*Article*

The International Journal of
Robotics Research
2016, Vol. 35(13) 1547–1563

DOI: 10.1177/0278364916645661
ijr.sagepub.com

# Robust Constrained Learning-based NMPC enabling reliable mobile robot path tracking

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

**Abstract**

*This paper presents a Robust Constrained Learning-based Nonlinear Model Predictive Control (RC-LB-NMPC) algorithm for path-tracking in off-road terrain. For mobile robots, constraints may represent solid obstacles or localization limits. As a result, constraint satisfaction is required for safety. Constraint satisfaction is typically guaranteed through the use of accurate, a priori models or robust control. However, accurate models are generally not available for off-road operation. Furthermore, robust controllers are often conservative, since model uncertainty is not updated online. In this work our goal is to use learning to generate low-uncertainty, non-parametric models in situ. Based on these models, the predictive controller computes both linear and angular velocities in real-time, such that the robot drives at or near its capabilities while respecting path and localization constraints. Localization for the controller is provided by an on-board, vision-based mapping and navigation system enabling operation in large-scale, off-road environments. The paper presents experimental results, including over 5 km of travel by a 900 kg skid-steered robot at speeds of up to 2.0 m/s. The result is a robust, learning controller that provides safe, conservative control during initial trials when model uncertainty is high and converges to high-performance, optimal control during later trials when model uncertainty is reduced with experience.*

**Keywords**

Model predictive control, Gaussian process, robust control

## 1. Introduction

Model Predictive Control (MPC) is a powerful algorithm capable of computing optimal inputs while seamlessly addressing state and input constraints. At each time-step, the controller solves a finite-horizon optimization problem based on the current state and a model of the system (Rawlings and Mayne, 2009). For mobile robots, state constraints may represent physical obstacles or localization limits, and, as a result, constraint satisfaction is tantamount to safety (Figure 1). However, system models used in practice are often uncertain and constraint satisfaction is difficult to guarantee in general. For example, operation in off-road terrain frequently leads to modeling errors due to surface materials (e.g. snow, sand, grass), terrain topography (e.g. side-slopes, inclines), and complex robot dynamics (Ostafew et al., 2015a).

Robust Constrained MPC (RC-MPC) is an active area of research and endeavors to provide guarantees on constraint satisfaction when considering uncertain systems (Mayne, 2014). At each time-step, RC-MPC aims to identify inputs such that all plausible predicted trajectories satisfy the given constraints considering a fixed estimate of model uncertainty. However, such approaches are generally conservative since the models are not updated online.

In this paper, we investigate a Robust Constrained Learning-based Nonlinear Model Predictive Control (RC-LB-NMPC) algorithm for a path-repeating mobile robot operating in challenging outdoor terrain. Learning-based algorithms alleviate the need for significant engineering work on identifying and modeling all effects that a model-based controller may be required to mitigate (Nguyen-Tuong and Peters, 2011; Schaal and Atkeson, 2010). Moreover, learning-based algorithms allow for the system to act in anticipation of disturbances.

The algorithm uses a fixed, simple robot model and a learned, nonparametric disturbance model. Disturbances represent measured discrepancies between the *a priori* model and observed system behavior. Essentially, the algorithm begins with a simple process model and high model uncertainty for the first trial and learns an accurate, low-uncertainty model over successive trials. Disturbances are modeled as a Gaussian process (GP) (Rasmussen, 2006)

University of Toronto Institute for Aerospace Studies

**Corresponding author:**
Chris J. Ostafew, University of Toronto Institute for Aerospace Studies, 4925 Dufferin St, Toronto, Ontario, Canada, M3H 5T6.
Email: chris.ostafew@mail.utoronto.ca

**Fig. 1.** State constraints for mobile robots frequently represent physical obstacles, thus constraint satisfaction is required for safety. We present a Robust Constrained Learning-based Nonlinear MPC algorithm to guarantee constraint satisfaction while improving performance through learning. The algorithm is tested on a 900 kg Clearpath Grizzly traveling up to 2.0 m/s on off-road paths with tight constraints.

based on previous experience as a function of state, input, and other relevant variables. We use a Sigma-Point Transform (Julier and Uhlmann, 2004) to efficiently compute the mean and variance of predicted state sequences given the two-component, learned model. We then apply restricted constraints to the predicted sequence such that the $3\sigma$ confidence region is contained within the desired constraints. In this way, the predictive controller automatically computes low-speed, conservative linear and angular inputs during initial trials when model uncertainty is high, and high-performance inputs during later trials when model uncertainty is reduced through learning (Figure 2). Finally, we present extensive experimental results, including over 5 km of travel by a 900 kg skid-steered robot at speeds up to 2.0 m/s.

Localization for the controller is provided by an onboard, Visual Teach & Repeat (VT&R) mapping and navigation algorithm enabling operation in large-scale, off-road environments (Furgale and Barfoot, 2010; Stenning et al., 2013). In the first operational phase, the teach phase, the robot is manually 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). 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, generating feedback for a path-tracking controller. 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.

This work differs from traditional constrained NMPC algorithms in two main ways. Firstly, in traditional implementations, the process model is specified *a priori* and remains unchanged during operation. In this work, we augment the process model with a learned disturbance model in order to predict the mean and uncertainty of effects not captured by the fixed process model. Secondly, constrained NMPC approaches do not typically account for model uncertainty. In this work, we apply robust constraints in real-time considering the learned uncertainty. We provide robust constraint satisfaction when uncertainty is high and increased performance as uncertainty is reduced through learning. This paper is a major extension of previous work. We have previously investigated unconstrained LB-NMPC (Ostafew et al., 2015a) and unconstrained Min–Max LB-NMPC, where control inputs are optimized based on worst-case scenarios (Ostafew et al., 2015b). However, the Min–Max approach does not provide tuning knobs to directly specify tracking error constraints (the only tuning inputs are the MPC weights). Significant additions include robust constraints considering the learned uncertainty and presentation of the necessary nonlinear techniques used to solve the resulting optimization problem. 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 RC-LB-NMPC algorithm and implementation details. Section 5 presents experimental results. Finally, Sections 6 and 7 present a discussion and conclusion.

## 2. Related work

This work is aimed at achieving robust constrained, high-performance path-tracking in spite of unknown disturbances. In the following, we provide a brief review of approaches involving constrained and unconstrained MPC (linear and nonlinear). Then we provide a background on learning control approaches.

### *2.1. Model Predictive Control*

Model Predictive Control (MPC) has received a significant amount of attention in recent years due to its ability to handle complex linear and nonlinear systems with state and input constraints (Mayne, 2014; Rawlings and Mayne, 2009). It has been proposed for mobile robots (Howard et al., 2009; Klančar and Škrjanc, 2007; Krenn et al., 2013; Kühne et al., 2005; Peters and Iagnemma, 2008), unmanned aerial vehicles (Kumar and Michael, 2012; Mueller and

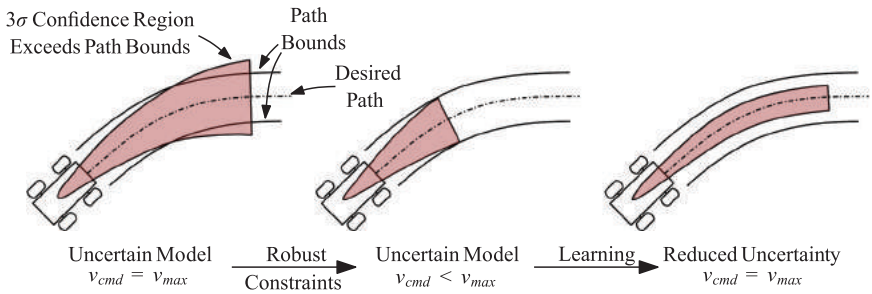


**Fig. 2.** The predictive controller optimizes both the linear and angular velocities over a prediction horizon such that the $3\sigma$ confidence region is contained within the path bounds. (Left) At full speed with an uncertain model, the controller cannot guarantee constraint satisfaction throughout the prediction horizon. (Center) As a result, the controller selects slower, more conservative inputs. (Right) Over successive trials, learning reduces model error and uncertainty, and the controller is able to guarantee constraint satisfaction at the maximum allowed speed.

D'Andrea, 2013), and humanoid robots (Erez et al., 2013; Gouaillier et al., 2010; Lafaye et al., 2014). However, in each of these examples, the proposed algorithms do not use learned models or account for model uncertainty when applying constraints. As a result, they either require high-accuracy models representing the system and environmental interactions, or risk violating the given constraints.

Tube MPC is a robust constrained algorithm that accounts for model uncertainty when considering constraints (Langson et al., 2004; Mayne et al., 2011). The algorithm applies restricted constraints to nominal predictions such that all possible state sequences based on the given uncertainty satisfy the constraints. Recently, González et al. (2011) and Farrokhsiar et al. (2013) apply Tube MPC to mobile robots. However, unlike their work, our robust constrained algorithm is based on a fixed *a priori* model and a learned, nonparametric disturbance model. This allows us to apply robust constraints while improving performance through learning. Aswani et al. (2013) propose Learning-based Tube-MPC with a bounded, learned model and use computationally expensive reachability analysis to compute restricted constraints. In this work, state sequences are predicted using an efficient Sigma-Point Transform, and restricted constraints are defined such that the predicted $3\sigma$ confidence region is contained within the true constraints. As a result, our algorithm efficiently computes and applies robust constraints in real time while also improving with experience.

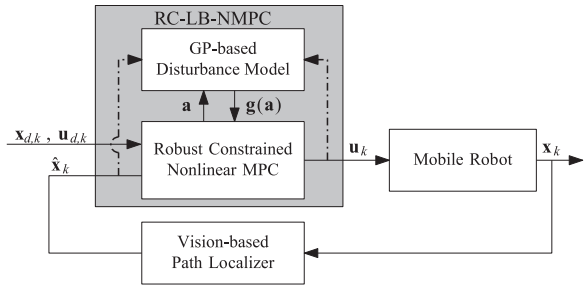


**Fig. 3.** The RC-LB-NMPC algorithm is composed of two parts: (i) the robust constrained, path-tracking NMPC algorithm based on an *a priori* process model, and (ii) the GP-based disturbance model. During the first trial, the RC-NMPC algorithm relies solely on the *a priori* process model to follow the desired path $\mathbf{x}_d$, considering the nominal inputs $\mathbf{u}_d$. In later trials, the RC-NMPC algorithm uses the disturbance model as a correction to the *a priori* model at states $\mathbf{a}$, to be defined in Section 3.1. Dashed lines indicate that the signals update the model. In practice, our overall system combines the RC-LB-NMPC algorithm with vision-based localization (Furgale and Barfoot, 2010) for off-road path-tracking.

