts, $c^{(j)}_{\text{feature},i}$, for trial 15.

path demonstrating the ability of the disturbance model to learn kinematics and dynamics of a significantly different mass and robot design[1] (Figure 16). Figure 17 shows plots of path characteristics, scheduled speed, and path-tracking errors vs. distance along the path. As in the second experiment, the speed scheduler tried to minimize the overall travel time and determined where along the path the system could tolerate higher speeds using experience from previous traversals. Over the course of the 10 trials, the LB-NMPC algorithm reduced the lateral and heading errors significantly while learning disturbances at speeds ranging from 0.5 to 1.2 m/s (Figure 18).

This last experiment highlighted the need for work on experience management and controller robustness. Between 85 and 90 m along the path, in all trials of experiment 3, the results showed a sharp change in path-tracking errors. For example, in trial 1, the VT&R state estimate produced a step-change in the lateral path-tracking error of ∼25 cm in a single time-step (Figure 17). In reality, the robot made no such movement. This artificial motion estimate was triggered by what Furgale and Barfoot (2010) called a "teach pass failure," resulting in a discontinuity in the state estimate during relocalization. In this case, the LB-NMPC

[1]Associated video at http://tiny.cc/RoverLearnsDisturbances

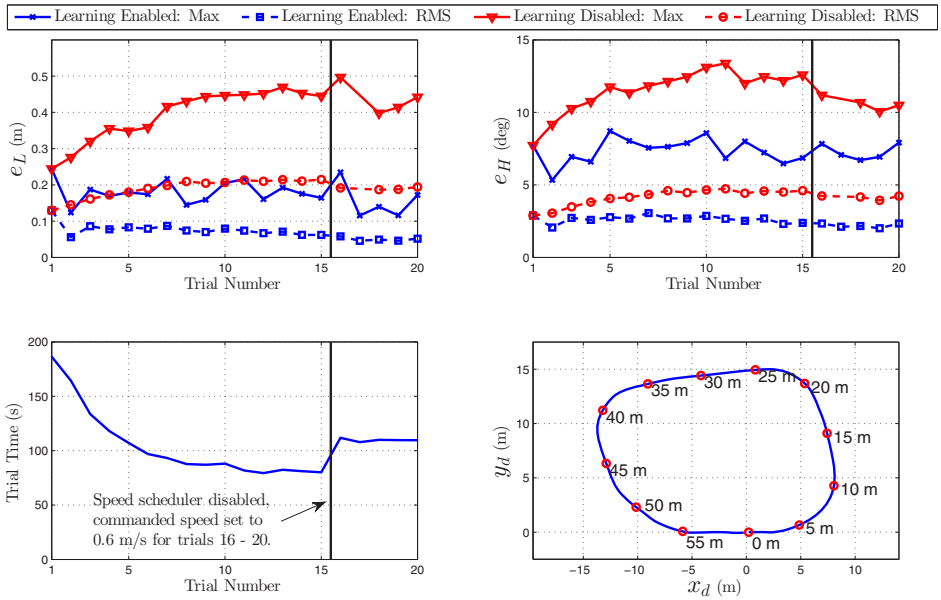

Figure 13. Here we show the reduction in maximum and root mean square (RMS) lateral and heading path-tracking errors vs. trial. Unlike the first experiment, the scheduled speeds for each trial were adjusted throughout the second experiment, resulting in a range of travel times when tracking the loop-shaped, 60-m-long path.