## 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. Learned models enable the controller to act in anticipation of disturbances. Learned models are commonly modeled as GPs, Locally Weighted Projection Regression, or Support Vector Regression (Sigaud et al., 2011). 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). Modeling a process model as a GP for learning has been previously proposed. For example, Nguyen-Tuong et al. (2008) and Meier et al. (2014) model the inverse dynamics of manipulator arms as a GP based on previous input–output data. Unlike these examples, we model the mean and uncertainty of forward dynamics and propose an RC-NMPC algorithm that seamlessly incorporates the GP model, and robust state and input constraints. Iterative Learning Control (ILC) is another common approach to learning from experience for the control of high-performance robots (Hehn and D'Andrea, 2014; Moore, 2012; Ostafew et al., 2013; Schoellig et al., 2012). ILC constructs an acausal feedforward signal over sequential trials using error information from any previous trial. However, ILC-based learning controllers are only capable

of incorporating state and input constraints when constructing the feedforward signal. On the other hand, our model-based RC-LB-NMPC algorithm uses the learned model at each time-step, enabling generalization from learned experiences and real-time, robust constraint satisfaction.

## 3. Mathematical formulation

At a given sample time, NMPC 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. The entire process is repeated at the next sample time for the new system state. In this section, we first present our overall robust constrained NMPC algorithm (Figure 3). We then present our methods of predicting uncertain state sequences and estimating disturbances.

### 3.1. Robust Constrained Nonlinear Model Predictive Control

Consider the following nonlinear, state-space system:

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

with observable system state $\mathbf{x}_k \sim \mathcal{N}(\bar{\mathbf{x}}_k, \mathbf{\Sigma}_k)$, $\mathbf{x}_k \in \mathbb{R}^n$ and control input, $\mathbf{u}_k \in \mathbb{R}^m$, both at time $k$. The true system, $\mathbf{f}_{\text{true}}(\mathbf{x}_k, \mathbf{u}_k)$, is not known exactly and is approximated by the sum of an *a priori* model and an experience-based, learned model

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

with disturbance dependency $\mathbf{a}_k \in \mathbb{R}^p$. The models $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$ are nonlinear process models: $\mathbf{f}(\cdot)$ is a known process model representing our knowledge of $\mathbf{f}_{\text{true}}(\cdot)$, and $\mathbf{g}(\cdot)$ is an (initially unknown) disturbance model representing discrepancies between the *a priori* model and the actual system behavior. Disturbances are modeled as a GP (Section 3.3), and thus $\mathbf{g}(\cdot)$ is normally distributed $\mathbf{g}(\cdot) \sim \mathcal{N}(\boldsymbol{\mu}(\cdot), \mathbf{\Sigma}_{\text{gp}}(\cdot))$. Previously, we showed that $\mathbf{g}(\cdot)$ could be used to learn higher-order dynamics by including historic states in the disturbance dependency (Ostafew et al., 2015a). However, for simplicity, we assume for now that $\mathbf{a}_k = (\bar{\mathbf{x}}_k, \mathbf{u}_k)$.

As previously mentioned, the goal of NMPC is to find a set of controls that optimizes the predicted plant behavior over a given horizon. We define the cost function to be minimized over the next $K$ time-steps as

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

where $\mathbf{Q} \in \mathbb{R}^{Kn \times Kn}$ is positive semi-definite, $\mathbf{R} \in \mathbb{R}^{Km \times Km}$ is positive definite, $\mathbf{x}_d = (\mathbf{x}_{d,k+1}, \ldots, \mathbf{x}_{d,k+K})$ is a sequence of desired states, $\mathbf{x} = (\mathbf{x}_{k+1}, \ldots, \mathbf{x}_{k+K})$ is a sequence of uncertain predicted states, $\bar{\mathbf{x}}$ is the sequence of mean values based on $\mathbf{x}$, $\mathbf{u}_d = (\mathbf{u}_{d,k}, \ldots, \mathbf{u}_{d,k+K-1})$ is a sequence of desired inputs, and $\mathbf{u} = (\mathbf{u}_k, \ldots, \mathbf{u}_{k+K-1})$ is a sequence of inputs. In this work, a sequence of predicted states, $\mathbf{x}$, is iteratively computed using a Sigma-Point Transform (Section 3.2) considering the normally distributed state estimate as provided by the vision-based path localizer, $\hat{\mathbf{x}}_k$, a sequence of control inputs, $\mathbf{u}$, and the learned model (2). For mobile robots, desired inputs can be used to schedule travel speed prior to driving according to knowledge of place-specific path characteristics such as roughness or localization reliability.

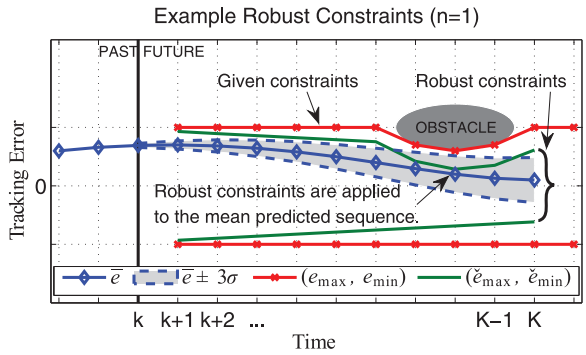


**Fig. 4.** Here we show a 1D example of the computation of robust constraints, $\check{e}_{\max,k}$ and $\check{e}_{\min,k}$, based on the given constraints, $e_{\max,k}$ and $e_{\min,k}$, and the predicted uncertainty sequence, $\sigma_k$. As the predicted state becomes more uncertain, the constraints are increasingly restricted in order to guarantee that the true state, contained within the $3\sigma$ confidence window, satisfies the given constraint.

We also define $2Kn$ probabilistic state constraints representing upper and lower tracking limits

$$\left.\begin{aligned} p(\mathbf{e}_{\max,k+i} - \mathbf{e}_{k+i} > \mathbf{0}) > l \\ p(\mathbf{e}_{k+i} - \mathbf{e}_{\min,k+i} > \mathbf{0}) > l \end{aligned}\right\} \quad i = 1 \ldots K \tag{4}$$

where $\mathbf{e}_{k+i} = \mathbf{x}_{d,k+i} - \mathbf{x}_{k+i}$ is the predicted tracking error, $\mathbf{e}_{\max,k+i}$ and $\mathbf{e}_{\min,k+i}$, are upper and lower tracking limits, respectively, $p(A)$ represents the probability of the event $A$, $l$ is a desired confidence level, and the inequalities are evaluated component-wise. We then compute robust deterministic limits, $\check{\mathbf{e}}_{\max,k+i} = \mathbf{e}_{\max,k+i} - \alpha\boldsymbol{\sigma}_{k+i}$ and $\check{\mathbf{e}}_{\min,k+i} = \mathbf{e}_{\min,k+i} + \alpha\boldsymbol{\sigma}_{k+i}$, given the predicted uncertainty, $\boldsymbol{\sigma}_{k+i} = (\sqrt{\mathbf{\Sigma}_{k+i}(1,1)}, \ldots, \sqrt{\mathbf{\Sigma}_{k+i}(n,n)})$ and apply them to the mean predicted states (Figure 4)

$$\check{\mathbf{e}}_{\max,k+i} > \bar{\mathbf{e}}_{k+i} > \check{\mathbf{e}}_{\min,k+i}, \quad i = 1 \ldots K \tag{5}$$

where $\bar{\mathbf{e}}_{k+i} = \mathbf{x}_{d,k+i} - \bar{\mathbf{x}}_{k+i}$ and the inequalities are evaluated component-wise. Typically, $\alpha = 3$ is chosen, representing a confidence level of approximately 0.997. In the case that the predicted uncertainty is larger than the given tracking limits, the system is set to stop and request a user input. For example, if the vision-based localization becomes lost, the state uncertainty will likely exceed the given constraints.

In addition, we consider $2Km$ actuator constraints

$$\mathbf{u}_{\max,k+i} > \mathbf{u}_{k+i} > \mathbf{u}_{\min,k+i}, \quad i = 0 \ldots K-1 \tag{6}$$

where $\mathbf{u}_{\max,k+i}$ and $\mathbf{u}_{\min,k+i}$ are defined by the actuator limits and the inequalities are evaluated componentwise. Finally, we define the complete set of constraints, $c_i(\bar{\mathbf{x}}, \mathbf{u}) > 0, i = 1 \ldots 2K(n+m)$, composed of both robust state (5) and input constraints (6).

Considering the current estimated system state, $\hat{\mathbf{x}}_k$, the process model, the constraints, and the cost function, we define the following constrained optimization problem

$$\{\mathbf{x}_{\text{opt}}, \mathbf{u}_{\text{opt}}\} = \arg\min_{\mathbf{x},\mathbf{u}} J(\bar{\mathbf{x}}, \mathbf{u}) \tag{7a}$$

$$\text{subject to} \quad \bar{\mathbf{x}}_{k+i+1} = \mathbf{f}(\bar{\mathbf{x}}_{k+i}, \mathbf{u}_{k+i}) + \mathbf{g}(\mathbf{a}_{k+i}), \quad i = 0 \ldots K-1, \tag{7b}$$

$$c_i(\bar{\mathbf{x}}, \mathbf{u}) > 0, \quad i = 1 \ldots n_c \tag{7c}$$

where the equality constraints are used to enforce the process model, and $n_c = 2K(n+m)$. We approximate the inequality constraints by introducing a slack variable, $\mathbf{w} = (w_1, \ldots, w_{n_c})$, and a logarithmic barrier term (Boyd and Vandenberghe, 2004)

$$\{\mathbf{x}_{\text{opt}}, \mathbf{u}_{\text{opt}}, \mathbf{w}_{\text{opt}}\} = \arg\min_{\mathbf{x},\mathbf{u},\mathbf{w}} J(\bar{\mathbf{x}}, \mathbf{u}) - \mu \sum_{i=1}^{n_c} \log(w_i) \tag{8a}$$

$$\text{subject to} \quad \bar{\mathbf{x}}_{k+i+1} = \mathbf{f}(\bar{\mathbf{x}}_{k+i}, \mathbf{u}_{k+i}) + \mathbf{g}(\mathbf{a}_{k+i}) \quad i = 0 \ldots K-1 \tag{8b}$$

$$c_i(\bar{\mathbf{x}}, \mathbf{u}) - w_i = 0, \quad i = 1 \ldots n_c \tag{8c}$$

where $\mu$ is a small positive scalar adjusted towards zero throughout the optimization process. Finally, we use Lagrange methods (Boyd and Vandenberghe, 2004) to solve (8) and define the Lagrangian $L(\mathbf{s})$

$$\begin{aligned} L(\mathbf{s}) = {} & J(\bar{\mathbf{x}}, \mathbf{u}) - \mu \sum_{i=1}^{n_c} \log(w_i) \\ & - \sum_{i=0}^{K-1} \boldsymbol{\lambda}_i^{\mathrm{T}} (\bar{\mathbf{x}}_{k+i+1} - \mathbf{f}(\bar{\mathbf{x}}_{k+i}, \mathbf{u}_{k+i}) - \mathbf{g}(\mathbf{a}_{k+i})) \\ & - \sum_{i=1}^{n_c} \gamma_i (c_i(\bar{\mathbf{x}}, \mathbf{u}) - w_i) \end{aligned} \tag{9}$$

where $\boldsymbol{\lambda}_i \in \mathbb{R}^n$, $\mathbf{s} = (\bar{\mathbf{x}}, \mathbf{u}, \mathbf{w}, \boldsymbol{\lambda}, \boldsymbol{\gamma})$, $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_0, \ldots, \boldsymbol{\lambda}_{K-1})$, and $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_{n_c})$. Considering the necessary condition for optimality, $\nabla L(\mathbf{s}) = \mathbf{0}$, we use Newton's method and iteratively linearize about an initial guess, $\mathbf{s} = \tilde{\mathbf{s}} + \delta\mathbf{s}$

$$\nabla L(\mathbf{s}) \approx \nabla L(\tilde{\mathbf{s}}) + \left.\frac{\partial \nabla L(\mathbf{s})}{\partial \mathbf{s}}\right|_{\tilde{\mathbf{s}}} \delta\mathbf{s} \tag{10}$$

solve for the value of $\delta\mathbf{s}$ such that

$$\nabla L(\tilde{\mathbf{s}}) + \left.\frac{\partial \nabla L(\mathbf{s})}{\partial \mathbf{s}}\right|_{\tilde{\mathbf{s}}} \delta\mathbf{s} = \mathbf{0} \tag{11}$$

**Algorithm 1:** RC-LB-NMPC

**Data**: $\mathbf{x}_d$, $\hat{\mathbf{x}}_k$, and $\mathbf{s}_{\text{init}}$
**Result**: $\mathbf{x}_{\text{opt}}$,$\mathbf{u}_{\text{opt}}$,$\mathbf{w}_{\text{opt}}$
initialization: $\tilde{\mathbf{s}} = \mathbf{s}_{\text{init}}$;
**while** $\|\delta\mathbf{s}\| > \eta$ **do**
  Compute state sequence $\mathbf{x}$ given the current state estimate $\hat{\mathbf{x}}_k$, $\tilde{\mathbf{s}}$ and (2);
  Compute robust constraints;
  Linearize $\nabla L(\mathbf{s}) = \mathbf{0}$ around $\tilde{\mathbf{s}}$ and solve for $\delta\mathbf{s}$;
  Update $\tilde{\mathbf{s}}$, $\tilde{\mathbf{s}} \leftarrow \tilde{\mathbf{s}} + \beta\delta\mathbf{s}$;

and compute an updated value for $\tilde{\mathbf{s}}$

$$\tilde{\mathbf{s}} \leftarrow \tilde{\mathbf{s}} + \beta\, \delta\mathbf{s} \tag{12}$$

where $0.1 \leq \beta \leq 1.0$ is selected at each iteration such that $w_i > 0$, $i = 1 \ldots n_c$. A good initial guess for $\tilde{\mathbf{s}}$ can be drawn from the previous time-step. After iterating to convergence (Algorithm 1), 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.

### *3.2. Predicting uncertain trajectories*

Since the state $\mathbf{x}_k$ is normally-distributed and (2) is nonlinear, we use a Sigma-Point Transform (Julier and Uhlmann, 2004) to iteratively predict state sequences (Figure 5) given a sequence of control inputs $\mathbf{u}$ and an estimate of the current state from the localizer, $\hat{\mathbf{x}}_k$. To predict $\mathbf{x}_{k+i+1}$, $i = 0 \ldots K$–1, we define a state $\mathbf{z}_i = (\bar{\mathbf{x}}_{k+i}, \boldsymbol{\mu}(\mathbf{a}_{k+i})) \in \mathbb{R}^{2n}$ representing the mean state and disturbance at time $k+i$ with uncertainty, $\mathbf{P}_i = \text{diag}(\boldsymbol{\Sigma}_{k+i}, \boldsymbol{\Sigma}_{\text{gp}}(\mathbf{a}_{k+i}))$. For $i = 0$, $\mathbf{z}_0$ is based on the current state estimate, $\hat{\mathbf{x}}_k$, otherwise $\mathbf{z}_i$ is based on a predicted state, $\mathbf{x}_{k+i}$. We compute $4n+1$ sigma points, $\mathcal{Z}_{i,j} = (\mathcal{X}_{i,j}, \mathcal{M}_{i,j})$, where $\mathcal{X}_{i,j}$ and $\mathcal{M}_{i,j}$ are the sigma points of $\mathbf{x}_{k+i}$ and $\boldsymbol{\mu}(\bar{\mathbf{x}}_{k+i}, \mathbf{u}_{k+i})$, respectively

$$\mathcal{Z}_{i,0} = \mathbf{z}_i \tag{13}$$

$$\mathcal{Z}_{i,j} = \mathbf{z}_i + \sqrt{2n+\kappa}\, \text{col}_j\, \mathbf{S}_i, \; j = 1 \ldots 2n \tag{14}$$

$$\mathcal{Z}_{i,j+2n} = \mathbf{z}_i - \sqrt{2n+\kappa}\, \text{col}_j\, \mathbf{S}_i, \; j = 1 \ldots 2n \tag{15}$$

where $\mathbf{S}_i\mathbf{S}_i^{\mathrm{T}} = \mathbf{P}_i$ with $\mathbf{S}_i \in \mathbb{R}^{2n \times 2n}$ derived from the Cholesky decomposition of $\mathbf{P}_i$, $\text{col}_j\, \mathbf{S}_i$ is the $j$th column of $\mathbf{S}_i$, and $\kappa$ is a tuning parameter. The sigma points are then passed through the nonlinear model

$$\mathcal{X}_{i+1,j} = \mathbf{f}(\mathcal{X}_{i,j}, \mathbf{u}_i) + \mathcal{M}_{i,j}, \; j = 0 \ldots 4n \tag{16}$$

where $\mathbf{f}(\cdot)$ is our *a priori* model. We recombine the sigma points into the predicted mean and uncertainty

$$\bar{\mathbf{x}}_{k+i+1} = \frac{1}{2n+\kappa} \left( \kappa \mathcal{X}_{i+1,0} + \frac{1}{2} \sum_{j=1}^{4n} \mathcal{X}_{i+1,j} \right) \tag{17}$$

$$\boldsymbol{\Sigma}_{k+i+1} = \frac{1}{2n+\kappa} \left( \kappa (\mathcal{X}_{i+1,0} - \bar{\mathbf{x}}_{k+i+1})(\mathcal{X}_{i+1,0} - \bar{\mathbf{x}}_{k+i+1})^{\mathrm{T}} + \frac{1}{2} \sum_{j=1}^{4n} (\mathcal{X}_{i+1,j} - \bar{\mathbf{x}}_{k+i+1})(\mathcal{X}_{i+1,j} - \bar{\mathbf{x}}_{k+i+1})^{\mathrm{T}} \right). \tag{18}$$

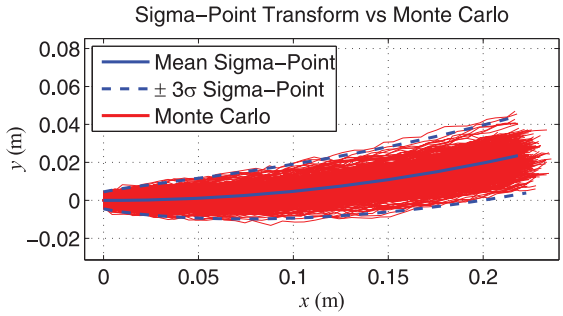


**Fig. 5.** Here we show the mean and lateral $3\sigma$ boundaries of a predicted sequence $\mathbf{x}$. Since our model uncertainty is normally distributed, we use a Sigma-Point Transform to efficiently compute the mean and uncertainty of predicted state sequences.

This process is repeated $K$ times, until the complete sequence $\mathbf{x}$ is generated. In this way, the $3\sigma$ confidence region accounts for uncertainty arising from both localization and modeling.

### 3.3. Gaussian process disturbance model

We model the disturbance $\mathbf{g}(\cdot)$ as a GP, which is a function of a disturbance dependency $\mathbf{a}$. Since we provided a detailed explanation of the learned model in previous work (Ostafew et al., 2015a) here we provide only a high-level sketch. The learned model depends on disturbance observations 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, the disturbance dependency $\mathbf{a}_{k-1}$ and the control input $\mathbf{u}_{k-1}$ to isolate (2) for $\hat{\mathbf{g}}(\mathbf{a}_{k-1})$

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

The resulting data pair, $\{\mathbf{a}_{k-1}, \hat{\mathbf{g}}(\mathbf{a}_{k-1})\}$, represents an individual experience. For trial $j$, we collect $N^{(j)}$ data pairs, where $N^{(j)}$ is the number of time-steps it took to travel the length of the path. At the end of each trial, we add newly collected pairs to a large dataset, $\mathcal{D}$, with generally $N$ data pairs, and drop the time-stamp, so that when referring to $\mathbf{a}_{\mathcal{D},i}$ or $\hat{\mathbf{g}}_{\mathcal{D},i}$, we mean the $i$th pair of data in the set $\mathcal{D}$.

In this work, we train a separate GP for each dimension in $\mathbf{g}(\cdot) \in \mathbb{R}^n$ to model disturbances affecting the *a priori* model. 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 GP

$$\hat{g}(\mathbf{a}_{\mathcal{D},i}) \sim \mathcal{GP}\left(0, k(\mathbf{a}_{\mathcal{D},i}, \mathbf{a}_{\mathcal{D},i})\right) \tag{20}$$

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, $\sigma_n^2$, 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

$$\begin{bmatrix} \hat{\mathbf{g}} \\ g(\mathbf{a}_k) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{k}(\mathbf{a}_k)^{\mathrm{T}} \\ \mathbf{k}(\mathbf{a}_k) & k(\mathbf{a}_k, \mathbf{a}_k) \end{bmatrix}\right) \tag{21}$$

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)

$$k(\mathbf{a}_i, \mathbf{a}_j) = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{a}_i - \mathbf{a}_j)^{\mathrm{T}} \mathbf{M}^{-2}(\mathbf{a}_i - \mathbf{a}_j)\right) + \sigma_n^2 \delta_{ij} \tag{22}$$

where $\delta_{ij}$ is the Kronecker delta: that is, 1 if and only if $i = j$ and 0 otherwise, and the constants $\mathbf{M}$, $\sigma_f$, and $\sigma_n$ 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 (10). In our implementation with $\mathbf{a}_k \in \mathbb{R}^p$, the constant $\mathbf{M}$ is a diagonal matrix, $\mathbf{M} = \mathrm{diag}(\mathbf{m})$, $\mathbf{m} \in \mathbb{R}^p$, representing the relevance of each component in $\mathbf{a}_k$, while the constants, $\sigma_f^2$ and $\sigma_n^2$, 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

$$g(\mathbf{a}_k) \,|\hat{\mathbf{g}} \sim \mathcal{N}\left(\mathbf{k}(\mathbf{a}_k)\,\mathbf{K}^{-1}\hat{\mathbf{g}},\; k(\mathbf{a}_k, \mathbf{a}_k) - \mathbf{k}(\mathbf{a}_k)\,\mathbf{K}^{-1}\mathbf{k}(\mathbf{a}_k)^{\mathrm{T}}\right). \tag{23}$$

Unlike our previous unconstrained LB-NMPC algorithm (Ostafew et al., 2015a), which used only the predicted mean of a disturbance, here we make use of both the predicted mean and variance. As previously mentioned (Section 3.2), the variance of the disturbance is used to predict state sequences with mean and uncertainty given a sequence of control inputs. Then, as detailed in Section 3.1, the predicted uncertainty is used to compute robust constraints in order to provide guarantees on constraint satisfaction. Finally, we include further detail on the storage and retrieval of observations for online operation in Section 4.2.