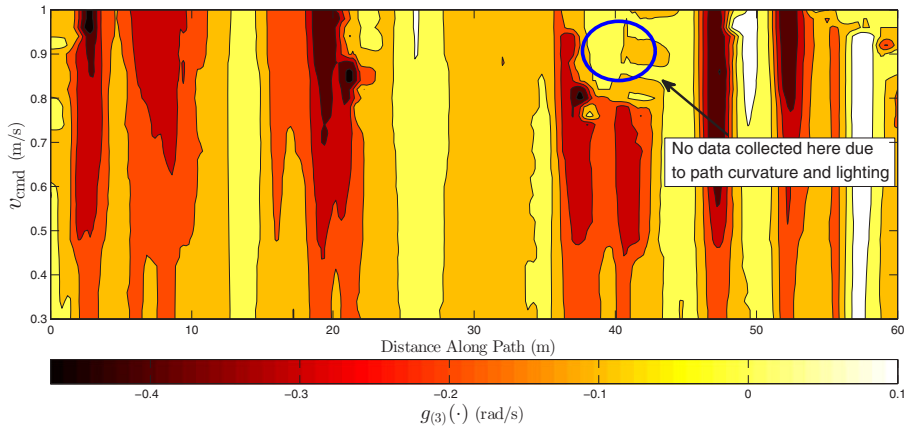

Figure 14. Here we show the learned values for the heading rate disturbance [i.e., the third element of $\mathbf{g}(\cdot)$] vs. commanded speed and distance along the path. Above 0.8 m/s, 40 m along the path (blue ellipse), there was very little data and the model was untrustworthy (Figure 15).

algorithm treated the side-step as a modeling error and learned to turn in anticipation of the (artificial) disturbance, thereby causing subsequent (real) path-tracking errors. While practical state estimation algorithms should avoid providing faulty estimates, the stakes are higher with learning algorithms that are capable of inadvertently incorporating such outlier measurements into the learned model and then acting on incorrect data. This is one motivation for our experience management scheme, which forgets experiences over time.

Figure 18 shows plots of the maximum and RMS path-tracking errors, and overall travel time vs. trial number. The LB-NMPC algorithm reduced the lateral and heading errors by more than 50% over the course of the 20 trials while learning disturbances at speeds ranging from 0.5 to 1.2 m/s. The maximum path-tracking errors (heading and lateral) in trials 5, 7, 9, and 10 occurred at the aforementioned section of the path between 85 and 90 m, where the localization system indicated an (artificial) disturbance.