### 3.4. Gaussian process hyperparameter selection

Solving for the optimal hyperparameters, $\mathbf{M}$, $\sigma_f^2$, and $\sigma_n^2$, 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., experience from previous trials). For the first trial, when the robot has no experience, the predicted disturbance is mean zero with relatively high uncertainty. Given a set of experiences, we find the optimal hyperparameters offline by maximizing the log marginal likelihood

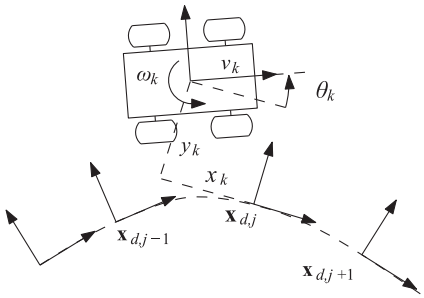


**Fig. 6.** Definition of the robot pose variables, $x_k$, $y_k$ and $\theta_k$, and velocities, $v_k$ and $\omega_k$. At each time-step the VT&R algorithm provides an estimate of the robot position relative to the nearest desired path pose, $\mathbf{x}_{d,j}$, by Euclidean distance.

of collected experiences (Rasmussen, 2006) using a gradient ascent algorithm. In order 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.

## 4. Implementation

### 4.1. Robot model

In this paper, robots are modeled as unicycle-type vehicles (Figure 6) with state $\mathbf{x}_k = (x_k, y_k, \theta_k)$. At every time-step, the VT&R localization algorithm provides the position of the robot, $\hat{\mathbf{x}}_k$, relative to the nearest desired pose by Euclidean distance (Figure 3). As such, we also define the lateral and heading path-tracking errors, $e_{L,k} = y_{d,k} - y_k$ and $e_{H,k} = \theta_{d,k} - \theta_k$, to be used when comparing results. Finally, the robots have two control inputs, their linear and angular velocities, $\mathbf{u}_k = (v_{\text{cmd},k}, \omega_{\text{cmd},k})$.

When the time between control signal updates is defined as $\Delta t$, the resulting nominal process model employed by the RC-LB-NMPC algorithm is

$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{x}_k + \Delta t \begin{bmatrix} \cos\theta_k & 0 \\ \sin\theta_k & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}_k \tag{24}$$

which represents a simple kinematic model for our robot; it does not account for dynamics or environmental disturbances. Instead, we depend on the learned model to account for these disturbances. As presented in previous work (Ostafew et al., 2015a), we enable the learned model (2) to represent a system with first-order dynamics by adding historic states and inputs to the disturbance dependency

$$\mathbf{a}_k = (\bar{\mathbf{x}}_k, \bar{\mathbf{v}}_{k-1}, \mathbf{u}_k, \mathbf{u}_{k-1}) \tag{25}$$

where $\bar{\mathbf{v}}_k = (\bar{v}_{\text{act},k}, \bar{\omega}_{\text{act},k})$, representing 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 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 RC-LB-MPC algorithm to learn these dynamics, as well as any other systematic disturbances, based on experience.

In order to build and query the learned model, $\mathbf{g}(\cdot)$, throughout the prediction horizon, we require all of the quantities in (25): $\mathbf{a}_{k+i} = (\bar{\mathbf{x}}_{k+i}, \bar{\mathbf{v}}_{k-1+i}, \mathbf{u}_{k+i}, \mathbf{u}_{k-1+i})$, $i = 0 \ldots K-1$. We know $\mathbf{u}_{k+i}$ and $\mathbf{u}_{k-1+i}$, as these are commanded inputs. We initially obtain the robot position from our vision-based localization system, $\mathbf{x}_k = \hat{\mathbf{x}}_k$, then from the Sigma-Point Transform, (17) and (18). Finally, we compute the velocity state variables, $\bar{\mathbf{v}}_{k-1+i} = (\bar{v}_{\text{act},k-1+i}, \bar{\omega}_{\text{act},k-1+i})$, based on the computed robot positions

$$\bar{v}_{\text{act},k-1+i} = \frac{\sqrt{(\bar{x}_{k+i} - \bar{x}_{k-1+i})^2 + (\bar{y}_{k+i} - \bar{y}_{k-1+i})^2}}{\Delta t} \tag{26}$$

$$\bar{\omega}_{\text{act},k-1+i} = \frac{(\bar{\theta}_{k+i} - \bar{\theta}_{k-1+i})}{\Delta t} \tag{27}$$

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

Since obstacle detection in unstructured terrain is still an open problem, lateral and heading tracking error limits are selected prior to experiments and a feasible, desired path is manually defined during the VT&R teach mode considering the given limits. Input constraints are then defined considering the maximum allowed linear and angular speeds, and the curvature of the desired path

$$v_{\max,k} = \frac{\omega_{\lim}}{\kappa_k + \omega_{\lim}/v_{\lim}} \tag{28}$$

and $\omega_{\max,k} = \omega_{\lim}$, where $\kappa_k$ is the path curvature and $v_{\lim}$ and $\omega_{\lim}$ are the maximum allowed linear and angular speeds. Input constraints are necessary in this work in order to prevent a situation where the algorithm attempts to compensate for a disturbance that physically cannot be addressed (i.e., model discrepancies caused by actuator limits).

### 4.2. Managing experiences

In order to ensure the RC-LB-NMPC algorithm is executed in constant computation time, our implementation requires the ability to use a subset of the observed experiences when computing a disturbance. In this work, we employ a local model based on a single sliding local model. As experiences are gained, 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 $v_{\text{bin}}$ represents the velocity discretization and $\lfloor \cdot \rfloor$ represents the floor function. When the number of experiences in a bin exceeds a threshold, $c_{\text{bin}}$, the oldest experience in the bin is discarded. Then, when computing a control input at the $i$th vertex, a 'local' dataset is created, drawing experiences from bins at nearby path vertices and commanded velocities, $\mathcal{D} = \{\mathcal{D}_{a,b} | \, a \in \{i - c_{\text{vertex}}, \ldots, i + c_{\text{vertex}}\}, b \in$

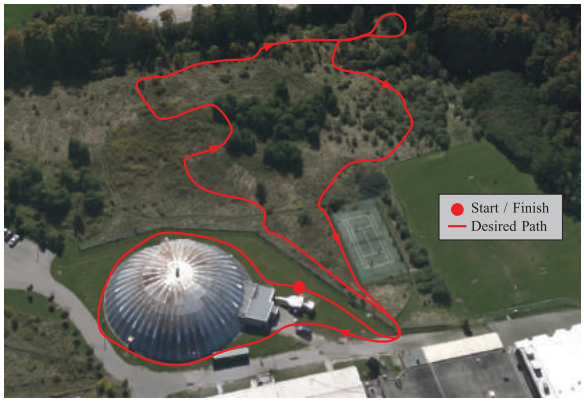


**Fig. 7.** Testing culminated in the third experiment where the path, shown here, was defined around the University of Toronto Institute for Aerospace Studies (UTIAS) campus. The 900-m-long path passed in between trees and structures and over vegetation, pavement, inclines, and side-slopes. (Imagery: Google)

$\{l - c_{\text{velocity}}, \ldots, l + c_{\text{velocity}}\}\}$, based on thresholds $c_{\text{vertex}}$ and $c_{\text{velocity}}$. 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. Experimental testing

### 5.1. Overview

We tested the RC-LB-NMPC algorithm on a 50 kg Clearpath Husky and a 900 kg Clearpath Grizzly in three different experiments with many different surface materials and topographies. This resulted in over 5 km of path-tracking by the RC-LB-NMPC algorithm. The first and second experiments (Sections 5.3 and 5.4) compared unconstrained LB-NMPC, constrained LB-NMPC (C-LB-NMPC), and the proposed RC-LB-NMPC algorithm. The three variants solved the same optimization problem (7) with the following two exceptions. Firstly, state constraints (5) were disabled for the LB-NMPC algorithm. Secondly, the C-LB-NMPC algorithm included only non-robust state constraints (i.e., $\alpha = 0$ when computing (5)). The third experiment (Section 5.5) tested the RC-LB-NMPC algorithm over the course of five trials on an outdoor 900-m-long path with pavement, dirt, sand, grass, inclines, and side slopes (Figure 7). The three experiments demonstrate the presented algorithm's ability to guarantee constraint satisfaction while improving performance through learning.

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 control signals were generated at approximately 10 Hz. As previously mentioned, hyperparameter selection is currently an offline process, taking up to five minutes for 5,000 accumulated experiences. Since GPS was not available, improvements due to the RC-LB-NMPC algorithm were quantified by the localization provided by 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 et al., 2013).

### 5.2. Tuning parameters

The performance of the system was adjusted using the NMPC weighting matrices $\mathbf{Q}$ and $\mathbf{R}$, the confidence level $l$ and the experience management parameters. The weighting matrices were selected in advance with a 25:1:10 ratio weighting path-tracking errors, angular control inputs, and linear control inputs for the 50 kg Husky and a 25:5:10 ratio for the 900 kg Grizzly robot. The increased weighting on the Grizzly inputs was selected to ensure controller stability at higher speeds. The prediction horizon $K$ was set to 10, giving the robot one second to slow down at a reasonable rate when required. The selected confidence level was $l = 0.997$, resulting in $\alpha = 3$ (i.e., three standard deviations). Local GP models were generated based on a sliding window of size, $c_{\text{vertex}} = 5$ and $c_{\text{velocity}} = 1$, where velocities were discretized by $v_{\text{bin}} = 0.25$ m/s. The maximum number of experiences per bin, $c_{\text{bin}}$, was set to four, resulting in local models based on up to 180 experiences.

### 5.3. Experiment 1: Indoor algorithm comparison

The first experiment compared three algorithms (unconstrained LB-NMPC, C-LB-NMPC, and RC-LB-NMPC) over three trials using a 50 kg Clearpath Husky robot on an indoor, flat, concrete surface (Figure 9). As previously mentioned, the algorithms solve the same optimization problem (7) at every time-step, considering identical tuning parameters with the exception of constraints. Since the concrete did not develop tire ruts and the lighting did not change over the course of the experiment, these results provide a clear comparison of the algorithms.

We show the maximum tracking errors and travel times vs. trial in Figure 8. The unconstrained LB-NMPC algorithm results in the largest tracking errors in the first trial, decreasing in later trials through learning. By adding constraints, the C-LB-NMPC algorithm reduced the errors during the first trial, but was overconfident and failed to satisfy the lateral constraints. On the other hand, the RC-LB-NMPC algorithm resulted in constraint satisfaction throughout all trials but incurred increased travel time relative to the other algorithms when the model was uncertain (i.e., during the first trial). Figure 10 shows tracking errors and control inputs vs. distance along the path for all trials. The LB-NMPC and C-LB-NMPC algorithms showed lateral constraint violations at 7, 16, and 24 m along the path (i.e., the turns) in the first trial. The RC-LB-NMPC algorithm avoided such constraint violations in the first

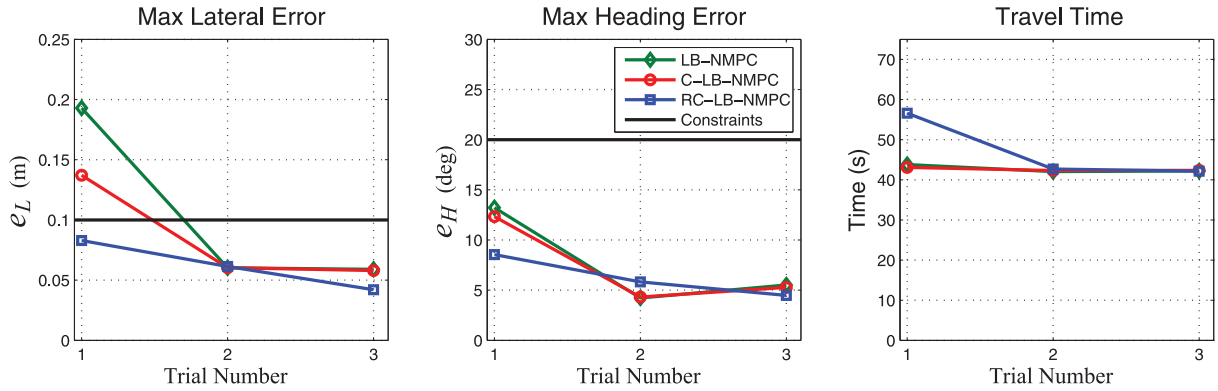


**Fig. 8.** The maximum tracking errors and travel times vs. trial for the three algorithms in experiment 1. The RC-LB-NMPC algorithm results in constraint satisfaction at the cost of increased travel time (i.e., slower speed) during the first two trials.

trial partly by lowering the speed to approximately 0.5 m/s throughout the path. This speed represents roughly half of the maximum speed possible by the Husky robot. The ability of the RC-LB-NMPC algorithm to optimally compute the linear and angular speeds of the robot in real time while robustly meeting constraints represents one of the advantages of this algorithm. In practice, control algorithms must be capable of reacting robustly to given state constraints in tandem with scheduled speeds. This experiment also shows how the RC-LB-NMPC algorithm produces optimal results similar to the unconstrained algorithm after the learned model has collected experience. In this way, the RC-LB-NMPC algorithm behaves conservatively when the learned model is uncertain, and optimally considering the given cost function when the model uncertainty has been reduced through learning.

## 5.4. *Experiment 2: Off-road algorithm comparison*

The second experiment once again compared the three algorithms (LB-NMPC, C-LB-NMPC, and RC-LB-NMPC). However, in this experiment we tested with a 900 kg Clearpath Grizzly robot on a more challenging and realistic 900-m-long, off-road path (Figure 12) in the University of Toronto Institute for Aerospace Studies (UTIAS) MarsDome. In this experiment, the Grizzly was limited to 2.0 m/s considering path roughness. Since the path was on loose material and the Grizzly is quite heavy, significant ruts developed over the course of the experiment, resulting in evolving disturbances from trial to trial. In this situation, we rely on the learned model uncertainty to capture the effect of the evolving disturbances, and only the RC-LB-NMPC algorithm uses the learned uncertainty to guarantee constraint satisfaction. We show the maximum tracking errors and travel times vs. trial in Figure 11. Without state constraints the LB-NMPC algorithm results in the fastest travel time and also the highest lateral errors of the first trial. During later trials, the errors are reduced through learning, with no notable changes in travel time. By adding constraints, the

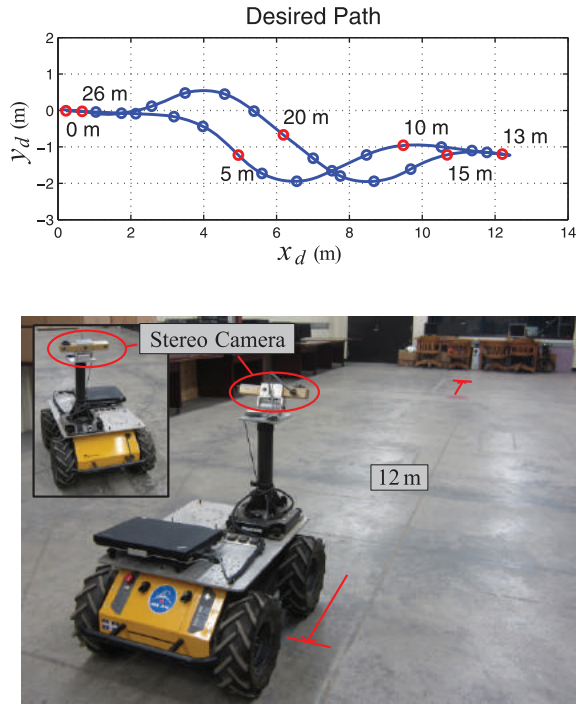


**Fig. 9.** (Top) The desired path and (bottom) the 50 kg Clearpath Husky at the start of the experiment 1 path. The smooth concrete floor and constant lighting providing identical path conditions for all trials.

C-LB-NMPC algorithm reduced the errors during the first trial but still failed to satisfy the lateral constraints. As with experiment 1, the RC-LB-NMPC algorithm resulted in constraint satisfaction throughout all trials and decreased travel time through learning. This confirmed our goal of providing guaranteed constraint satisfaction while improving performance through learning. Figure 13 shows tracking errors and control inputs vs. distance along the path for trials 1, 2, and 10. Without constraints, the LB-NMPC algorithm resulted in the highest speeds and errors. The constraints led the C-LB-NMPC algorithm to reduce speeds in general but

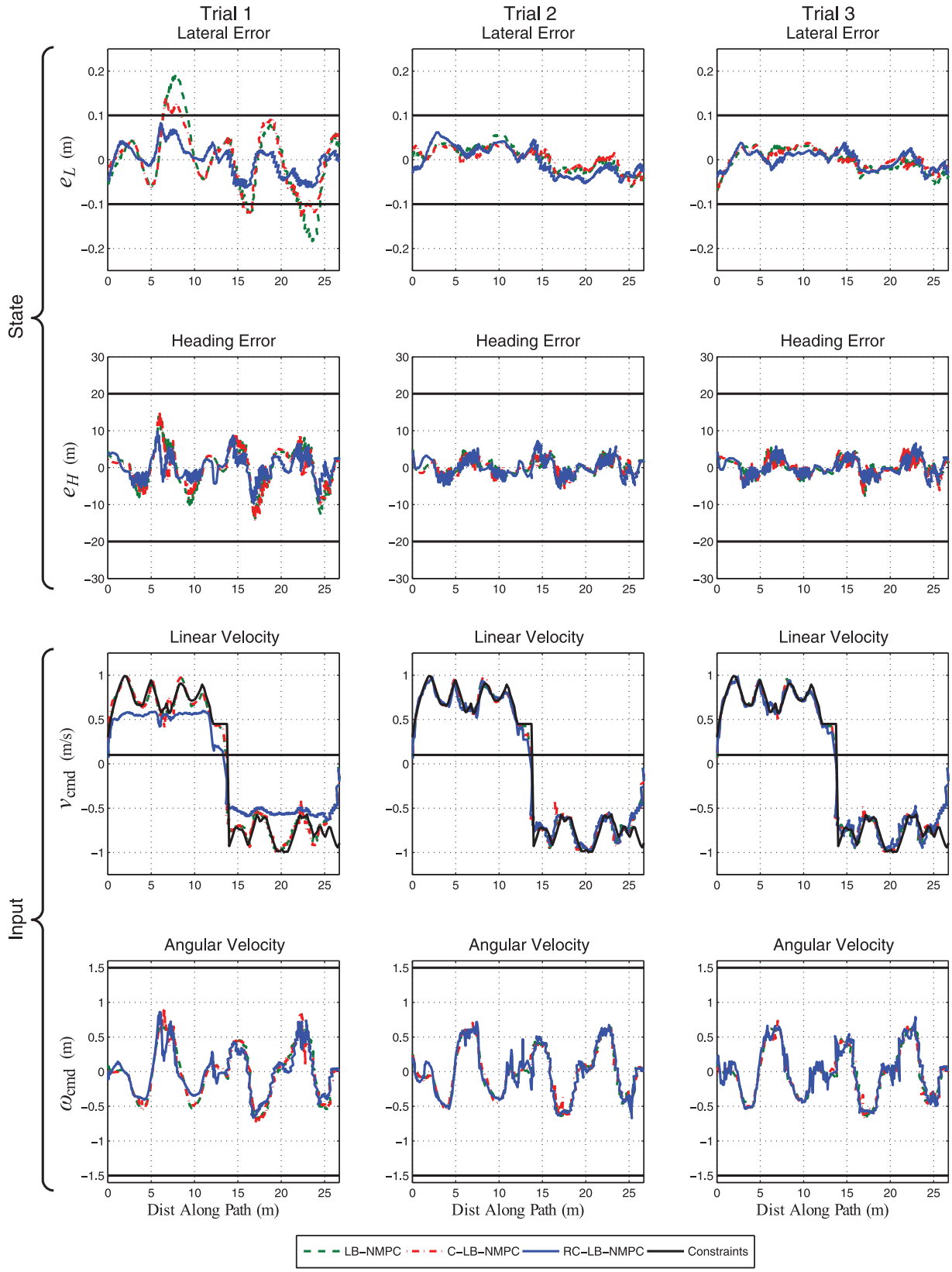


**Fig. 10.** Tracking errors and commanded inputs vs. distance during the first, second, and third trials of experiment 1. The RC-LB-NMPC algorithm automatically reduces speed in order to meet lateral and heading constraints during the first trial. As model uncertainty is reduced through learning, the RC-LB-NMPC algorithm naturally increases speed and thus performance.