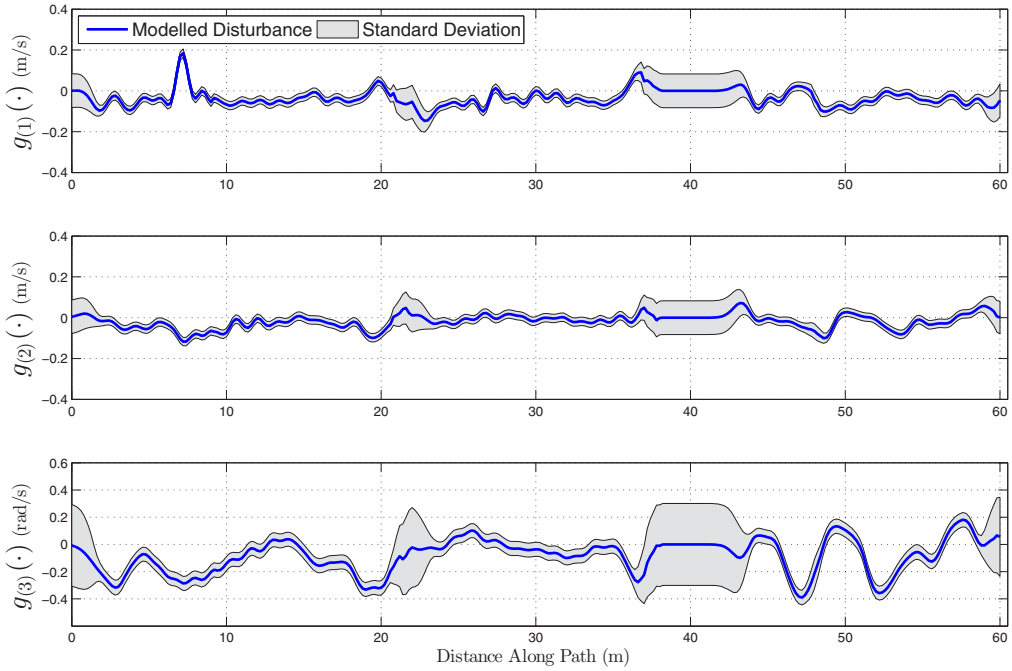

Figure 15. Modeled disturbances, $\mathbf{g}(\cdot) = (g_{(1)}(\cdot), g_{(2)}(\cdot), g_{(3)}(\cdot))$, for $v_{\text{cmd}} = 0.9$ m/s. With no experience above 0.8 m/s, 40 m along the path, the modeled disturbance is zero and relatively uncertain.

Figure 16. In experiment 3, we tested with a 600 kg, Ackermann-steered robot repeating a 100-m-long path. As in the previous experiments, the nominal model used by the LB-NMPC algorithm was a unicycle model, demonstrating the algorithm's ability to be applied to robots with significantly different designs.

6. DISCUSSION

6.1. Controller Robustness

In general, our LB-NMPC algorithm is initialized with a known nominal model and learns the discrepancies between the known model and the actual robot behavior. Therefore, by its very structure, the augmented process model used by our NMPC algorithm has varying levels of uncertainty while learning. Controller robustness, i.e., the capability of a controller to stabilize a system in spite of model uncertainty, is an open question for learning controllers in general (Schaal et al., 2010). In this paper, we do not explicitly consider the robustness of the controller but focus on the practical application of LB-NMPC to mobile robots. However, having established the effectiveness of the LB-NMPC algorithm at reducing control errors with few *a priori* assumptions, our future work will focus on techniques to leverage the covariance estimates provided by our GP-based model in a robust control framework. Ideally, the controller will automatically choose between conservativeness, when the model is relatively uncertain, and optimality, when the model is less uncertain.

6.2. Convergence Rates

Determination of convergence rates is also an open problem in model-based learning controllers (Nguyen-Tuong & Peters, 2011). Unlike techniques such as Iterative Learning Control (Ahn, Chen, & Moore, 2007; Bristow, Tharayil, & Alleyne, 2006), which assume identical initial conditions and desired trajectories for all trials in order to make claims on convergence rates, model-based learning controllers, such as the work illustrated in this paper, address a more