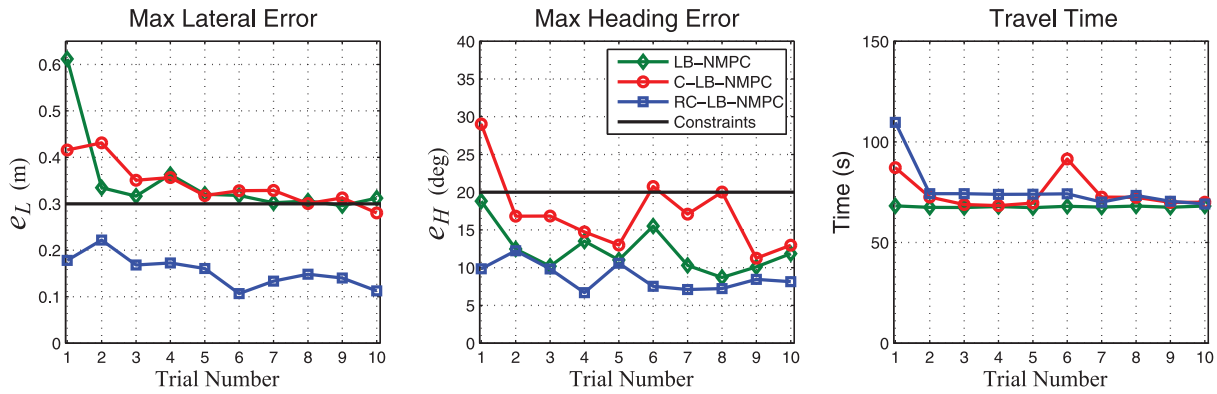


**Fig. 11.** The maximum tracking errors and travel times vs. trial from experiment 2. Of the three algorithms, RC-LB-NMPC is the only one to respect constraints in all trials at the cost of higher travel time.

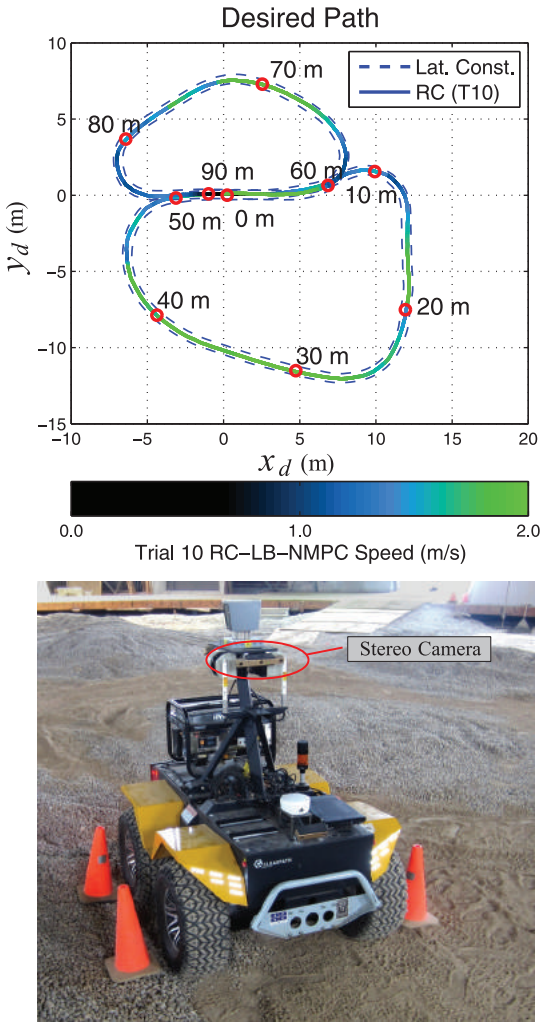


**Fig. 12.** (Top) Lateral constraints outlining the experiment 2 path, along with the RC-LB-NMPC actual path and speed (trial 10). (Bottom) The 900 kg Clearpath Grizzly at the path start.

the algorithm was overconfident and exceeded limits during each trial. Specifically, the turn at 65m is an example where significant ruts developed, causing wheel slip. Since the RC-LB-NMPC algorithm robustly incorporated learned model uncertainty, the robot successfully passed through this section during each trial. Figure 14 shows measured and predicted disturbances used by the RC-LB-NMPC algorithm. The most complex disturbances affected the angular state of the robot, $g_{(3)}$.

## 5.5. *Experiment 3: Field test*

Finally, the third experiment thoroughly tested the RC-LB-NMPC algorithm on a 900-m-long, off-road path repeated five times (Figures 7 and 15). The third path covered a wide range of surfaces, including grass, dirt, pavement, side slopes, and inclines while passing through trees, solid structures, and dense foliage. Once again, we tested with a 900 kg Clearpath Grizzly robot limited to 2.0 m/s considering the path roughness. We show that over the course of the experiment, the RC-LB-NMPC algorithm met constraints and improved performance over sequential trials (Figure 16). Moreover, the experiment shows that the algorithm as formulated was able to consistently deliver control input updates at 10 Hz despite the relatively large dataset gathered over the course of the five trials. Specifically, the learned model had accumulated over 25,000 experiences by the fifth trial, relying on the experience management scheme to extract up to 180 relevant experiences at any given time-step. Figure 17 shows tracking errors and control inputs vs. distance along the path for trials one, two, and five. The first trial is representative of a conservative, non-learning RC-NMPC algorithm: without experience, the predicted disturbance is zero at every time-step but with high uncertainty, capturing the wide range of possible disturbances. After just one trial, the learned model has collected enough experience to significantly reduce uncertainty and increase performance. Modeling disturbances as a GP enables efficient interpolation and extrapolation from data collected during previous trials. Finally, Figure 18 shows tracking error and velocity distributions from trials one and five. The results show that over the course of the

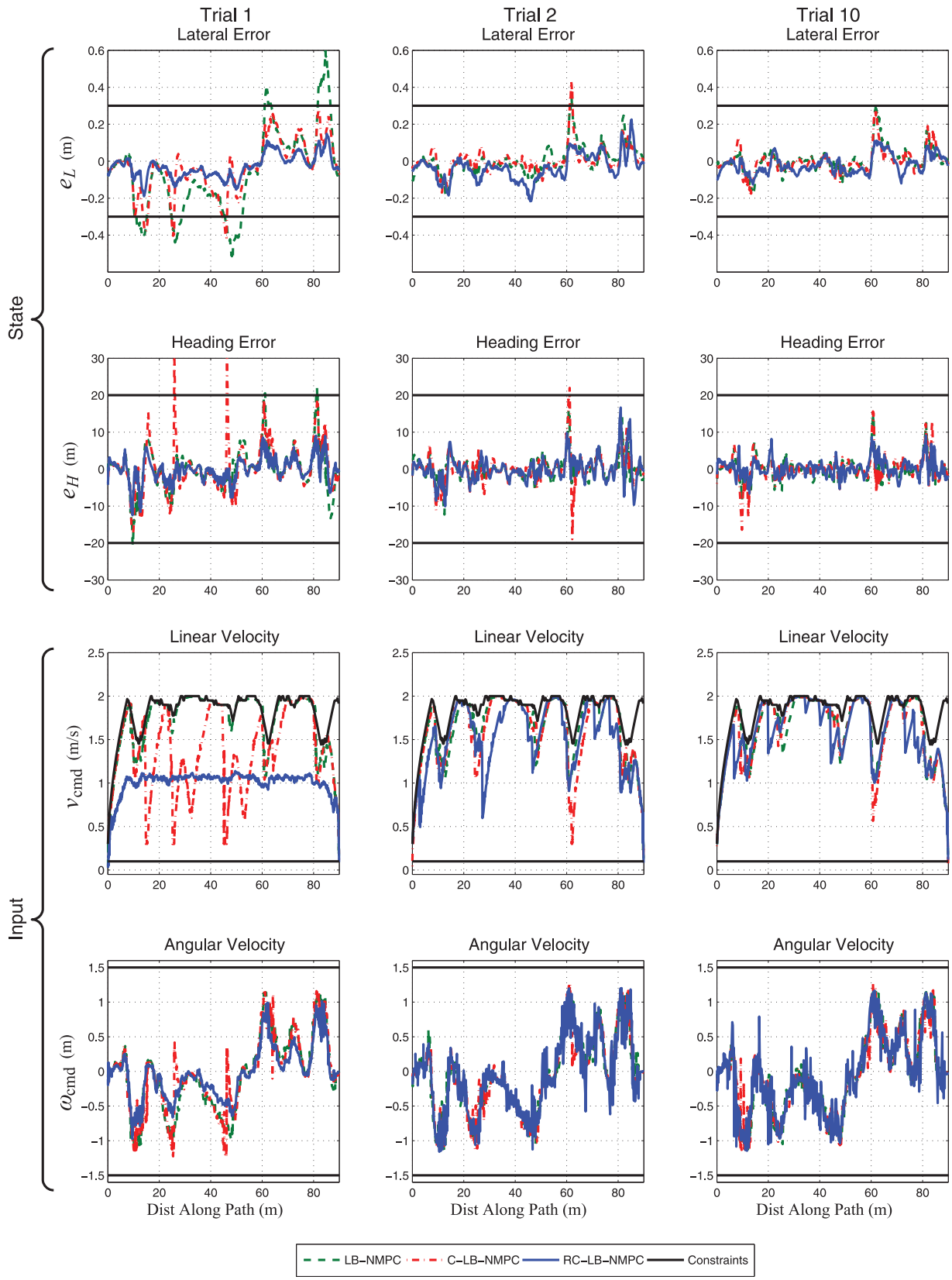


**Fig. 13.** Tracking errors and commanded inputs vs. distance during the first, second, and tenth trials of experiment 2. The most challenging turn occurred at 60m along the path, where significant ruts developed in the loose gravel.

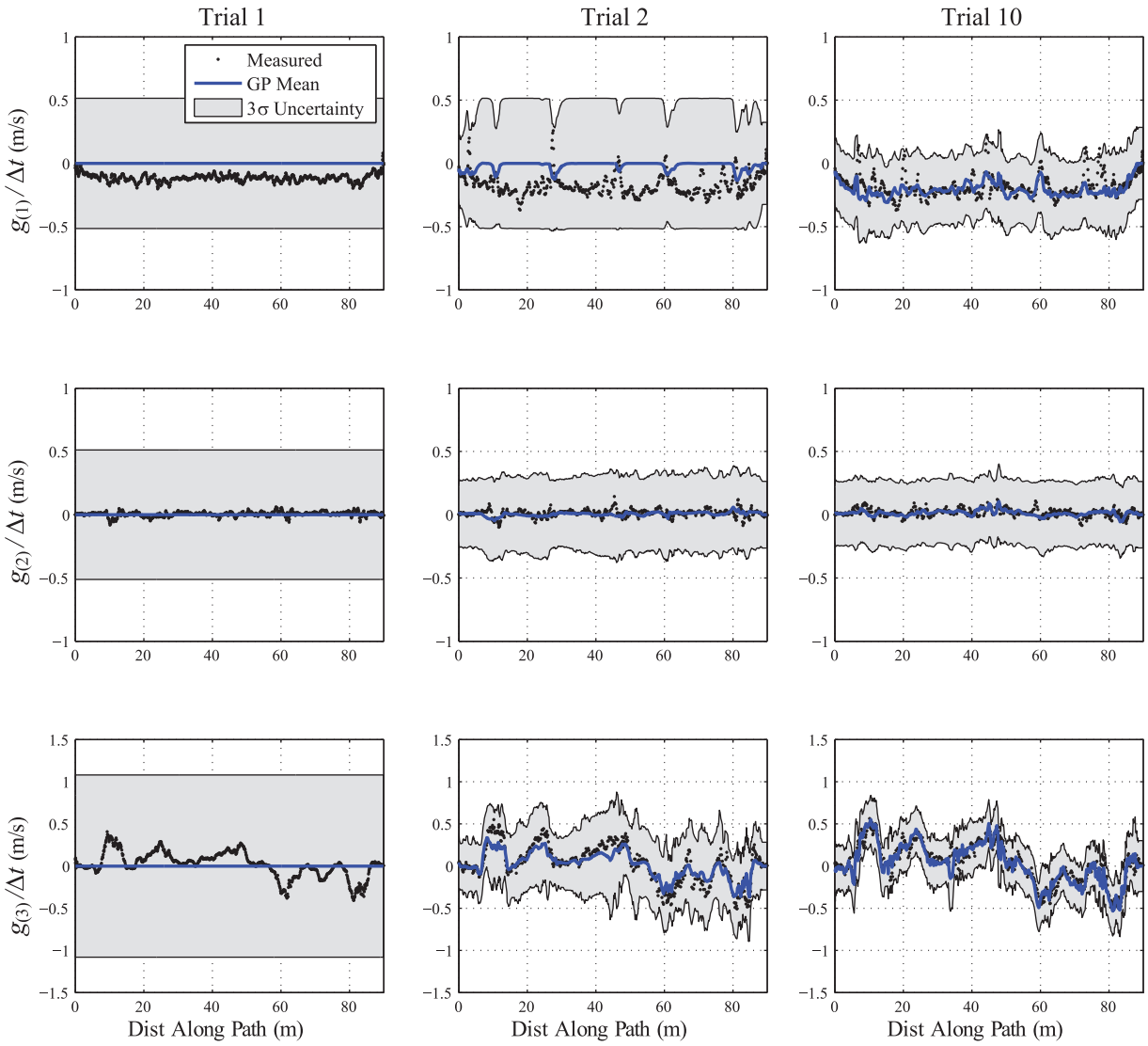


**Fig. 14.** Modeled disturbances, $\mathbf{g}(\cdot)=(g_{(1)}(\cdot),g_{(2)}(\cdot),g_{(3)}(\cdot))$, representing disturbances affecting the forward, lateral, and angular velocities of the robot during the first, second, and tenth trials during the first, second, and tenth trials of the second experiment. As the algorithm gathers experience, the model uncertainty decreases and model accuracy increases.

experiment, the lateral error distribution widens but continues to fit within the limits. In practice, we do not expect the tracking errors to go to zero, since the RC-LB-NMPC algorithm is an optimal control algorithm, balancing tracking errors and control inputs subject to the constraints.

## 6. Discussion

This work combines concepts of RC-MPC and machine learning in a flexible way. Through extensive experimental results, we have shown practical operation considering a relatively large robot, challenging terrain, and paths requiring tight state constraints. With respect to future work, our modular approach provides the opportunity to investigate and improve both the underlying GP disturbance model and the RC-NMPC algorithm.

With respect to the learned model, Jordan and Mitchell (2015) present a survey of recent trends and prospects for machine learning. Specifically, they highlight the opportunity and need for further research into team-based learning. The goal is to identify effective methods of transferring data between robots such that any improvements found by one robot can be shared by all robots in a team. As opposed to our experiments demonstrating a single robot tracking a single path, one could imagine a large network of paths with many robots improving performance collectively and safely. Also, one of our key requirements for real-time operation is the ability to rapidly identify relevant experiences for our local disturbance model. In our work, experiences are selected based on the nearest path vertex and recent commanded velocity and old experiences are discarded in

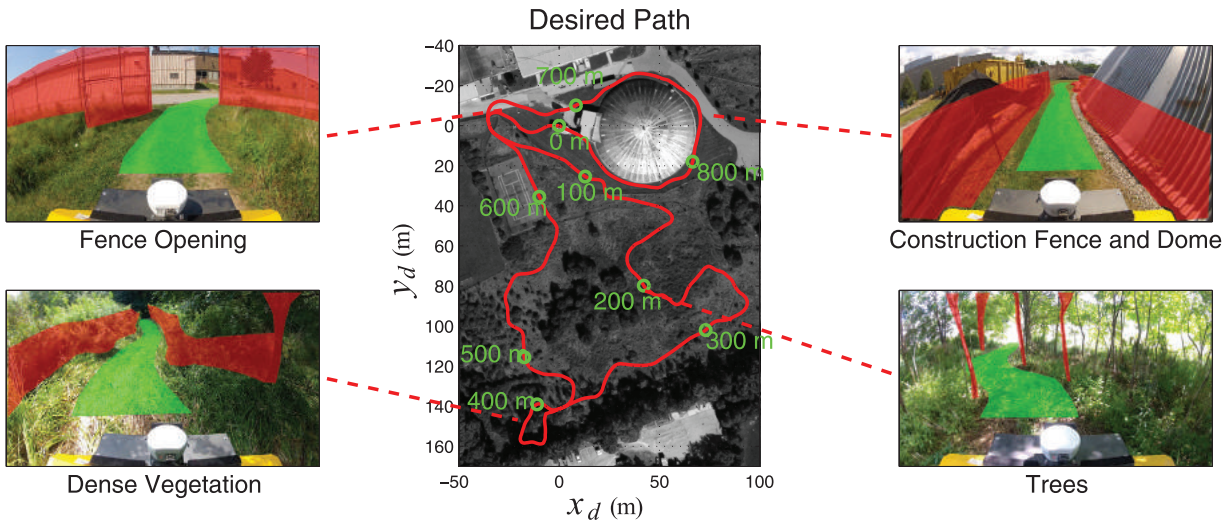


**Fig. 15.** (Center) The experiment 3 desired path roughly overlaid on the UTIAS campus. (Left/Right) images showing path sections with constraint-satisfying terrain (green) and unsafe obstacles (red) highlighted. Without an obstacle detection algorithm, the desired path is manually taught with the required clearance and safe/unsafe areas shown are manually highlighted for illustration purposes. (Satellite Imagery: Google)

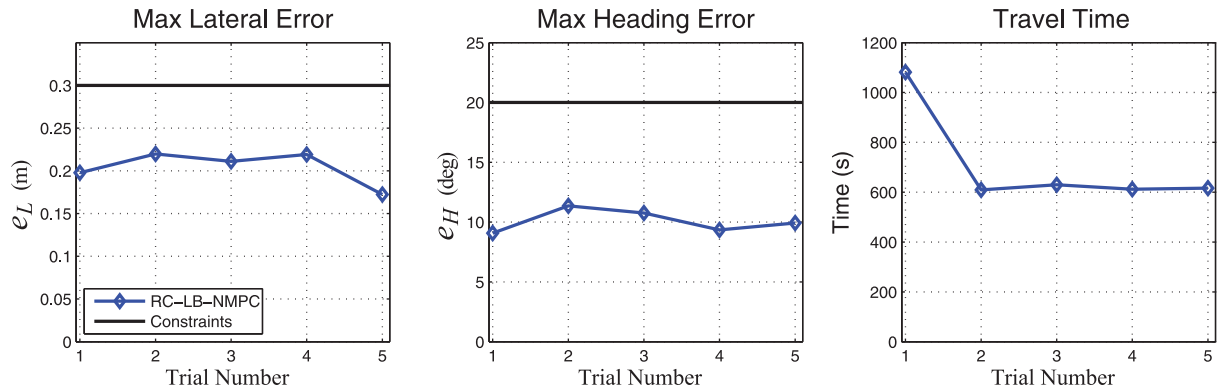


**Fig. 16.** The maximum tracking errors and travel times vs. trial for experiment 3. Tracking errors met the given constraints while learning reduced the travel time by almost 50%.

order to maintain a manageable database. However, further work could involve evaluating the sensitivity of the learned model to this selection process and proposing improvements. For example, we do not currently address the situation where disturbances change predictably over time, such as the arrival of snow or puddles. As such, our algorithm will likely be overconfident driving a snowy path for the first time if the algorithm had previously learned the path disturbances during the summer. Finally, convergence of the learned model is also an open question. In practice, convergence rates are complicated by the evolution of the environment due to the robot's activity and daily variations. For example, repeating the same path caused ruts to form, which resulted in a change in the disturbances affecting the nominal process model. In such conditions, it is unclear when the algorithm will converge. In general, it is expected that improvements to the disturbance model will result in the largest improvements to overall algorithm performance and reductions in computational complexity.

With respect to the RC-NMPC algorithm, future work includes further integration of the state constraints with a real-time obstacle detection algorithm and investigations into the conservativeness of the algorithm. As previously mentioned, one key benefit of our algorithm is that disturbances and robust constraints are computed in real-time. As such, future work on incorporating new or dynamic obstacles in the optimization problem would be of use for robots exploring new environments or operating in dynamic environments. Secondly, conservativeness of robust algorithms is an ongoing topic for MPC research in general. As can be seen in Figures 10, 13, and 17, the tracking errors remain relatively far from the actual constraints. Reductions in conservativeness will allow for increases in the performance of practical robots.