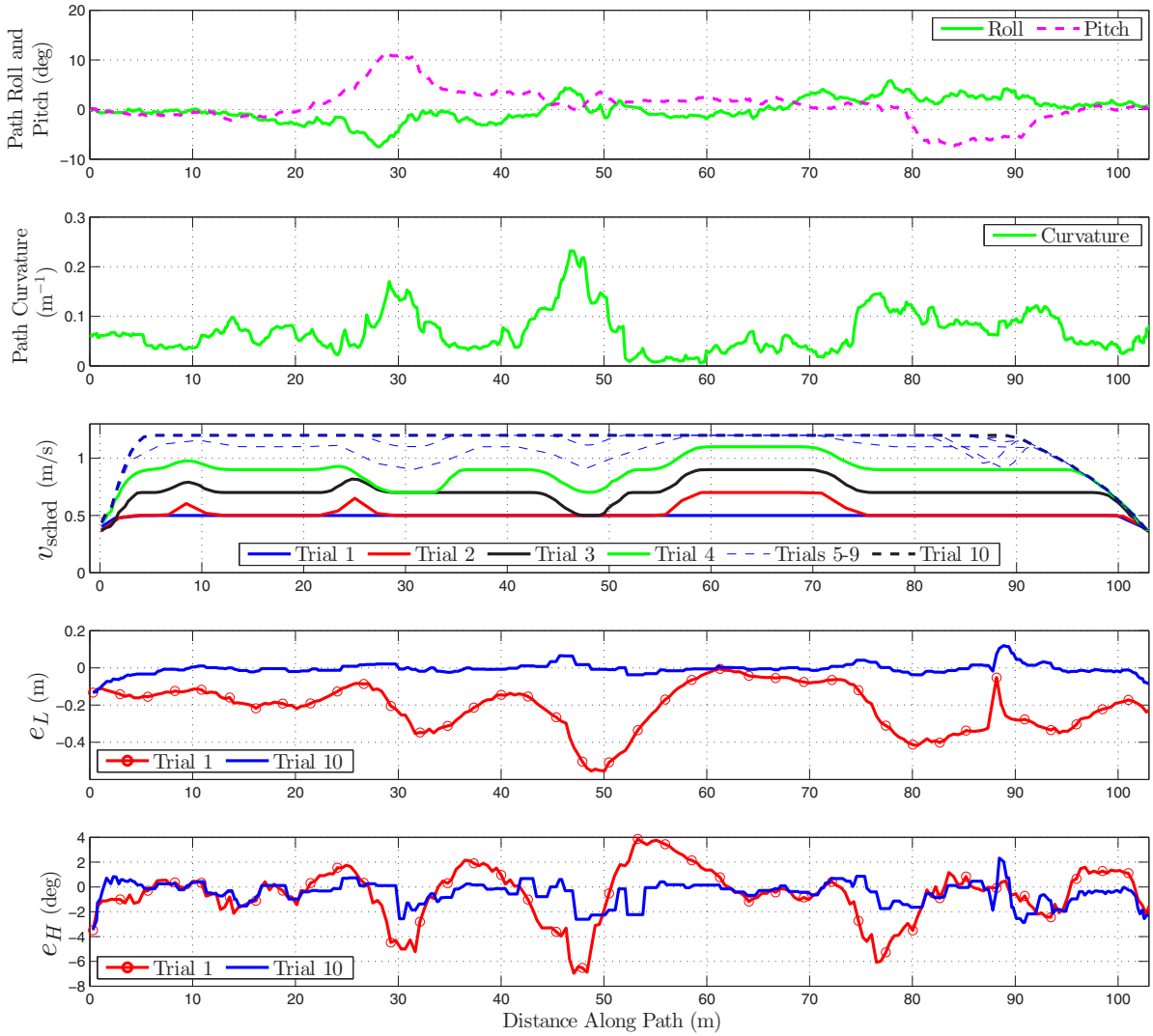

Figure 17. The path for the third experiment formed a large loop with turns at 30, 45, and 75 m along the path (Figure 18). As in experiment 2, the robot learned to drive the path at a range of speeds, from 0.5 to 1.2 m/s, generated by the speed scheduler. At around 50 m along the path, it took three trials before path-tracking errors were reduced sufficiently such that the scheduled speed could be increased.

general problem of trying to learn with arbitrary initial conditions, paths, and speed schedules. This enables a more flexible robot use since it is able to learn more than one path at more than one speed. However, it also presents essentially a sporadic approach to learning, in that it is not guaranteed when or if a state will be revisited for continued learning. Furthermore, convergence rates are complicated by the evolution of the environment caused by the robot's activity. For example, repeating the same path caused ruts to form, which resulted in a change in the disturbances affecting the nominal process model. This was also a motivation in using only the most recent observations (Section 4.3).

7. CONCLUSION

In summary, this paper presents a Learning-based Nonlinear Model Predictive Control (LB-NMPC) algorithm for a path-repeating, mobile robot negotiating large-scale, GPS-denied outdoor environments. The goal is to reduce path-tracking errors using real-world experience instead of pre-programming accurate analytical models of wheel-terrain interaction, terrain topology, or robot dynamics. The LB-NMPC controller is based on a fixed, simple process model and a learned disturbance model. Disturbances effectively represent measured discrepancies between the given nominal model and the observed system behavior. Disturbances

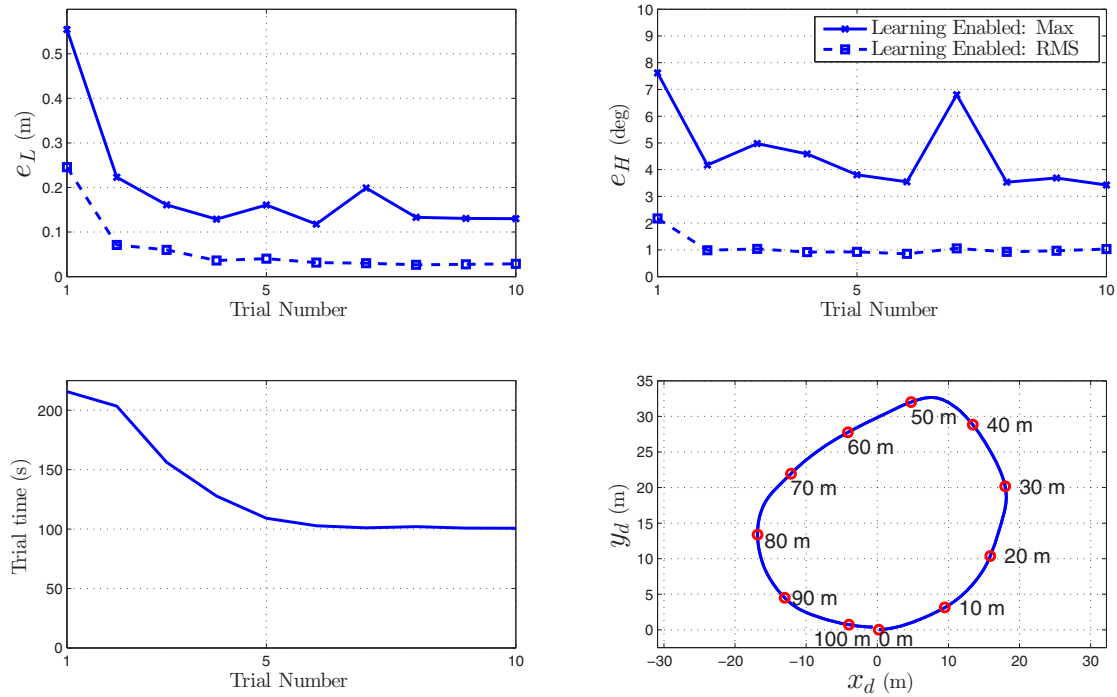

Figure 18. Over the course of 10 trials in experiment 3, the LB-NMPC algorithm reduced the lateral and heading path-tracking errors by over 50%, while simultaneously learning to drive at faster speeds around the loop-shaped, 100-m-long path. The desired speeds were provided by the automated speed scheduler (Section 4.2).

are modeled as a Gaussian process (GP) based on observations as a function of relevant variables such as the system state and input. Modeling the disturbances as a GP enables the algorithm to learn complex nonlinear model discrepancies and to generalize to novel situations. Localization for the controller is provided by an onboard, Visual Teach & Repeat mapping and navigation system. The paper also presents an experience-based speed scheduler that plans time-optimal schedules while guaranteeing low path-tracking errors and reliable localization.

Three experiments on three significantly different robots, including over 3 km of travel on challenging paths, demonstrated the system's ability to handle unmodeled terrain and robot dynamics, and also to interpolate and extrapolate from learned disturbances. In the second and third experiments, the experience-based speed scheduler addressed the classic exploration vs. exploitation tradeoff balancing speed, path-tracking errors, and localization reliability. The LB-NMPC approach proved to be flexible and effective at reducing path-tracking errors and increasing the reliability of the localization system. Even beginning with only the simple unicycle model, the algorithm was capable of being seamlessly deployed to multiple platforms where it learned to reduce vehicle- and trajectory-specific path-tracking errors using experience. However, robust stability is a largely unanswered question for state-of-the-art learning control algorithms. In this work, we only make use of the predicted mean value of disturbances. However, in future work, we plan to investigate robust learning control, leveraging the predicted disturbance uncertainty as well as the mean when making control decisions.