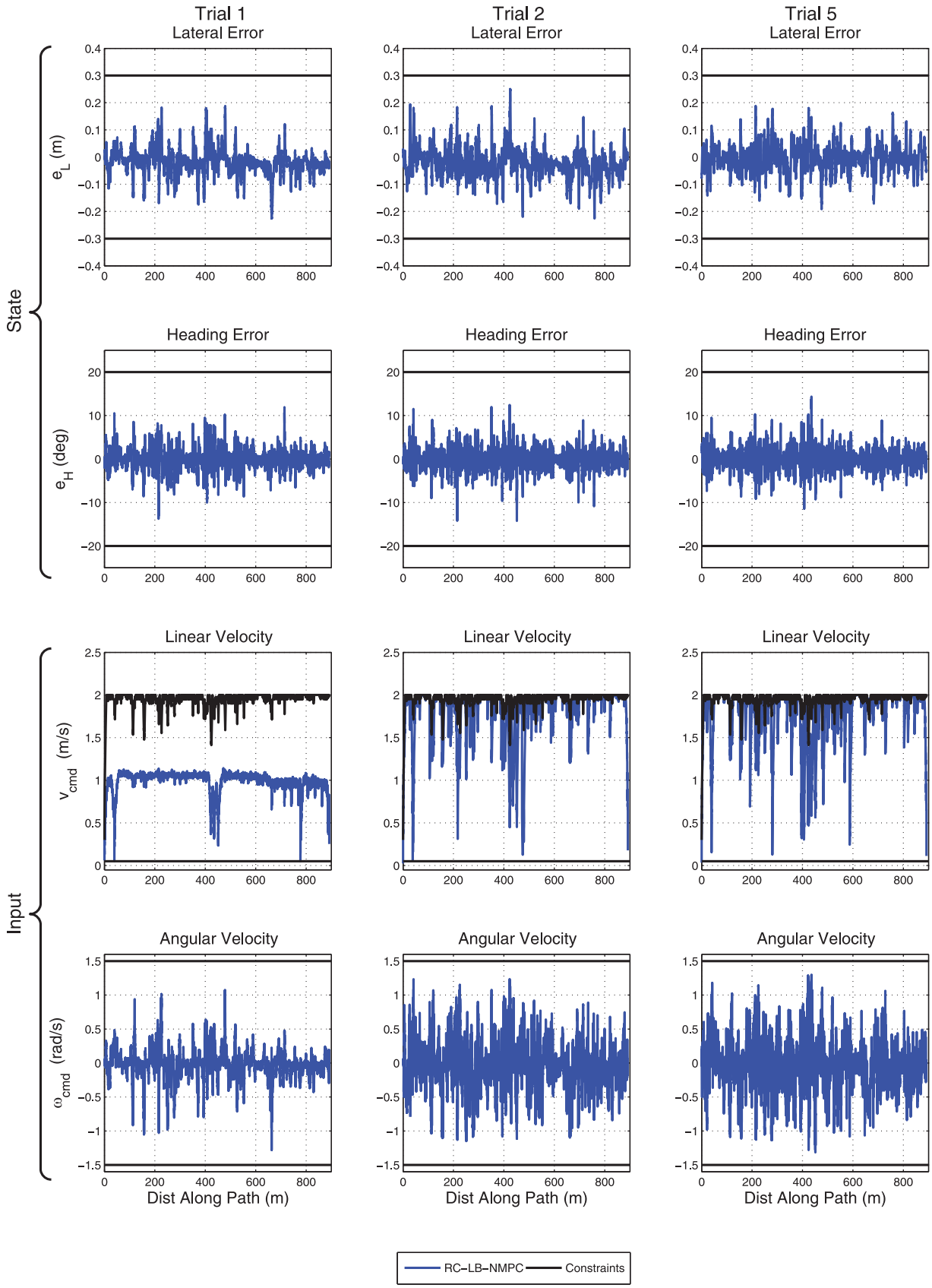


**Fig. 17.** Tracking errors and commanded inputs vs. distance during the first, second, and fifth trials of experiment 3. Results in the first trial mimic that of a non-learning RC-NMPC algorithm with high model uncertainty representing all possible disturbances and thus conservative inputs.

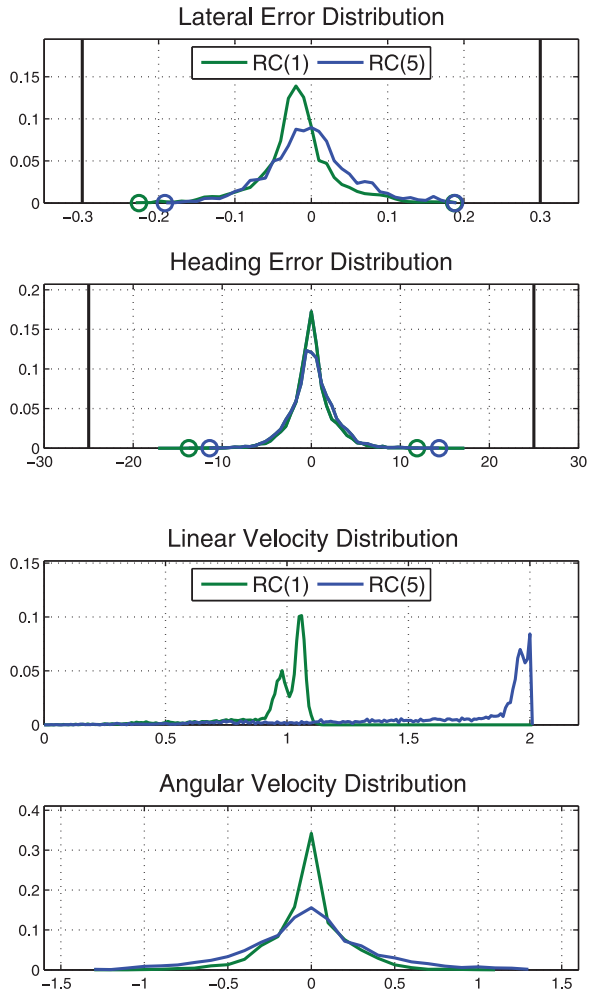


**Fig. 18.** Error and velocity distributions for trials one and five of experiment 3. Learning resulted in a relatively wider distribution of lateral errors as the algorithm became more confident. In practice, the algorithm is seeking to balance tracking errors and control inputs subject to the given contraints. As a result, we do not expect that tracking errors will be driven to zero.

## 7. Conclusion

In summary, this paper presents a Robust Constrained Learning-based Nonlinear Model Predictive Control (RC-LB-NMPC) algorithm for a path-repeating, mobile robot operating in challenging off-road terrain. The goal is to guarantee constraint satisfaction while increasing performance through learning. In previous work, we demonstrated unconstrained LB-NMPC, where tracking errors were reduced using real-world experience instead of pre-programing accurate analytical models (Ostafew et al., 2015a). This work represents a major extension where we use the learned model uncertainty to compute and apply robust state and input constraints.

The RC-LB-NMPC algorithm uses a fixed, simple robot model and a learned, nonparametric disturbance model. Disturbances represent measured discrepancies between the *a priori* model and observed system behavior. We use a Sigma-Point Transform to efficiently compute the mean and variance of predicted state sequences given the two-component, learned, stochastic model. Finally, we apply restricted constraints to the mean predicted sequence such that the $3\sigma$ confidence region is contained within the desired constraints. Localization for the controller is provided by an on-board, vision-based mapping and navigation system enabling operation in large-scale, off-road environments. We provide extensive experimental results comparing unconstrained LB-NMPC, constrained LB-NMPC, and the RC-LB-NMPC algorithm, using two significantly different robots and over 5 km of off-road travel. The results show that the RC-LB-NMPC algorithm efficiently provides real-time robust constraint satisfaction and performance improvements over sequential trials through learning.

### Acknowledgements

This research was funded by the Ontario Ministry of Research and Innovation's 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.

### References

Aswani A, Gonzalez H, Shankar Sastry S, et al. (2013) Provably safe and robust learning-based model predictive control. *Automatica* 49: 1216–1226.

Boyd S and Vandenberghe L (2004) *Convex Optimization*. Cambridge: Cambridge University Press.

Erez T, Lowrey K, Tassa Y, et al. (2013) An Integrated system for real-time model predictive control of humanoid robots. In: *Proceedings of the IEEE-RAS international conference on humanoid robots (humanoids)*, pp.292–299.

Farrokhsiar M, Pavlik G and Najjaran H (2013) An integrated robust probing motion planning and control scheme: A tube-based mpc approach. *Robotics and Autonomous Systems* 61(12): 1379–1391.

Furgale P and Barfoot TD (2010) Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics* 27(5): 534–560.

González R, Fiacchini M, Guzmán JL, et al. (2011) Robust tube-based predictive control for mobile robots in off-road conditions. *Robotics and Autonomous Systems* 59(10): 711–726.

Gouaillier D, Collette C and Kilner C (2010) Omni-directional closed-loop walk for NAO. In: *Proceedings of the IEEE-RAS international conference on humanoid robots (humanoids)*, pp.448–454.

Hehn M and D'Andrea R (2014) A Frequency domain iterative learning algorithm for high-performance, periodic quadrocopter maneuvers. *Mechatronics* 24(8): 954–965.

Howard TM, Green CJ and 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.

Jordan M and Mitchell T (2015) Machine learning: Trends, perspectives, and prospects. *Science* 349(6245): 255–260.

Julier SJ and Uhlmann JK (2004) Unscented filtering and nonlinear estimation. *Proceedings of the IEEE* 92(3): 401–422.

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

Krenn A, Gibbesch G, Binet G, et al. (2013) Model predictive traction and steering control of planetary rovers. In: *Proceedings of the 12th symposium on advanced space technologies in robotics and automation*, Noordwijk, The Netherlands, pp.1–8.

Kühne F, Lages WF and Silva JMG (2005) Mobile robot trajectory tracking using model predictive control. In: *Proceedings of the Latin-American robotics symposium*, pp.1–7.

Kumar V and Michael N (2012) Opportunities and challenges with autonomous micro aerial vehicles. *International Journal of Robotics Research* 31(11): 1279–1291.

Lafaye J, Gouaillier D and Wieber PB (2014) Linear model predictive control of the locomotion of pepper, a humanoid robot with omnidirectional wheels. In: *Proceedings of the IEEE-RAS international conference on humanoid robots (humanoids)*, pp.336–341.

Langson W, Chryssochoos I, Raković SV, et al. (2004) Robust model predictive control using tubes. *Automatica* 40(1): 125–133.

Mayne DQ (2014) Model predictive control: recent developments and future promise. *Automatica* 50(12): 2967–2986.

Mayne DQ, Kerrigan EC, Van Wyk E, et al. (2011) Tube-based robust nonlinear model predictive control. *International Journal of Robust and Nonlinear Control* 21(11): 1341–1353.

Meier F, Hennig P and Schaal S (2014) Efficient bayesian local model learning for control. In: *Proceedings of the IEEE international conference on intelligent robotics systems*, pp.2244–2249.

Moore KL (2012) *Iterative Learning Control for Deterministic Systems*. Berlin: Springer Science & Business Media.

Mueller MW and D'Andrea R (2013) A model predictive controller for quadrocopter state interception. In: *Proceedings of the IEEE european control conference*, pp.1383–1389.

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

Nguyen-Tuong D, Peters J and Seeger M (2008) Local gaussian process regression for real time online model learning. In: *Proceedings of the neural information processing systems conference 21*, pp.1193–1200.

Ostafew C, Collier J, Schoellig AP, et al. (2015a) Learning-based nonlinear model predictive control to improve vision-based mobile robot path tracking. *Journal of Field Robotics* 33(1): 133–152.

Ostafew C, Schoellig A and Barfoot T (2015b) Conservative to confident: treating uncertainty robustly within learning-based control. In: *Proceedings of the IEEE conference on robotics and automation*, pp.421–427.

Ostafew C, Schoellig, AP and 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 international conference on intelligent robots and systems*, pp.176–181.

Peters S and Iagnemma K (2008) Mobile robot path tracking of aggressive maneuvers on sloped terrain. In: *Proceedings of the international conference on intelligent robots and systems*, pp.242–247.

Rasmussen CE (2006) *Gaussian Processes for Machine Learning*. Cambridge: The MIT Press.

Rawlings JB and Mayne DQ (2009) *Model Predictive Control: Theory and Design*. WI, USA: Nob Hill Publishing.

Schaal S and Atkeson CG (2010) Learning control in robotics. *IEEE Robotics & Automation Magazine* 17(2): 20–29.

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

Sigaud O, Salaün C and Padois V (2011) On-line regression algorithms for learning mechanical models of robots: A survey. *Robotics and Autonomous Systems* 59(12): 1115–1129.

Stenning B, McManus C and 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.