ACKNOWLEDGMENTS

This research was funded by the Ontario Ministry of Research and Innovations Early Researcher Award Program, by the Natural Sciences and Engineering Research Council of Canada (NSERC) through the NSERC Canadian Field Robotics Network (NCFRN), and by Clearpath Robotics.

APPENDIX: MULTIMEDIA DESCRIPTION

A multimedia extension has been prepared to accompany this work. The extension shows the results from experiment 3, where the learning-based controller is tested on a 600 kg Ackermann-steered robot and learns to reduce vehicle- and trajectory-specific path-tracking errors. The extension also shows the experience-based speed scheduler generating speed schedules for each trial. The video is available as a Supporting Information file in the online version of this article or at http://tiny.cc/RoverLearnsDisturbances.

REFERENCES

Abbeel, P., Quigley, M., & Ng, A. (2006). Using inaccurate models in reinforcement learning. In Proceedings of the 23rd International Conference on Machine Learning (pp. 1–8). Omni Press.

Ahn, H.-S., Chen, Y., & Moore, K. L. (2007). Iterative learning control: Brief survey and categorization. IEEE Transactions on Systems, Man, and Cybernetics: Applications and Reviews, 37(6), 1099–1121.

Angelova, A., Matthies, L., Helmick, D., & Perona, P. (2007). Learning and prediction of slip from visual information. Journal of Field Robotics, 24(3), 1205–231.

Aswani, A., Gonzalez, H., Shankar Sastry, S., & Tomlin, C. (2013). Provably safe and robust learning-based model predictive control. Automatica, 49, 1216–1226.

Bristow, D. A., Tharayil, M., & Alleyne, A. G. (2006). A survey of iterative learning control. IEEE Control Systems, 26(3), 96–114.

Cariou, C., Lenain, R., Thuilot, B., & Berducat, M. (2009). Automatic guidance of a four-wheel-steering mobile robot for accurate field operations. Journal of Field Robotics, 26(6-7), 504–518.

Deisenroth, M., Fox, D., & Rasmussen, C. (2014). Gaussian processes for data-efficient learning in robotics and control. IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(5), 1–20.

Diehl, M., Ferreau, H. J., & Haverbeke, N. (2009). Efficient numerical methods for nonlinear MPC and moving horizon estimation. In Lecture Notes in Control and Information Sciences, Nonlinear Model Predictive Control (vol. 384, pp. 391–417). Springer.

Furgale, P., & Barfoot, T. (2010). Visual teach and repeat for long-range Rover autonomy. Journal of Field Robotics, 27(5), 534–560.

Guillet, A., Lenain, R., & Thuilot, B. (2013). Off-road path tracking of a fleet of WMR with adaptive and predictive control. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 2855–2861).

Helmick, D., Angelova, A., & Matthies, L. (2009). Terrain adaptive navigation for planetary Rovers. Journal of Field Robotics, 26(4), 391–410.

Helmick, D., Roumeliotis, S., Cheng, Y., Clouse, D., Bajracharya, M., & Matthies, L. (2006). Slip-compensated path following for planetary exploration rovers. Advanced Robotics, 20(11), 1257–1280.

Howard, T., Green, C., & Kelly, A. (2009). Receding horizon model-predictive control for mobile robot navigation of intricate paths. In Proceedings of the 7th Annual Conference on Field and Service Robotics (pp. 69–78). Springer-Verlag Berlin Heidelberg.

Iagnemma, K., Kang, S., Shibly, H., & Dubowsky, S. (2004). Online terrain parameter estimation for wheeled mobile robots with application to planetary rovers. IEEE Transactions on Robotics, 20(5), 921–927.

Ishigami, G., Nagatani, K., & Yoshida, K. (2009). Slope traversal controls for planetary exploration rover on sandy terrain. Journal of Field Robotics, 26(3), 264–286.

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

Ko, J., Klein, D. J., Fox, D., & Haehnel, D. (2007). Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In Proceedings of the International Conference on Robotics and Automation (pp. 742–747).

Kocijan, J., Murray-Smith, R., Rasmussen, C., & Girard, A. (2004). Gaussian process model based predictive control. In Proceedings of the American Control Conference (vol. 3, pp. 2214–2219).

Kühne, F., Lages, W. F., & Silva, J. (2005). Mobile robot trajectory tracking using model predictive control. In Proceedings of the IEEE Latin-American Robotics Symposium (pp. 1–7).

Langson, W., Chryssochoos, I., Raković, S., & Mayne, D. Q. (2004). Robust model predictive control using tubes. Automatica, 40(1), 125–133.

Meier, F., Hennig, P., & Schaal, S. (2014). Efficient Bayesian local model learning for control. In Proceedings of the IEEE International Conference on Intelligent Robotics Systems (pp. 2244–2249).

Nguyen-Tuong, D., & Peters, J. (2011). Model learning for robot control: A survey. Cognitive Processing, 12(4), 319–340.

Nguyen-Tuong, D., Peters, J., & Seeger, M. (2009). Local Gaussian process regression for real time online model learning. Advances in Neural Information Processing Systems, 22, 1193–1200.

Nocedal, J., & Wright, S. (1999). Numerical Optimization (vol. 2). New York: Springer.

Ostafew, C., Collier, J., Schoellig, A. P., & Barfoot, T. (2014a). Speed Daemon: Experience-based mobile robot speed scheduler. In Proceedings of the Conference on Computer and Robot Vision (pp. 56–62). IEEE.

Ostafew, C., Schoellig, A. P., & Barfoot, T. (2013). Visual teach and repeat, repeat, repeat: Iterative learning control to improve mobile robot path tracking in challenging outdoor environments. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 176–181).

Ostafew, C., Schoellig, A. P., & Barfoot, T. (2014b). Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments. In Proceedings of the IEEE International Conference on Robotics and Automation (pp. 4029–4036).

Peters, S., & Iagnemma, K. (2008). Mobile robot path tracking of aggressive maneuvers on sloped terrain. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 242–247).

Quinonero-Candela, J., & Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. The Journal of Machine Learning Research, 6, 1939–1959.

Rasmussen, C. E. (2006). Gaussian processes for machine learning. Cambridge, MA: MIT Press.

Rasmussen, C. E., & Ghahramani, Z. (2002). Infinite mixtures of Gaussian process experts. Advances in Neural Information Processing Systems, 2, 881–888.

Rawlings, J. B., & Mayne, D. Q. (2009). Model predictive control: Theory and design. Nob Hill Publishing.

Schaal, S., & Atkeson, C. (2010). Learning control in robotics. IEEE Robotics & Automation Magazine, 17(2), 20–29.

Schoellig, A. P., Mueller, F., & D'Andrea, R. (2012). Optimization- based iterative learning for precise quadrocopter trajectory tracking. Autonomous Robots, 33, 103–127.

Seegmiller, N., Rogers-Marcovitz, F., Miller, G., & Kelly, A. (2013). Vehicle model identification by integrated prediction error minimization. The International Journal of Robotics Research, 32(8), 912–931.

Snelson, E., & Ghahramani, Z. (2007). Local and global sparse Gaussian process approximations. In Proceedings of the International Conference on Artificial Intelligence and Statistics (pp. 524–531). Journal of Machine Learning Research – Proceedings Track.

Stenning, B., McManus, C., & Barfoot, T. (2013). Planning using a network of reusable paths: A physical embodiment of a rapidly exploring random tree. Journal of Field Robotics, 30(6), 916–950.

Xie, F., & Fierro, R. (2008). First-state contractive model predictive control of nonholonomic mobile robots. In Proceedings of the American Control Conference (pp. 3494–3499). IEEE.